Sentiment-Aware Conversational Agent

Isabel de Lima Henriques Dias

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisor(s): Prof. Maria Luísa Torres Ribeiro Marques da Silva Coheur
Prof. Ricardo Costa Dias Rei

November 2021

Acknowledgments

First and foremost, I would like to thank my supervisors, Prof. Luísa Coheur and Prof. Ricardo Rei, for allowing and trusting me to pursue my ideas for this work, and for all the meaningful discussions, guidance and validation, which were crucial for the development of this thesis.

I would like to thank my friends, who through these weird times have always been by my side. In particular, I want to thank my friend Rita, for how much we have grown together, from the day we both started at Técnico to today. I am incredibly proud of us.

I am also thankful to my aunts, uncles, cousins, brother, and grandmother for their unconditional friendship and support. A special thanks to my cousins Kika and Leonor, for being my surf mates, for listening to me during stressful times, and for always making me laugh.

To my parents, thank you for being such an inspiration of hardworking and ever-growing people, and for always believing and steering me in the right direction throughout my life.

Lastly, and most importantly, I want to thank Pedro for bringing out the best in me, teaching me how to be fearless, cheering me on during the toughest times, and for the constant patience and caring all these years. I could not have done this without you.

Resumo

Os modelos mais recentes de geração automática de texto no contexto de diálogo dependem de grandes quantidades de dados para conseguirem implicitamente gerar texto fluente e apropriado. Algumas aplicações, como apoio ao cliente, começaram a usar agentes automáticos para manter e aumentar a confiança dos seus clientes, levando a uma resolução mais rápida e eficaz de problemas. No entanto, existem poucos dados disponíveis para este tipo de aplicações, o que pode dificultar o seu treino, resultando em respostas genéricas. Este aspeto é problemático, dado que uma resposta que não tenha em conta a emoção, pode causar o descontentamento dos clientes.

Neste trabalho propomos abordar estes problemas com um agente conversacional que tem em conta sentimento. Para o fazer vamos desenvolver três modelos: um modelo de classificação de sentimento; um modelo para prever o sentimento da frase seguinte, que considera o contexto do diálogo para prever um sentimento para o agente expressar na sua resposta; e um modelo de geração, que é condicionado com a previsão do sentimento e o contexto do diálogo, para conseguir gerar uma resposta que é apropriada em termos do contexto e do sentimento.

Quer a avaliação humana quer a avaliação automática, mostram que guiar explicitamente o modelo de geração com um conjunto de frases pré-definido, leva a claras melhorias, em particular para modelos treinados com conjuntos de dados pequenos. Finalmente, mostramos que o modelo de previsão do sentimento da frase seguinte é o maior obstáculo do sistema, e discutimos futuras possíveis maneiras de o melhorar.

Palavras-chave: Processamento de Língua Natural; Classificação de Sentimento; Previsão do Sentimento da Frase Seguinte; Geração de Texto Condicionado.

Abstract

Current state-of-the-art dialogue text generation models rely on large amounts of data in order to implicitly learn how to generate fluent and appropriate text. Some applications, such as customer support, have started to rely on such systems to retain and increase the confidence of customers with a fast and effective resolution of possible problems. However, the data available for such applications is often scarce, which might not allow to properly train these models, leading to automatic generic answers, which is problematic, since sentiment is often regarded as an important aspect of customer satisfaction.

We propose to tackle these issues by developing an end-to-end sentiment-aware conversational agent. To do so, we will develop three models: a sentiment classification model, tasked with classifying the sentences of the dialogue; a reply sentiment prediction model, which leverages the context of the dialogue in order to predict an appropriate sentiment for the agent to express in its reply; and a text generation model, which is conditioned on the predicted sentiment and the context of the dialogue, in order to produce a reply that is both context and sentiment appropriate.

Both automatic metrics and human evaluation show that explicitly guiding the text generation model with a pre-defined set of sentences leads to clear improvements, in particular for models fine-tuned with small datasets. Finally, we show that the reply sentiment prediction model is the bottleneck of the system, and discuss future approaches.

Keywords: Natural Language Processing; Sentiment Classification; Reply Sentiment Prediction; Conditioned Text Generation.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Conversational agents have become popular over the years in various forms, such as personal assistants, or as an automatic way of a company to provide information to its customers, among others.

The first conversational agents that emerged, like ELIZA (Weizenbaum, 1966), had the goal of simulating a conversation between a human and a machine. These systems were rule based, and “generated” sentences by pattern matching the input sentences by the user with a pre-defined set of rules. The ability of the model of engaging in discourse, and of rephrasing the user’s input with its own pre-defined set of rules, made the users believe they were speaking with an actual person, many claiming the software was able to understand them and show empathy and intelligence. However, such rule-based systems are not scalable, since each rule has to be individually defined by a human.

Other approaches took advantage of information retrieval in order to fundamentally change the way conversational agents interacted with knowledge bases (Leuski et al., 2006; Leuski and Traum, 2011; Ji et al., 2014). Distance-based similarity metrics, such as Jaccard and Dice distances, and information retrieval statistical metrics, such as TF-IDF (Jurafsky and Martin, 2009), allowed these systems to compare the user’s query with the already known queries stored in its knowledge base, and rank them in order to answer with the most appropriate reply. These systems work particularly well in specific domains, such as customer support, where users usually have questions about particular products or services. An interesting historical overview that goes more in depth on the evolution of conversational agents is discussed in Pereira et al. (2019).

With the emergence of social networks, which made dialogue data become more available, and the increase of computational power, Machine Learning and, in particular, Deep Learning approaches, were able to improve the performance of dialogue conversational agents through the use of neural network models and word embeddings (LeCun et al., 2015; Vinyals and Le, 2015; Shang et al., 2015). These methods remove the manual effort of building rules and knowledge bases we have described in the aforementioned works. In contrast, they require large amounts of data in order to produce grammatically correct and adequate answers. In particular, Transformer models (Vaswani et al., 2017) and dense contextual word embeddings (Mikolov et al., 2013; Peters et al., 2018) revolutionized the way models learn text characteristics, which enables them to produce lexically diverse and context appropriate answers. For example, Rashkin et al. (2019) shows that training these models in a large empathetic corpus enables them to generate text that expresses emotions. However, Miao et al. (2020) demonstrates that it is hard to achieve good generation results when fine-tuning current models with small datasets. Furthermore, in Rashkin et al. (2019) it is shown that models trained on “spontaneous internet conversation data are not rated as very empathetic”. The problem is that for specific-domains, such as customer support, the amount of data can be scarce or not publicly available due to privacy constraints. Additionally, emotion-labelled corpora, in particular with a more fine-grained set of labels, is difficult to build in a large scale. Existing works have started to find ways to explicitly condition text generation models in order to produce certain attributes, such as emotion (Zhou et al., 2018; Huang et al., 2018), personality traits (Wolf et al., 2019), among others (Dathathri et al., 2019; Keskar et al., 2019). The drawback of these approaches is that they can not be used in a dialogue setting, given that they do not introduce a mechanism to automatically predict the next appropriate attribute, as discussed in Xie and Pu (2021), and thus condition the text generation model. Considering once again the customer support scenario, nowadays, companies rely on having a good customer support service to retain and increase the confidence of customers, with a fast and efficient resolution of possible problems. Due to these expectations, some companies have started to rely on conversational agents. This raises a concern: receiving what seems like an automated and generic reply message might not please most customers. As mentioned in Davidow (2003), taking into account the emotions and answering accordingly is fundamental for the improvement of customer satisfaction. However, available conversational agents for customer support are able to produce answers that are both grammatically correct and useful, but they do not take into account the emotions expressed by the customer (Hu et al., 2018). Thus, an end-to-end system that aids such applications by combining informative answers with the appropriate reply emotion that best suits the customer’s state-of-mind, could help improve user satisfaction by providing a fast, adequate and non-generic answer. For example, in Figure 1.1, we can observe that, while both answers can be considered as correct, an answer that expresses an appropriate emotion may be more satisfying for the user, rather than the generic reply.

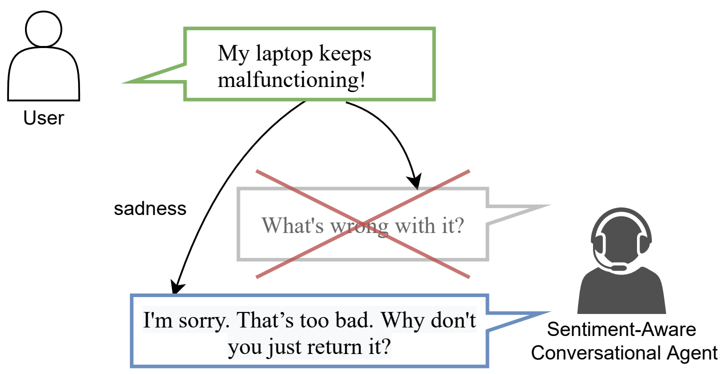

User
My laptop keeps malfunctioning!
sadness
What's wrong with it?
I'm sorry. That’s too bad. Why don't you just return it?
Sentiment-Aware Conversational Agent

Figure 1.1: Example of a conversation when using the sadness emotion is appropriate.

1.2 Objectives

Following Li and Zhang (2018), which highlights that in human-human conversations the emotions expressed in two subsequent utterances from different speakers often change and therefore are important when predicting the "correct emotion for an upcoming response before generation", in this work we propose an end-to-end sentiment-aware conversational agent, which can be observed in Figure 1.2.

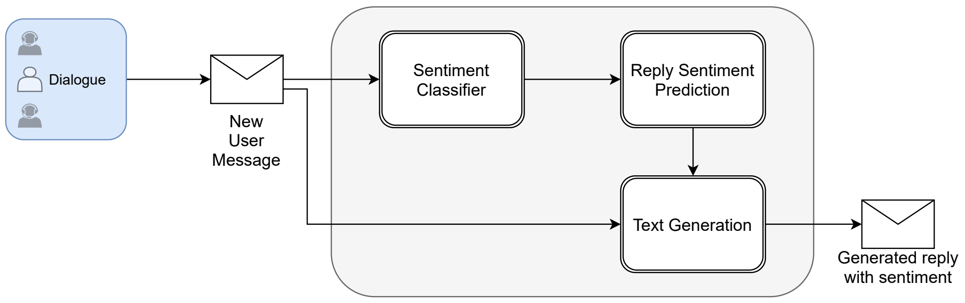


Figure 1.2: Proposed end-to-end sentiment-aware conversational agent.

The goal of the proposed conversational agent is to be able to generate sentiment-appropriate sentences during a dialogue through the use of three models:

- A sentiment classification model, which classifies the user's input sentence;
- A reply sentiment prediction model, which leverages the predictions of the sentiment classification model, along with previous dialogue context, in order to the predict the appropriate reply sentiment that should be expressed by the conversational agent;
- A text generation model which receives the previous dialogue context and is conditioned on the sentiment predicted by the reply sentiment prediction model, in order to generate a sentence that is context-aware and expresses the predicted sentiment.

1.3 Contributions

The main contributions of this work are the following:

1. We explore an end-to-end conversational-agent approach which is able to: leverage information about the sentiment of the received input sentence, through a sentiment classification model; consider the most appropriate reply sentiment to that input, through a reply sentiment prediction model; and generate text conditioned on a given sentiment, through a text generation model;
2. We perform an extensive experimentation cycle for both sentiment classification and reply sentiment prediction models. In particular, we explore the possibility of including multiple context sentences, and the use of retrieval augmentation techniques in order to bias the model towards the correct sentiment label;

3. We adapt the work proposed by Wolf et al. (2019) to the task of sentiment-conditioned text generation. In particular, we show that this adaptation has clear benefits for models fine-tuned with small datasets;

4. We conduct a human evaluation that aims to assess the adequacy and sentiment accuracy of the proposed system. Furthermore, we show that this evaluation correlated well with the used automatic metrics, which motivates their use during the development of new models.

1.4 Document Outline

The overview of the present document is described as follows:

In Chapter 2 we start by introducing some important topics in order to understand this work, in particular, neural networks (Section 2.1), language models (Section 2.2), word embeddings (Section 2.3), text generation (Section 2.4), and text classification (Section 2.5).

In Chapter 3 we describe some of the available corpora for the tasks at hand (Section 3.1). Next, we mention the state-of-the-art models used for the tasks of sentiment classification (Section 3.2), reply sentiment prediction (Section 3.3) and text generation (Section 3.4).

In Chapters 4 and 5 we present the models that are going to be used in order to build the sentiment-aware conversational agent.

Chapter 6 consists on an experimental cycle performed for each model, along with support experiments and examples that help validating the obtained results. After obtaining the automatic metrics' results, we perform a human evaluation, in order to correlate the results obtained with the automatic metrics, as reported in Chapter 7.

We finalize this document with a conclusion of our findings in Chapter 8, along with future considerations for this work.

Chapter 2

Background

In this section we will introduce some key concepts needed for the development of this work. Firstly, we will present some background on neural networks. Then, we will detail language models and word embeddings. Finally, we will introduce the tasks of text generation and text classification.

2.1 Neural Networks

2.1.1 Multi-Layer Perceptron

In order to understand more complex neural networks, we will first start by defining the basic perceptron, which represents a linear model of its input.

The Perceptron, proposed by Rosenblatt (1958) defines a linear transformation of the input **x** by applying the set of weights **w** and then adding the bias **b**. The Perceptron is defined below:

$$y(x; w, b) = w \cdot x + b. \tag{2.1}$$

Since not all problems have linearly separable points, this calls for the necessity of extending the linear model to solve non-linear problems. In order to solve that class of problems the Multi-Layer Perceptron (MLP) was introduced. This model differs from the basic perceptron by making use of non-linear activation functions, which allow the model to tackle non-linear problems. Therefore, the MLP is composed of at least three layers: one input layer, one hidden layer (followed by a non-linear activation function), and one output layer. Some commonly used activation functions are the sigmoid function, the hyperbolic tangent and the rectifier activation function (Glorot et al., 2011). This mechanism allows us to make the model more or less complex by composing multiple hidden layers as described below,

$$y(x; \mathbf{w}_n, \mathbf{b}_n) = \theta_n(...(\theta_2(\theta_1(w_1 \cdot x + b_1)w_2 + b_2)...)w_n + b_n) \tag{2.2}$$

where θ_n represents the different activation functions used in each layer.

In Figure 2.1 we can see an example of a MLP with two hidden layers. This model can also be seen as a network, given how the units in the layers connect to each other, forming an acyclic directed graph.

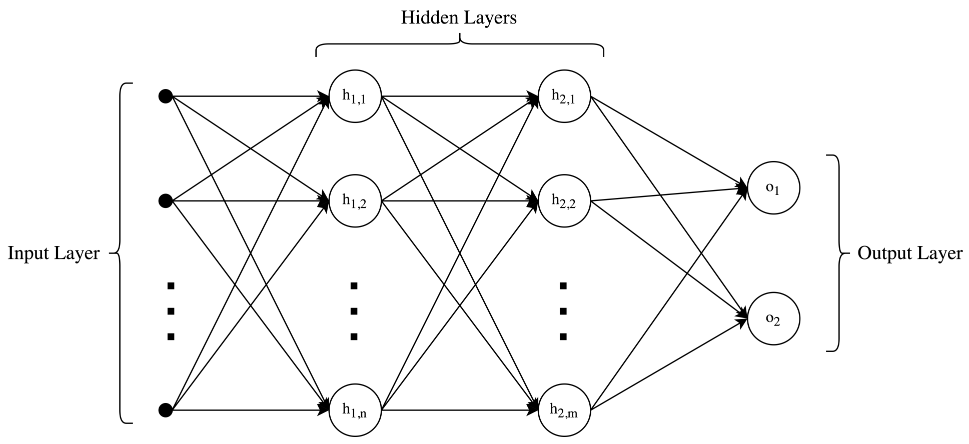


Figure 2.1: Multi-Layer Perceptron with two hidden layers.

A commonly used method to train these kind of networks is the Backpropagation algorithm (Rumelhart et al., 1985). In order to implement this algorithm we first need to define a loss function $L(\hat{y}, y)$, which calculates the error between the predicted output, $\hat{y}$, and the true output, y. The goal of the backpropagation algorithm is to use this error to adjust the parameters of the model, so that the predictions become more accurate during training. The execution of this algorithm is divided in two steps:

- **Forward Pass**: In Figure 2.1 we can observe how each layer propagates its results to the next layer. In the Forward Pass the objective is to pass the input through all the layers in the network and output a prediction. Then, we calculate the error using the chosen loss function.
- **Backward Pass**: In this step, the algorithm propagates backwards the cost from the loss function through the network, in order to compute the gradients of each weight of the model. The update of the gradients is done through an optimization method. Some commonly used methods are the Stochastic Gradient Descent (LeCun et al., 1998) and Adam (Kingma and Ba, 2015).

2.1.2 Recurrent Neural Networks

In this work we will be working with text data, which means we will be dealing with sequences. Usually, in text, the order in which words appear in a sentence is very important to properly classify a sentiment, or to generate cohesive text, for example. Furthermore, text samples tend to have different input sizes, e.g., different sentences can have a different number of words. MLPs, mentioned in Section 2.1.1, are not able to deal with any of the aforementioned particularities of text data. In contrast, Recurrent Neural Networks (RNNs) (Elman, 1990) are a family of neural networks that are able to both deal with sequential data and with examples with a variable number of input sizes. Both these characteristics make this class of neural networks particularly useful for NLP tasks.

In Figure 2.2, we can observe an example of a RNN. We can think about the RNN as an “unrolled” MLP, where the result of each hidden state is used as input to the next state. We can also think about it as repeating the MLP t times, where at each timestep t the calculated information serves as input to the network state at timestep $t + 1$. Furthermore, an input sequence is passed to the model one token at a

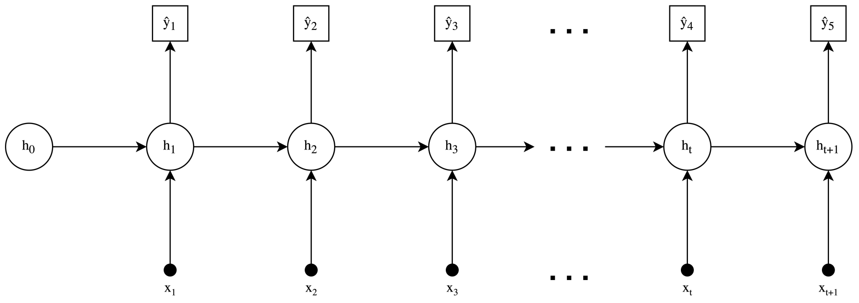


Figure 2.2: Simple Recurrent Neural Network with t+1 hidden states. h_0 is the initial hidden state.

time. For example, if we were in the presence of a sentence as input, x_1 would correspond to the first word, x_2 to the second, and so forth. This aspect guarantees the ability of the model to handle sequential data. The hidden states take into consideration the sequence information, combining at each timestep both new input information, and the previous sequence representation calculated by the previous states. The computation done on each hidden state at timestep t is described as,

$$h_t = \theta_1(w_{hh} \cdot h_{t-1} + w_{hx} \cdot x_t + b_h) \tag{2.3}$$

where θ_1 represents the activation function, h_t the hidden layer, w_{hh} a transformation between consecutive hidden states ($\mathbb{R}^{d_h \times d_h}$), w_{hx} a transformation between the hidden state and the input ($\mathbb{R}^{d_h \times d_x}$), x_t the input, and b_h the bias associated with the hidden state. We can observe how the computation is similar to the one performed in the MLP, but in this case we also take into consideration the value from the previous hidden state. Furthermore, the output of each hidden state at timestep t is given by,

$$\hat{y}_t = \theta_2(w_{yh} \cdot h_t + b_y), \tag{2.4}$$

where $\hat{y}_t$ is the predicted output, θ_2 the activation function, w_{yh} a transformation between the output and the hidden state ($\mathbb{R}^{d_y \times d_h}$), and b_y the bias associated with the output. The training of these kind of networks requires a modification of the backpropagation algorithm, called the Backpropagation Through Time (Werbos, 1990).

As mentioned before, RNNs can process inputs of arbitrary size. Moreover, previous hidden states' information is propagated across the network in order to take into consideration the input sequence. However, Pascanu et al. (2013) mentions that RNNs are prone to the vanishing gradient problem, which is particularly relevant for long sequences where the backpropagation algorithm results in small gradient values which can lead the network to eventually stop training. A more complex architecture that is able to tackle this issue is the Long Short-Term Memory (LSTM) model (Hochreiter and Schmidhuber, 1997). LSTMs have a similar structure to the RNNs, where the hidden states hold information about the previous states, but introduce the following aspects to tackle the vanishing gradient problem:

- **Cell State**: The cell state, C_t, is an additional state vector used by the LSTM to calculate the hidden state, h_t, at each timestep t.

- **Gates**: The purpose of the gates is to change the information in the cell state. There are three different kinds of gates:
  - **Forget Gate Layer**: The forget gate, f_t, selects which portions of the Cell State are going to be *deleted*. It does so by applying the sigmoid function, whose output is 1 or 0, and results in which parts are kept or removed, respectively,

$$f_t = \sigma(w_f \cdot (h_t + x_t) + b_f). \tag{2.5}$$

  - **Input Gate Layer**: The input gate, i_t, updates the cell state and it is divided in three parts. In Equation 2.6 it selects which portions of the cell state are going to be updated. In Equation 2.7 it creates an input vector of new values, $\tilde{C}_t$, which is used to update the selected portions. Finally, in Equation 2.8, both steps are combined in order to update the cell state, C_t.

$$i_t = \sigma(w_i \cdot (h_{t-1} + x_t) + b_i), \tag{2.6}$$

$$\tilde{C}_t = \tanh(w_C \cdot (h_{t-1} + x_t) + b_C), \tag{2.7}$$

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t. \tag{2.8}$$

  - **Output Gate Layer**: The output gate, o_t, decides which parts of the updated cell state we want to output. The value of the hidden layer is finally given by h_t,

$$o_t = \sigma(w_o \cdot (h_{t-1} + x_t) + b_o), \tag{2.9}$$

$$h_t = o_t \cdot \tanh(C_t). \tag{2.10}$$

Having described RNNs as a model that can process sequential data and then LSTMs as a model that helps mitigate the vanishing gradient problem, we can observe how both these models only take into consideration the context of the input sentence from left to right. Bidirectional RNNs (bi-RNN) were proposed by Schuster and Paliwal (1997) as a means to also consider the future context of the input sentence (from right to left). In this model we have two RNNs, one processing the input sentence as described before, and the other processing the same input sentence but backwards. This means that each state has two different representations that are concatenated. This architecture can also be applied to the LSTMs, resulting in an approach called the Bidirectional LSTM (bi-LSTM).

2.1.3 Transformer Encoder

Despite the success achieved by recurrent bidirectional models in various NLP tasks, for instance, in neural machine translation (Cho et al., 2014), language modelling (Peters et al., 2018), or sentiment analysis (Socher et al., 2013), the current model architecture that achieves state-of-the-art results is the Transformer (Vaswani et al., 2017). In this section we will describe the Encoder portion of the Transformer model. An example of one layer of a Transformer encoder can be observed in Figure 2.3.

Each Transformer layer is composed of two sub-layers. The first sub-layer corresponds to the self-attention mechanism which can be interpreted as how the model builds the context of the sequence, as further detailed below. The output of the first sub-layer serves as input to the second sub-layer, a MLP. An important aspect to notice is that the outputs of the self-attention are fed independently to the MLP, which means the computation can be done in parallel.

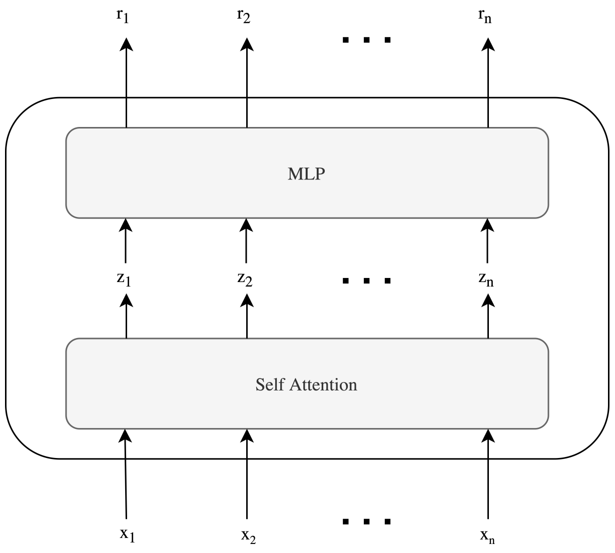


Figure 2.3: Transformer Model Layer (Encoder). x_n represents the inputs, z_n the outputs of the self-attention layer, and r_n the outputs of the layer.

As mentioned before, the self-attention layer is how the model builds the context of the sequence. In another words, for a given token, we are calculating how relevant the other tokens in the sentence are with regard to the one we are evaluating. To do so, the model creates three matrices: Query (Q), Key (K), and Value (V) matrices, by computing the dot product between the input sequence matrix, X, and the three weight matrices learned in the training phase, W^Q, W^K and W^V,

$$Q = X \cdot W^Q \tag{2.11}$$

$$K = X \cdot W^K \tag{2.12}$$

$$V = X \cdot W^V \tag{2.13}$$

where $Q, K, V \in \mathbb{R}^{N \times d_{\text{model}}}$, $X \in \mathbb{R}^{N \times d_{\text{model}}}$ and $W^Q, W^K, W^V \in \mathbb{R}^{d_{\text{model}} \times d_{\text{model}}}$, with N equal to the number of tokens, and d_{model} equal to the size of the model output.

Next, it calculates a vector of attention scores, i.e., how related each token in the sequence is with regard to all other tokens. This computation is described as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{Q \times K^T}{\sqrt{d_{\text{model}}}}\right) \cdot V, \tag{2.14}$$

where first we start by calculating the dot product between the Q and K matrices. Then, in order to help keeping the gradients stable, we divide this result by d_{model}. Finally, the Softmax transformation is applied so that we have a probability score for each token in the sequence, and we multiply the V matrix by the obtained values. This results in matrix $Z \in \mathbb{R}^{N \times d_{\text{model}}}$, which corresponds to a representation of dimension d_{model} for each of the tokens in the sequence.

In order to further improve the attention layer, the authors of the Transformer model introduced a multi-head attention mechanism, described by Equation 2.15. This was done so that the attention score vector for a given token would not exclusively give a high probability score to the token itself, lowering the attention scores given to the other tokens in the sequence. This mechanism involves having multiple sets, i.e. “heads”, of weight matrices and therefore outputting that many attention score matrices, each following the approach previously described. These matrices are then concatenated and multiplied by an extra weight matrix, W^O, yielding the final output of the self-attention layer, Z. W^O guarantees that the output of the layer has the same dimensions of the Z matrix without multi-head attention, in order to use it as input to the MLP layer.

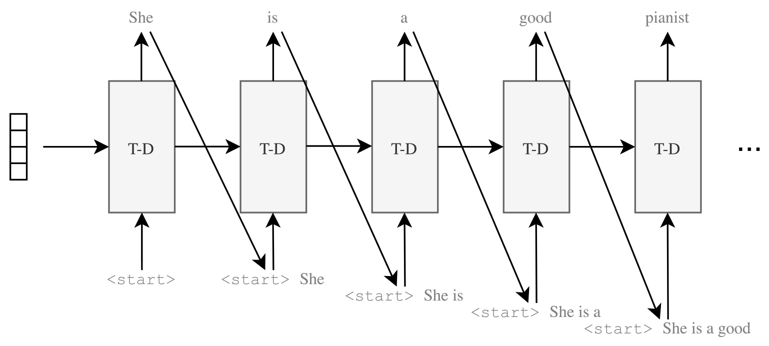
$$\begin{gathered}\mathrm{MultiHead}(Q, K, V) = \text{concat}(\mathrm{head}_1, \mathrm{head}_2, ..., \mathrm{head}_k) \cdot W^O \\ \text{where } \mathrm{head}_i = \mathrm{Attention}(QW_i^Q, KW_i^K, VW_i^V),\end{gathered} \tag{2.15}$$

with $W_i^Q, W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$, $W^O \in \mathbb{R}^{hd_v \times d_{\text{model}}}$, with $d_k = d_v = d_{\text{model}}/h$ and h equal to the number of heads.

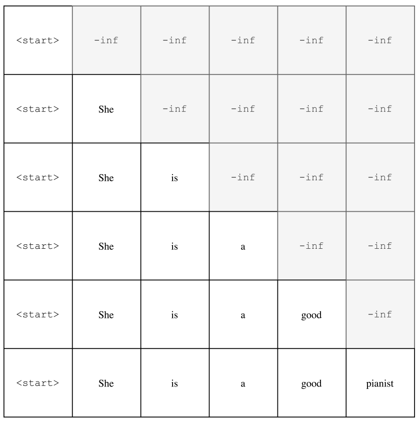
2.1.4 Transformer Decoder

The Transformer Decoder, also introduced in Vaswani et al. (2017), has two key differences when compared to the Transformer encoder. First, instead of a self-attention layer, it has a Masked Self-Attention Layer. In the Transformer encoder subsection we saw how the model receives the full sequence as input and how the self-attention layer builds the full context for each input token. The main idea of the masked self-attention is that when the model is predicting the token x_i, it only receives as input the sequence $(x_1, ..., x_{i-1})$, which requires masking the remainder of the sequence. The second key difference between the Transformers encoder and decoder is an extra layer between the masked self-attention and the MLP, called the Encoder-Decoder Self-Attention Layer. This layer performs the same self-attention previously described for the Transformer encoder, following Equations 2.14 and 2.15, with the only difference being that K and V matrices come from the output of the encoder, while the Q matrix comes from the output of the decoder's masked self-attention layer. This step helps the decoder know which parts of the encoder's input should be focused, and has the same goal as the encoder-decoder attention mechanism used in traditional Sequence-to-Sequence models (Cho et al., 2014; Sutskever et al., 2014; Bahdanau et al., 2016). The encoder-decoder approach uses two neural network models, one as the encoder, and the other as the decoder, taking advantage of the attention mechanism by feeding each decoder step with an extra context vector based on the encoder states. This vector helps guiding the decoder's output by making it aware of the input sequence's contextual information built by the encoder. The final layer is a MLP, similarly to the Transformer encoder.

In Liu et al. (2018) the Transformer decoder was explored as a self-sufficient model, that can be used



(a) Model execution.


<start> -inf -inf -inf -inf -inf
<start> She -inf -inf -inf -inf
<start> She is -inf -inf -inf
<start> She is a -inf -inf
<start> She is a good -inf
<start> She is a good pianist

(b) Model input.

Figure 2.4: Example of the Transformer Decoder (T-D) generating the sentence "*She is a good pianist*".

without an associated Transformer encoder. The main difference is that this model has only two layers, the masked self-attention and the MLP.

An example of the execution of the Transformer Decoder proposed in Liu et al. (2018) can be observed in Figure 2.4a. In the example the model is generating the sentence "*She is a good pianist*". The model generates one token at a time and uses the previously generated tokens as input to the model (in Section 2.2 this idea will be further explored when we introduce autoregressive language models). In Figure 2.4b we can observe what the input matrix looks like. The masked elements of the sequence are set to `-inf`, so that they do not influence the output scores of the self-attention mechanism.

2.2 Language Models

In this Section we will start by describing the task of language modelling, followed by some of the theories behind simpler language models that led to autoregressive and masked language models, the current state-of-the-art for this task.

2.2.1 Classic Language Model

Language Modelling is a task that results in a model that is able to calculate the probability of a sequence of words in a given corpus. A simple approach to this task can be represented by,

$$p(w_1, ..., w_n) = \frac{\text{count}(w_1, ..., w_n)}{N} \tag{2.16}$$

where the probability of a sequence of words $(w_1, ..., w_n)$ is given by the number of times we have seen that sequence over the total number of sequences in a given corpus. The problem with this approach is that a sequence that is not part of the corpus will yield a probability of zero.

2.2.2 N-gram Language Model

In order to support unseen examples, the n-gram language model relies on the Markov Assumption, which states that "the future is independent of the past given the present" (Goldberg, 2017). This means that, in order to assign a probability to a sequence, we can opt to observe only a number words before it,

$$\begin{aligned} p(w_1, ..., w_n) &= \prod_{i=1}^{n} p(w_i|w_1, ..., w_{i-1}) \\ &= p(w_1)p(w_2|w_1)p(w_3|w_1, w_2)... \end{aligned} \tag{2.17}$$

This way, the n-gram model can calculate the probability of a sequence given any number of words. For example, for bigrams we compute the probability of the word w_i given the word w_{i-1}, so we are only using as context the previous word,

$$p(w_i|w_1, ..., w_{i-1}) = p(w_i|w_{i-1}) = \frac{\text{count}(w_{i-1}, w_i)}{\text{count}(w_{i-1})}. \tag{2.18}$$

Since we are not considering the full sentence at once, we are able to give a non-zero probability to sentences that are not present in the corpus. Furthermore, in order to deal with counts of zero for unseen N-grams it is also possible to use smoothing techniques such as the Laplace Smoothing (Lidstone, 1920) which adds one to all counts.

2.2.3 MLP-based Language Model

Even though N-gram language models solve some of the problems of the classic language model, Bengio et al. (2003) highlights two shortcomings of N-gram language models. First, N-gram models can only capture a small context "window" of words. Second, they do not take into account the "similarity" between words. For that matter, Bengio et al. (2003) proposes the MLP-based language model, which uses the multi-layer perceptron neural network (described in Section 2.1.1) in order to calculate a probability distribution over all words in the vocabulary,

$$p(w_i|w_1, ..., w_{i-1}) = \text{MLP}(w_{1:i-1}). \tag{2.19}$$

2.2.4 Autoregressive Language Model

As discussed in Section 2.1.2, MLPs are not able to take into consideration the recurrent structure of text data. By using RNNs or a Transformer decoder we can use the left-wise context of a sentence in order to compute the probability of a word w_i (Kombrink et al., 2011; Radford et al., 2018). Both RNNs and Transformer decoders, at each timestep, calculate a probability distribution over all words in the vocabulary, which is then used to select the output word. These models have the particularity of being autoregressive. This means that the previous output words are used as input when generating subsequent outputs. This way, the model predicts new words based on the previously predicted ones.

2.2.5 Masked Language Model

As mentioned before, autoregressive language models use the left-wise context of a sentence to compute the probability of new words. Nonetheless, it would be desirable to take into account the full context of the sentence. An example where this can be seen is the following: "*The bat **flew** towards the beetle.*". The word "*bat*" can have different meanings. It can be "*bat*" the animal or "*bat*" the sports equipment. Predicting the word "*flew*" with only "*The bat*" as the context can be confusing given the two meanings of the word. If the model had access to the full context of the sentence it would be easier to understand that the sentence is referring to the animal. As mentioned in Devlin et al. (2019), autoregressive language models can only be trained exclusively using left-to-right or right-to-left sequences, given that bidirectional models would allow the word being predicted "to see itself". In order to address this problem, Devlin et al. (2019) makes use of the masked language modelling approach, first proposed in Taylor (1953). This approach enables the training of the language model to be done in a bidirectional manner, by "masking" a percentage of words in the text. The goal of this language model is then to correctly predict those masked words, restoring the original input sentence.

In Section 2.3 we will see how training models with the task of language modelling can be used to generate word embeddings that can be applied to different tasks. This process is referred to as Transfer Learning, i.e., leveraging representations learned in a different, and usually more generic, task than the one we are trying to solve. Moreover, a more direct use of language models is text generation. In Section 2.4 we will see how autoregressive language models can be used to generate text token by token.

2.3 Word Embeddings

In order to process text through the models mentioned in Section 2.1, we need to transform the words into a suitable numeric representation. In this section we will briefly describe the origin and evolution of word embeddings followed by the current state-of-the-art in calculating these word representations.

2.3.1 One-Hot Embeddings

The first basic representation of a word embedding we will describe is the One-Hot Encoding, which generates a sparse representation of a word. This representation is obtained directly from the corpus being used. Each word is expressed by a vector with dimension equal to the length of the vocabulary of the corpus, with each position of the vector representing a word from the vocabulary. All the positions in the vector are 0 except the position correspondent to the word being embedded. This representation also allows to have other features such as the part-of-speech of the word (e.g., whether it is a name, determinant, etc.). One downside of this representation is that it can become computationally expensive when dealing with large vocabularies.

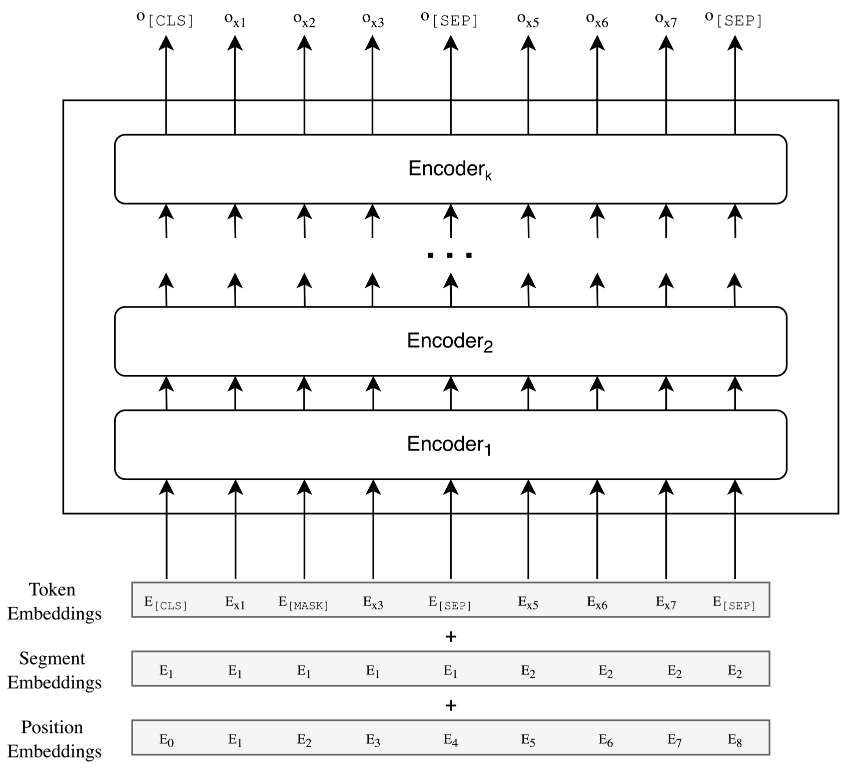
2.3.2 Dense Static Embeddings

Dense Embeddings aim to reduce the dimension of each word's embedding by encoding it into a smaller dimension vector, which is part of an embedding space shared by all words. Furthermore, it also allows the representation of words to capture semantic relationships between them, something that we could not achieve with one-hot embeddings. Some of the first models that introduced Dense Embeddings were Word2Vec (Mikolov et al., 2013) and GloVe (Pennington et al., 2014). These models have the particularity of being static, i.e., they do not take into consideration the context of the word when generating its embedding. This means that each word has only one representation regardless of its context. In order to obtain word embeddings, the Word2Vec model is trained as a MLP-based language model (as described in Section 2.2). It starts by making samples of the corpus, using either the Continuous Bag of Words (CBOW) architecture, or the Skipgram architecture. In the CBOW approach we sample a "window" of text at a time, obtaining for a window of three words for example, the first and the last words as input and the second word as target. In another words, the model should to be able to predict the target word given its two surrounding words. In the skipgram architecture, using the same window of three words as example, the input of the model is the second word, and the targets are the two surrounding words. In this task, the goal of the model is to predict the surrounding words. After training, this model allows us to obtain an embedding matrix, i.e., a unique fixed word embedding representation for each word in the vocabulary. The GloVe model obtains a similar embedding matrix by first creating a matrix of occurrences of each word given a context. Then it factorizes it into two matrices, being one of them the embedding matrix. As in the Word2Vec approach, the embedding matrix of GloVe can also be used as a unique fixed word embedding representation for each word in the vocabulary.

By analyzing the obtained word embeddings matrices for either Word2Vec or GloVe, we can observe that semantically similar words will be close to each other in the embedding space.

2.3.3 Dense Contextual Embeddings

As previously mentioned, dense static embeddings allow each word to have only one representation regardless of its context. In order to tackle this limitation, Peters et al. (2018) introduced the Embeddings from Language Models (ELMo) approach. ELMo is based on the bi-LSTM architecture, and akin to Word2Vec and GloVe, is also trained on the task of language modelling. As mentioned in Section 2.1.2, the bi-LSTM architecture is composed of two LSTMs, one receiving the original input sequence, and the other the same input sequence but backwards. This guarantees that the model takes into consideration the context of a word when generating its embedding. Furthermore, this approach is able to leverage the full context of a word within a sentence and not only a window surrounding it. With the introduction of the Transformer model, Devlin et al. (2019) introduced the Bidirectional Encoder Representations from Transformers (BERT). This model is also trained on the task of language modelling, more specifically masked language modelling, described in Section 2.2.5, and next sentence prediction. In the masked language modelling task, the model has to predict the masked words in a sentence. To do so, it masks 15% of the words in the input given to the model, by replacing the words with the token `[MASK]`. In the next sentence prediction task, the model has to predict whether two sentences appear consecutively in the text. In order to do so, it starts by sampling sequence pairs, balancing them so that half of the samples appear consecutively in the text and the other half do not. BERT is built using a Transformer encoder stack (described in Section 2.1.3). The model's architecture can be observed in Figure 2.5.



o[CLS] ox1 ox2 ox3 o[SEP] ox5 ox6 ox7 o[SEP]
Encoderk
...
Encoder2
Encoder1
Token Embeddings
E[CLS] Ex1 E[MASK] Ex3 E[SEP] Ex5 Ex6 Ex7 E[SEP]
+
Segment Embeddings
E1 E1 E1 E1 E1 E2 E2 E2 E2
+
Position Embeddings
E0 E1 E2 E3 E4 E5 E6 E7 E8

Figure 2.5: BERT architecture.

The input given to the Transformer is the sum of three different embeddings: token, segment and position embeddings. First, the token embeddings correspond to the input sequence. The vector starts with a `[CLS]` token, and if there is more than one sentence in the input, they are separated with a `[SEP]` token. The segment embedding contains information on which sentence each token belongs to. Finally, the position embedding, which is learned when the model is trained, represents the position of each token in the sequence and the distances to the other tokens. This last embedding guarantees that the model takes into account the order of the sequence. As observed in Devlin et al. (2019), the model can be used for different tasks such as classification, question answering, or sentence tagging. Other works have explored the Transformer architecture in order to obtain word embeddings and further improved upon BERT scores, such as RoBERTa (Liu et al., 2019), which differs from BERT by training with more data for longer, different hyperparameter values, and only for the task of masked language modelling, among others. It is important to notice that the word embeddings obtained from either ELMo or BERT can be computed as any combination of the layers in the model, for example, using only the last layer or the concatenation/average/sum of any number of layers. A possible drawback of contextual embedding models is that since they rely on the sentence for context for each word it is necessary to run the full model for each new input, which is computationally expensive, while for static embedding models each word has only one embedding regardless of its context.

Given that the goal of word embedding models is to have a good representation of a vocabulary and the relationships among words in it, it is important to pre-train these models using large volumes of data. This allows the model to generate embeddings with a better understanding of semantics that would not be possible with smaller corpora.

2.4 Text Generation

In Section 2.2 we explored language models that predict the probability of a sequence in a given corpus. As mentioned in Goldberg (2017), any language model can also be used to generate random sentences using the following process:

1. Begin with a `<start>` symbol;
2. Calculate the probability of all words in the vocabulary conditioned on the `<start>` symbol, $p(w_1|\texttt{<start>})$, yielding a probability distribution;
3. Choose a word from the probability distribution using the chosen decoding method, such as the greedy search or the beam search approaches;
4. Calculate the probability distribution conditioned on the chosen word, $p(w_2, \texttt{<start>}, w_1)$;
5. Repeat steps 3 and 4 until the model outputs the end of sentence symbol.

In this section we will mainly focus on autoregressive language models, which make use of architectures such as the RNN or the Transformer in order to tackle the task of computing the next token in a sequence. In particular, in Section 2.2.4, we mentioned how the tokens that are being generated by the model are used as input to generate the next tokens, making the models autoregressive.

2.4.1 Training

An important detail with regard to the training of text generative models is a technique called teacher forcing (Williams and Zipser, 1989). In simple terms, during training the new tokens are always generated conditioned on the gold sequence, i.e., the correct sequence present in the corpus. This means that when generating a new word, even if the model makes a wrong prediction, the next word will be generated using the expected one. The goal of this method is to teach the model to predict the correct words given the best possible context.

2.4.2 Decoding

Throughout this work we have described how language models compute a probability distribution for all the words in the vocabulary at each timestep. What is is left to explain is how to select a word from the probability distribution. The simplest decoding approach is the Greedy Search, which simply chooses the most probable word according to the calculated probability distribution. The problem with this method is that by assuming a greedy strategy at each timestep, the final sequence can end up not being optimal because we are not searching subsequent paths. An alternative method to deal with this lack of optimality is the Beam Search, which is a more generic version of the greedy search method. In order to implement this method we start by defining a parameter b, which states how many paths the algorithm keeps at each timestep. When all paths reach the end of sentence token, the algorithm selects as the output sequence the path with the highest average token probability, $\frac{1}{N}\sum_{i=0}^{N} p(i)$, where N is the sequence length. Moreover, if we set $b = 1$, at each timestep the algorithm keeps only one path, and thus beam search behaves as the greedy search.

2.4.3 Transformer-based Generators: GPT

In Section 2.1.4 we described the Transformer decoder introduced by Liu et al. (2018), which can act as a standalone Transformer decoder and is trained as an autoregressive language model (Section 2.2.4). The decoder outputs one token at a time, using the previously generated tokens as input to generate subsequent ones, which makes it ideal for the task of text generation. The Generative Pre-trained Transformer (GPT) model (Radford et al., 2018) is a stack of Transformer decoders which is trained in two stages. The first stage corresponds to an unsupervised pre-training of a language model. In the second stage the model is fine-tuned in several datasets of four supervised tasks: classification, textual entailment, question-answering and natural language inference. GPT-2 (Radford et al., 2019) is a model built based upon GPT with the key differences being that it is a larger model trained on a larger dataset and that it is no longer fine-tuned on additional tasks. Furthermore, the authors of the paper highlight the “zero-shot” capabilities of the model. GPT-3 (Brown et al., 2020) further improves on GPT-2, using an even larger number of parameters and being trained on a larger dataset. The idea behind GPT-2 and GPT-3, and why they outperform GPT is that they are larger models trained on larger datasets, and therefore are able to model different tasks without the need of additional supervised training.

2.4.4 Perplexity

In order to evaluate the quality of text generation models, one of the commonly used metrics is the Perplexity, which is a metric used to compare language models. It works by evaluating the quality the model’s predicted samples, and is described by the following equation,

$$\text{Perplexity} = 2^{-\frac{1}{n}\sum_{i=1}^{n}\log_2 \text{LM}(w_i|w_{1:i-1})}. \tag{2.20}$$

The intuition behind this metric is that, when comparing two models, the one with the lowest perplexity has the highest probability of correctly generating an unseen example from a test set.

2.5 Text Classification

Text classification is a supervised machine learning task, which means that it relies on a model that is first trained using annotated data. Given an unseen example, the model is able to classify it based on what it learned during the training step. Classification can be binary (e.g., positive or negative sentiment) or multi-labelled. The goal of this is to model output a probability for each of the labels.

2.5.1 Sentiment Classification

Sentiment classification is the task of classifying text with regard to its sentiment. The majority of sentiment classification appropriate corpora are commonly classified with three sentiments: positive, negative and neutral. With the necessity of developing more detailed classification models, works such as Hsu et al. (2018) and Demszky et al. (2020) have explored larger taxonomies. The selected emotions usually follow psychological theories about the fundamental sentiments expressed by humans. In particular, Plutchik (1980) proposed a wheel of eight primary emotions (*joy*, *trust*, *fear*, *surprise*, *sadness*, *disgust*, *anger*, and *anticipation*), which when combined can express other emotions. Ekman (1992) proposed the six basic emotions (*joy*, *anger*, *fear*, *sadness*, *disgust*, and *surprise*).

According to Zhang et al. (2018) we can consider three different levels of granularity in sentiment analysis: Document, Sentence and Aspect Level Sentiment Analysis. In this work we will be focusing on sentiment analysis at a sentence level, which is adequate in dialogues, where users tend to express themselves in single sentences.

2.5.2 Evaluation Metrics

In order to evaluate the quality of text classification models after training, it is important to analyse how the model classifies unseen examples. To do so, it is useful to visualise a confusion matrix (Table 2.1). The True Positives (TP) are the examples the model classified correctly, the False Postives (FP) are the examples the model classified as positive but were in fact negative, the False Negatives (FN) are the examples the model classified as Negative but were in fact Positive, and finally, the True Negatives (TN), are the examples the model classified as Negative correctly.

		Predicted Class	
		+	**-**
Actual	**+**	TP	FN
Class	**-**	FP	TN

Table 2.1: Confusion Matrix.

Some of the evaluation metrics that require these values are the following:

- **Accuracy**: measures the percentage of examples the model classified correctly,

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{2.21}$$

- **Precision**: measures the percentage of examples the model classified correctly out of the examples that it classified as positive,

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2.22}$$

- **Recall**: also referenced as True Positive Rate, measures the percentage of examples the model classified correctly out of the total number of examples that are positive,

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2.23}$$

- **F1**: harmonic average of the Precision and the Recall. This value varies between 0 and 1. A low value of F1 is indicative of poor Precision and Recall values, while a high value indicates that the Precision and the Recall are both achieving good values,

$$\text{F1} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.24}$$

The aforementioned metrics can only be applied to binary classification problems. In order to calculate the accuracy for a multi-label classification problem the average accuracy is used. Regarding the precision, recall and F1 metrics there are two solutions: macro and micro level metrics.

- **Macro Level Metrics**: calculates an average of the metric being evaluated. For example, the sum of the precision for all labels is calculated, and then divided by the number of labels. In this case, all classes weigh the same for the calculation of the metrics.

- **Micro Level Metrics**: calculates the sum of the TP, TN, FP and FN for all classes. These values are used to calculate the metrics using the formulas mentioned previously. It is important to notice that the micro level metrics give different weights to the classes, depending on the number of examples of each class. This means that classes with a larger number of examples will have a stronger weight when using this metric.

Chapter 3

Related Work

In this chapter we will start by describing the available customer support and sentiment corpora. Furthermore, we will describe the current most relevant state-of-the-art regarding sentiment classification, reply sentiment prediction, and dialogue and conditioned text generation.

3.1 Available Corpora

As described in Section 1.1, the goal of this work is to develop an end-to-end sentiment-aware conversational agent. In particular, we want to overcome the difficulties of using small datasets for fine-tuning state-of-the-art Transformer models (Vaswani et al., 2017), which typically require large amounts of data and capacity to be able to generate fluent and adequate text (Wolf et al., 2019). Therefore, we will focus on small datasets in order to validate our hypothesis, considering dialogue corpora that is labelled on every dialogue turn.

In Section 1.1 we also introduced as an example the domain of customer support, as it would be an application that would greatly benefit from such systems, given the importance of automatic conversational agents in order to fulfil the growing customers' requests in a timely manner. The problem is that the generated messages by these systems is often generic and does not consider the user's emotion (Hu et al., 2018). As mentioned in Davidow (2003), taking into account the emotions of the customer and answering accordingly, is fundamental for the improvement of customer satisfaction.

Considering the aforementioned goals, throughout this section we will describe some of the available customer support, dialogue, and sentiment-labelled corpora.

Customer Support on Twitter

The "Customer Support on Twitter" dataset[1] is a corpus of over three million tweets with interactions between customer support accounts on Twitter of twenty companies, such as Apple and Amazon, and their customers. This aspect makes it ideal for tasks like dialogue text generation since we can simulate

[1] `https://www.kaggle.com/thoughtvector/customer-support-on-twitter`

multiple interactions between an agent (the customer support account) and a user. However, this dataset is not labelled with sentiment.

Twitter US Airline Sentiment

The “Twitter US Airline Sentiment” dataset[2] is a corpus containing tweets of users who mention one of six different US airlines. The tweets are labelled with either the positive, negative or neutral label, and the annotators were also required to categorize the reason for labelling them that way. This corpus is useful for analysing customer satisfaction, but since it does not contain interactions, it can not be used for dialogue text generation.

EmotionLines

EmotionLines (Hsu et al., 2018)[3] is an emotion dialogue dataset labelled on each utterance. This dataset has two different corpus: “Friends TV Scripts”, where 1000 different scenes are treated as dialogues; and “EmotionPush Chat Logs”, which is composed of 1000 private conversations from Facebook Messenger. Both corpora use as labels the six Ekman’s basic emotions (Ekman, 1992), and *neutral*.

DailyDialog

The DailyDialog corpus (Li et al., 2017)[4] was built from websites that are used to practice English in a daily life scenario. It contains 13118 multi-turn dialogues, divided in 10 different themes such as: Finance, Politics, Health, Work, etc. It is labelled with the six Ekman’s basic emotions, and *neutral*.

GoEmotions

The GoEmotions corpus (Demszky et al., 2020) is a compilation of approximately 58000 English comments collected from popular Reddit[5] subforms and manually annotated with one of 27 different sentiment categories or neutral. Given the large number of emotions in its taxonomy, it is a good fit for a more fine-grained analysis of customer feedback.

ScenarioSA

The ScenarioSA (Zhang et al., 2020) is a corpus of 2214 English conversations (with over 24000 utterances) over various domains, retrieved from various websites that provide online communication services. Each utterance is labelled according to its polarity (*positive*, *negative* or *neutral*). Moreover, the final emotion of each participant in the dialogue is also labelled. This dataset is ideal for the task of interactive sentiment analysis.

[2]`https://www.kaggle.com/crowdflower/twitter-airline-sentiment`
[3]`http://doraemon.iis.sinica.edu.tw/emotionlines/index.html`
[4]`http://yanran.li/dailydialog.html`
[5]`https://www.reddit.com`

EmpatheticDialogues

The EmpatheticDialogues corpus (Rashkin et al., 2019) contains 25000 dialogues, which were retrieved from a crowdsourcing platform by letting participants communicate while constrained to one of 32 emotions. In this corpus the speakers describe one or more events associated with one of the 32 emotions, and the listeners respond according to 8 empathetic intents (*questioning*, *agreeing*, *acknowledging*, *sympathizing*, *encouraging*, *consoling*, *suggesting*, and *wishing*), and *neutral*.

Twitter Emotional Customer Care

Herzig et al. (2016) produced a corpus of 2413 conversations retrieved from two customer support Twitter accounts. Each customer sentence is labelled with one of eight emotions (*confusion*, *frustration*, *anger*, *sadness*, *happiness*, *hopefulness*, *disappointment*, *gratitude*, and *politeness*), while each agent's sentence is labelled with one four emotions (*empathy*, *gratitude*, *apology*, and *cheerfulness*). These labels are a result of a research survey performed by Gelbrich (2010).

Dataset	Sentiment Label	Dialogue	Customer Support	Publicly Available
Customer Support on Twitter	×	√	√	√
Twitter US Airline Sentiment	√	×	√	√
EmotionLines	√	√	×	√
DailyDialog	√	√	×	√
GoEmotions	√	×	×	√
ScenarioSA	√	√	×	√
EmpatheticDialogues	√	√	×	√
Twitter Emotional Customer Care	√	√	√	×

Table 3.1: Available customer support, dialogue, and sentiment-labelled corpora.

In Table 3.1 we can observe a summary of the previously introduced corpora. The EmotionLines, DailyDialog, and ScenarioSA datasets satisfy the requirements we have for our work: being a dialogue corpus labelled with sentiment at sentence-level. These kinds of datasets will enable us not only to train generative models, but also to make predictions about the appropriate reply sentiment in an ongoing conversation and to train sentiment classifier models. Additionally, the "Twitter Emotional Customer Care" corpus also fulfilled the requirements, and would allow us to test our proposed model in a customer support setting. Unfortunately, we were not given access to the data by the authors due to privacy constraints. Nonetheless, the approaches we will be following in this work can easily be applied to a dataset in a specific scenario, such as customer support, that fulfills the previously mentioned requirements.

3.2 Sentiment Classification

The current state-of-the-art for text classification tasks, and particularly sentiment classification, is making use of dense contextual word embedding models, such as BERT (Devlin et al., 2019) or

RoBERTa (Liu et al., 2019), via transfer learning, i.e., using the embeddings obtained from a pre-trained model to train a classification model. The motivation for using pre-trained models is that the embeddings learned by larger models and datasets can encode meaningful features for the task at hand that we would not be able to obtain by learning the embeddings from scratch.

As described in Devlin et al. (2019) and Sun et al. (2020), fine-tuning such models for the task of sentiment classification involves using the final hidden state of the embedding model corresponding to the first input token (the `[CLS]` token, as described in Section 2.3.3) as the sentence representation, $o_{\texttt{[CLS]}}$. Then, an extra linear layer with the softmax transformation is used to predict the probability of the sentence belonging to label l,

$$p(l|o_{\texttt{[CLS]}}) = \mathrm{Softmax}(W \cdot o_{\texttt{[CLS]}}), \tag{3.1}$$

where W represents the learned weights by the classification model. Peters et al. (2019) further separates transfer learning into two different approaches: feature extraction and fine-tuning. Feature extraction is done by "freezing" all the weights learned by the pre-trained model. This means that the pre-trained embedding model will not be further trained and only the weights corresponding to the added classification model require training. On the other hand, fine-tuning involves "unfreezing" some of the learned weights, i.e., re-training some, or all, of the layers of the pre-trained embedding model. Despite being more computationally expensive, it expands the capabilities of the pre-trained model by allowing it to better adapt to the target dataset.

Other works on sentiment classification made use of contextual, speakers, speech acts and topics information (Kim et al., 2020), further pre-trained BERT with a dataset similar to the target sentiment-labelled dataset (Huang et al., 2019), or made use of acyclic graph networks as a way to "model the information flow between long-distance conversation background and nearby context" (Shen et al., 2021).

3.3 Sentiment Prediction

One critical aspect of systems that deal with sentiment-aware text generation in a dialogue context is how to define the appropriate sentiment for the upcoming reply. A possible way to do so is to automatically learn the sentiment that should be expressed through the use of sentiment prediction models. Herzig et al. (2016) makes use of a support vector machine model (Cortes and Vapnik, 1995) in order to predict the sentiment of an upcoming agent sentence in a customer support scenario. In particular, dialogue and textual features, such as the time between interactions, or the emotion of previous sentences, are used as additional information to aid the model. In Bothe et al. (2017) an LSTM is used to classify the sentiment of the next utterance. To do so, the network is trained on a dataset where each example corresponds to two consecutive utterances from two different speakers (in a dialogue style) and the target is the sentiment of the sentence that follows. Li and Zhang (2018) explores a similar approach, using a bi-LSTM as the classification model, but instead of using a pair of utterances, it uses multiple examples of possible answers to the first utterance in order to capture the next reply emotion.

Wen et al. (2021) aims to simulate the emotion transition of humans in a dialogue. To do so, it makes use of the Valence-Arousal-Dominance emotion space (Mehrabian, 1996), which encodes the emotion of words in a 3-dimensional vector space, to calculate the "emotion transition as the variation between the preceding emotion and the response emotion". It is interesting to observe that the reported scores for the aforementioned approaches fall short of expectations, which shows how difficult it is to predict the correct sentiment of the next sentence with current approaches.

3.4 Text Generation

In this section we will start by presenting works that leverage two different alternatives of conditioning text with a given attribute: controllable text generation (Section 3.4.1) and dialogue text generation (Section 3.4.2). The key difference between the two is that controllable text generation involves pre-defining an attribute that is used as an extra input to condition the generated text, while dialogue text generation consists of an end-to-end system that is able to implicitly learn from the data how to generate the desired properties. In particular, since controllable text generation models depend on the conditioning attribute to defined explicitly, they require a mechanism to predict the attribute of the upcoming sentence in order to be used as an end-to-end dialogue system. The sentiment-aware conversational agent proposed in Section 1.1 tackles this issue by introducing a reply sentiment prediction model as part of the system's pipeline, thus, explicitly conditioning the text generation model automatically. We will explore existing approaches to solving this problem in Section 3.4.3.

3.4.1 Controllable Text Generation

In Zhou et al. (2018), "the problem of generating emotional responses in open-domain conversational systems" is addressed through the use of emotion embeddings based on an external vocabulary memory, which are used to condition a sequence-to-sequence (Seq2Seq) (Sutskever et al., 2014) model. Hu et al. (2018) proposed a "Tone-aware Chatbot for Customer Care on Social Media" based on a Seq2Seq architecture with LSTMs. It concatenates an extra tone vector to the input word vector in each step of the decoder with information related to three tones: empathetic, passionate and neutral. These tones are the ones the authors found to be the most significant for customer support satisfaction and therefore are used when responding to customers. Zhou and Wang (2018) builds a collection of dialogues from Twitter that include emojis and assumes the emojis as the underlying emotion in the sentence. Then, it trains a Seq2Seq model on this corpus, and conditions the decoding with an embedding corresponding to the target emoji. Colombo et al. (2019) improves Zhou et al. (2018) by, besides using emotion embeddings to condition a Seq2Seq model, also penalizing neutral words and forcing the model to generate words related with the desired emotion. Santhanam and Shaikh (2019) proposes a similar idea, also using an emotion embedding to condition the language model with a target sentiment, but using a Transformer model (Vaswani et al., 2017), in particular, the GPT-2 language model (Radford et al., 2019). Also using the same GPT-2 model, Dathathri et al. (2019) proposes the "Plug and Play Language Model" (PPLM) for controllable language generation, in which a pre-trained language model is combined with attribute models, allowing for conditional text generation without fine-tuning the language model any further. In order to control the generation, PPLM uses attribute models that compute $p(a|x)$, where x is the generated sequence and a are the attributes we want to model, e.g., a topic or a sentiment. The attribute models can be Bag-of-Words (BoW) or simple classifiers. The way the attribute models work is, at each timestep it performs a forward pass in order to predict $p(a|x)$, i.e., how close is the generated sentence thus far to the desired attributes. Next, it performs a backward pass and uses the obtained gradients in order to update the probability distribution of the generative model, so that the generation of the new token is conditioned to be closer to the desired attributes. Other works on controllable text generation, such as Keskar et al. (2019) propose a conditional language model which is trained from scratch with several control codes to guide generated text. The control codes, which can be topics, URLs, etc., allow a user to control the generated text by choosing a code that can specify content or style of text, for example.

3.4.2 Dialogue Text Generation

The state-of-the-art literature in dialogue text generation mainly consists in data-driven end-to-end models which are capable of generating fluent, appropriate, and meaningful responses in a dialogue setting, by using previous context sentences as input to text generation models. One of the first successful approaches was proposed in Vinyals and Le (2015), which leverages the Seq2Seq architecture and recurrent neural networks to predict an upcoming sentence by using as input to the model the previous context of the conversation, thus, allowing the model to be used in a dialogue scenario. Wolf et al. (2019), applies this idea to the Transformer architecture, and fine-tunes the GPT-2 model using two additional special tokens that are used to separate the sentences belonging to different speakers in the model's input. Regarding implicit text generation with sentiment, Rashkin et al. (2019) fine-tunes a GPT-2 model with the EmpatheticDialogues dataset, proposed in the same work, and concludes that a large-scale empathetic corpus enables the models to express appropriate emotions in dialogue.

3.4.3 Controllable Dialogue Text Generation

Controllable dialogue text generation approaches make use of a mechanism that is able to condition the text generation model automatically, thus allowing the system to work as an end-to-end model. In this section we will consider works that have done this in two ways: using a pre-defined set of rules or heuristics; and through the use of data-driven language models.

Asghar et al. (2018) proposed a conversational model which embeds words using the Valence-Arousal-Dominance (VAD) emotion space (Mehrabian, 1996), and explores decoding by using different Beam Search techniques that aim to incorporate affective diversity in candidate outputs. Furthermore, it designs "training loss functions to explicitly train an affect-aware Seq2Seq conversation model", following three heuristics: minimizing affective dissonance (the generated text emotion should be similar to the input's emotion); maximizing affective dissonance (the generated text emotion should not be aligned with the input's emotion); and maximizing affective content (the generated text emotion should avoid being neutral). Huang et al. (2018) adopts an "emotion mining from text" classifier, developed in Yadollahi et al. (2017), to classify the emotions expressed in previous conversation context, and uses this information, together with pre-defined mapping rules defined by the authors, to decide which emotion should be expressed in the reply. This emotion is then either concatenated to the input of the model, or injected into the decoder. Li and Sun (2018) uses a neural network to predict which emotion keyword, selected from a pre-defined lexicon dictionary (Xu et al., 2008), should be used in the response, which, similarly to Huang et al. (2018), is then introduced in the decoder. In Zhong et al. (2019a), the VAD emotion space is used to understand the emotion expressed in previous context. The model's response is then conditioned by following a similar or opposite emotion to the speaker's assessed emotion.

Xie and Pu (2021) argues that approaches that rely on a pre-defined set of rules or heuristics, such as the previously mentioned, are not supported by psychology literature, and therefore emotional interactions in human-human conversations should be explored instead with a large-scale emotional corpus by using data-driven language models. In order to acheive that, some works leverage a multi-task approach that jointly trains an emotion encoder and the text generation model, which is conditioned on the emotional state assessed by the emotion encoder (Lubis et al., 2018; Rashkin et al., 2018). On the other hand, Xie and Pu (2021) incorporates a emotion/intent predictor, which is separately trained from the text generation model, with the goal of deciding the emotion/intent for the reply to be generated. That emotion is predicted based on previous context, and is then encoded and fed to the text generation model. An interesting aspect to notice about the aforementioned works is that the evaluation done focuses on whether the generated texts are empathetic, and not whether they are generating a specific emotion. During our work we will also be evaluating whether the developed models are capable of generating specific sentiments.

Other existing works focus on dialogue text generation but conditioned on different properties, such as a persona. One such example is Wolf et al. (2019), which is based on the GPT-2 model and fine-tuned for the dialogue task. The agent generates new sentences making use of a pre-defined knowledge base, which includes sentences about a persona, and a dialogue history with sentences from both the agent and the user. This is achieved by passing as input to the GPT-2 model a concatenation of multiple sentences of the persona, and a portion of the conversation history.

Chapter 4

Sentiment Classification and Reply Sentiment Prediction

The goal of this thesis is to create a conversational system that is sentiment aware. In order to achieve that, it is not only important to understand the sentiment behind each sentence in the dialogue, but also the appropriate reply sentiment in the context of the conversation. To do so, we propose to use two models: a sentiment classification model, responsible for classifying the sentences with a sentiment; and a reply sentiment prediction model, which is able to predict the appropriate reply sentiment given the conversation's context.

The importance of the sentiment classification model is twofold, as it will become clear in Chapter 5: first, it is used to classify messages sent by the customer, as we will see in Section 4.4; secondly, it is a key automatic metric to evaluate the performance of the text generation model with regard to the expressed sentiment, as it is our goal to condition the generated text on a desired sentiment. On the other hand, the reply sentiment prediction model is key to guide the conversational agent towards an appropriate sentiment, as its predictions are used by the generation model to condition the answers delivered to the user of the conversational system.

In this chapter, we will start by describing the base model we are going to use. In particular, we are going to make use of pre-trained Transformer models, such as BERT (Devlin et al., 2019), which have been shown to provide strong baselines for the task of sentiment classification. Then, we will describe the improvements made to the baseline, such as, using previous dialogue context and the sentiment labels of similar examples retrieved from the train corpus as input to the model. Finally, we will describe the reply sentiment prediction classifier and how, given a dialogue context, it can be used to predict the appropriate sentiment that should be conveyed by the reply of the conversational system.

4.1 Base Model Description

As mentioned in Section 3.2, the state-of-the-art for sentiment classification models are dense contextual word embedding models (Section 2.3.3) based on a Transformer Encoder (Section 2.1.3), such as BERT (Devlin et al., 2019) or RoBERTa (Liu et al., 2019), with a classification layer on top. In this section we will specify how these models can be used to classify a sentence: from obtaining the embeddings output by the model; to using those embeddings to create a suitable representation of the sentence; and, finally, using this representation as input to the classification layer.

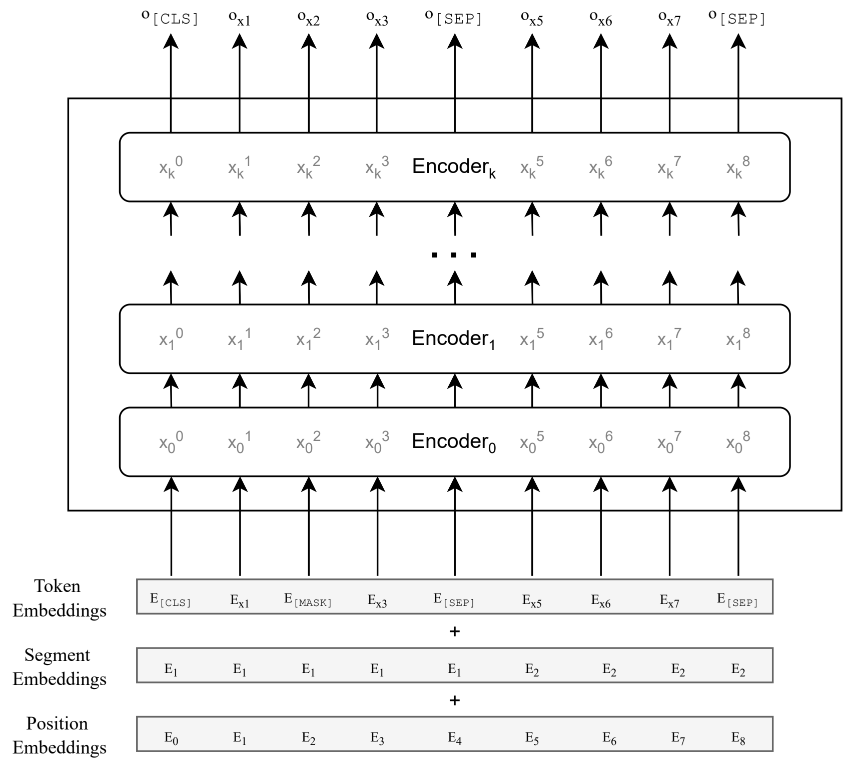


Figure 4.1: BERT architecture.

In Figure 4.1, we can observe the basic architecture of the BERT model, which we will use as an example throughout this section. In particular, each token has a given representation after each layer, represented in Figure 4.1 as x_k^i, where k corresponds to a layer and i to a token position within layer k. These representations are updated throughout the training of the model. As we have seen in Section 3.2, the standard sentence representation is given by the embedding of the `[CLS]` token from the last hidden layer. Nonetheless, as described in Devlin et al. (2019), we can use multiple combinations of the embeddings of the model, for example, averaging all the embeddings obtained in the last hidden layer, or even making use of token representations of intermediate layers.

After choosing a suitable sentence representation, we need to input it to a classification layer in order to obtain the prediction of the model. The usual setup is to use a linear layer with the softmax transformation to predict the probability of a sentence being classified with the label l. This can be described by Equation 4.1, where W represents the learned weights by the classification model.

$$p(l|o_{\texttt{[CLS]}}) = \text{Softmax}(W \cdot o_{\texttt{[CLS]}}), \tag{4.1}$$

This classification layer receives the aforementioned sentence embedding, and outputs a probability distribution over the possible labels. Choosing the label that represents the sentiment of the sentence is then a matter of finding the label with the highest probability. For example, in Figure 4.2, we can observe that the model is giving 34% probability of the sentence being of the red label, and 66% of the sentence being classified as the green label, which means the sentence will be classified with the green label.

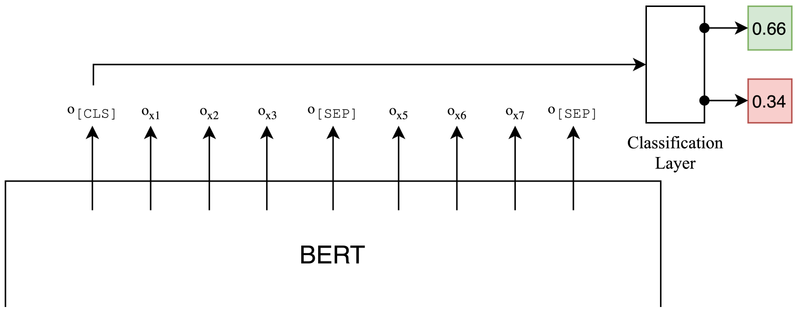


Figure 4.2: Example of a classification layer with the softmax transformation.

4.2 Contextual Sentiment Classification

Given that one of the goals of this work is to develop a dialogue conversational system, we have explored the possibility of using the previous context of the dialogue to further improve the performance of our classification model.

In order to add context to the input of the model, we will take advantage of a particularity of the BERT architecture, the `[SEP]` token, described in Section 2.3.3. Since both BERT and RoBERTa models are limited to 512 input tokens, similarly to Huang et al. (2019), we consider that the first sentence after the `[CLS]` token is the sentence we are trying to classify, and it is followed by its context. This way, if the input surpasses 512 tokens none of the tokens belonging to the sentence will be cut (provided that the sentence on its own does not exceed the maximum number of input tokens). Thus, given a dialogue $D = (s_1, s_2, ..., s_n)$, with n equal to the number of sentences in the dialogue, in order to classify the sentence s_i with x sentences as context, the input to the model is, $\texttt{concat}(s_i, s_{i-1}, ..., s_{i-x})$.

Sentence	**Label**	**Representation**
Does it cost anything?	NEU	`[CLS]Does it cost anything?[SEP]`
Yeah 20$ per month.	NEU	`[CLS]Yeah 20$ per month.[SEP]Does it cost anything?[SEP]`
Ohh!	SUR	`[CLS]ohh![SEP]Yeah 20$ per month.[SEP]`

Table 4.1: Example of a dialogue from the EmotionPush dataset and the sentence representation with one sentence as context ($x = 1$). NEU corresponds to the label *neutral*, and SUR to the label *surprise*.

In Table 4.1 we can observe an example of a dialogue from the EmotionPush dataset. On the leftmost column we have sentences from the dialogue along with their labels, which are the examples the model is going to classify. On the right column we have the corresponding sentence representation considering $x = 1$. The first sentence, s_1, does not have any context so we end the representation with a `[SEP]` token. The second sentence, s_2 has $s1$ as context, so after the first `[SEP]` token, we put s_1, followed by another `[SEP]` token. The same happens for subsequent sentences in the dialogue.

4.3 Sentiment Classification using Retrieval Augmentation

Due to the recent success of retrieval augmentation approaches in NLP tasks (Dinan et al., 2018; Moghe et al., 2018; Lewis et al., 2020), including for sentiment classification (Ramos et al., 2021), we will explore a mechanism that relies on nearest neighbors to aid classification. In particular, the work by Ramos et al. (2021) uses this logic for approaches that are based on the LSTM model. Since we are dealing with Transformer models, whose architecture is significantly different, we will adapt their approach to this model, with a focus on how to provide the retrieved labels as input.

4.3.1 Retrieve Nearest Neighbors

The first part of this method involves finding the nearest training example for each train/validation/test example. To do so, we use the Sentence Transformer (Reimers and Gurevych, 2019)[1] library to create sentence representations of all examples using the `paraphrase-distilroberta-base-v1` model. Next, we make use of the FAISS (Johnson et al., 2019)[2] library to build an index with the sentence embeddings that belong to the train set, and also to find the closest training examples. In particular, we chose the Euclidean distance to calculate the distances between the examples. For each example in the train/validation/test sets, we want to find the closest example in the training set, which is described by,

$$j = argmin_i ||x - x_i||, \tag{4.2}$$

with x_i being the set of embeddings in the index, and x the new embedding from one of the three sets.

Finally, each training/validation/testing example is assigned a label corresponding to the label of the closest training example (or the second closest training example for the training data, to avoid retrieving the examples' own label). The goal is to incorporate this information into the Transformer model in order to guide it towards the corresponding retrieved label.

4.3.2 Incorporate Sentiment Embeddings

After retrieving the nearest neighbors information, we need to apply it to the Transformer, which is where our method differs from Ramos et al. (2021), that, as previously mentioned, applies it to LSTM models. In order to do so, we first initialize an extra set of embeddings, which we will call Sentiment Embeddings (SE), one for each sentiment label, which are trained alongside the model. Then, for each example, we incorporate the embedding corresponding to the nearest training example label in the Transformer model. To do so, we explored two distinct approaches:

- **sum**: As described in Section 2.3.3, the input of the BERT architecture is the sum of three different sets of embeddings: the token embeddings, the segment embeddings, and the position embeddings. In this approach, we sum the aforementioned extra set of embeddings, the sentiment embeddings, to the input of the model, according to the label of the retrieved example.

[1] https://www.sbert.net
[2] https://faiss.ai

- **concat**: In this approach, the embedding corresponding to the label of the nearest training example is concatenated to the output of the Transformer model, which we will represent by Out. The resulting vector is the input to the classification layer which is defined as,

$$\mathrm{concat}\big(\mathrm{Out}; \mathrm{SE}_i\big), \tag{4.3}$$

  with i being the index of the label corresponding to the nearest training example.

Regarding the initialization of the sentiment embeddings we experimented with two approaches:

- **no init**: the sentiment embeddings are initialized randomly, following a normal distribution, $\mathcal{N}(0, 1)$.
- **init**: we were inspired by the initialization of the memory state of the LSTM described in Ramos et al. (2021), where different initialization approaches were experimented with. Since the Transformer model does not have a memory state, we initialize instead the embedding of each sentiment in SE with the average of the sentences' corresponding to that given sentence. E.g., the sentiment embedding for *joy* will be initialized with the average of the embeddings corresponding to the sentences labelled with the sentiment *joy* in our training set. For this use case, we use the average of the token embeddings of the last hidden layer as the embedding of a given sentence.

4.4 Reply Sentiment Prediction

Having defined the sentiment classification model and how context and retrieval augmentation can be used to improve it, next, we will explore how to choose an appropriate sentiment to guide the text generation model. To do so, we introduce the Reply Sentiment Prediction model, which is the second model in our proposed system's pipeline. This model receives as input the previous context of a dialogue in order to predict the appropriate sentiment to be expressed in the conversational agent's next reply. This aspect makes it crucial for the usage of the system, given that it will be the model that will allow the conversational agent to be sentiment-aware.

4.4.1 Basic Input

In Section 3.3 we described some approaches to implement the reply sentiment prediction model, and, in particular, how the input for this model is built. Similarly to the contextual sentiment classification described in Section 4.2, we will also make use of the `[SEP]` token to separate the different sentences that are part of the input. However, there are two main differences: first, the input of the reply sentiment prediction model is only the previous context sentences; secondly, the gold label is the sentiment of the upcoming sentence. In Table 4.2 we can observe the same sentences we saw in Section 4.2, but with the changes on the representations this task requires. The gold label of each representation will now be the sentiment of the upcoming sentence. The last sentence in a dialogue will not be considered as a valid example, given that there are no sentences that follow it. In this task we can also use an arbitrary number of sentences as context. We will use as default setup two sentences, one from the conversational agent and one from the user. The goal is then to predict the sentiment that should be expressed by the conversational agent in the upcoming message.

Sentence	Label	Representation	Label
Does it cost anything?	NEU	`[CLS]Does it cost anything?[SEP]`	NEU
Yeah 20$ per month.	NEU	`[CLS]Yeah 20$ per month.[SEP]Does it cost anything?[SEP]`	SUR
Ohh!	SUR	-	-

Table 4.2: Example of a dialogue from the EmotionPush dataset and the sentence representation and corresponding label for the response sentiment prediction task, with two previous sentences as context ($x = 2$). NEU corresponds to the label *neutral*, and SUR to the label *surprise*.

4.4.2 Adding Sentiment Context

Now that we described the basic input for the reply sentiment prediction model, we can explore additional methods to add more information to the input of the model. In addition to the retrieval augmentation, described in Section 4.3, which will be used in a similar manner for this task, we can also add to the input of the classification layer the sentiment of the sentences from the context.

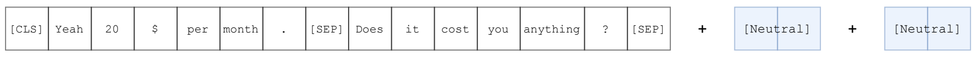


Figure 4.3: Example of the input of the classification layer of the reply sentiment prediction model with information from the labels of the context sentences. This method involves concatenating vector representations of the sentiments corresponding to the context sentences.

In Figure 4.3, we can observe an example of the input of the classification layer reply sentiment prediction model using the second sentence representation from Table 4.2. As we saw before, the labels of the context sentences are both the label *neutral*. This method involves concatenating a vector representation of the context labels to the embeddings output by the contextual model, similarly to what was done in the **concat** method of the retrieval augmentation described in Section 4.3.2.

4.5 Summary

In order to guide the task of conditioned text generation and be able to apply it to a dialogue conversational agent, we proposed using a sentiment classifier and a response sentiment prediction model as a way to feed the text generation model information about the appropriate sentiment to generate. Additionally, the sentiment classifier model is key to automatically evaluate the quality of the sentiment-conditioned text generation model. In this section we started by defining the task of sentiment classification. Next, we explored methods to improve the baseline models such as taking advantage of being in a dialogue context by adding previous context sentences, and using retrieval augmentation approaches to bias the models towards the correct sentiment labels. Finally, we defined the task of reply sentiment prediction which, given contextual information, such as previous sentences and its sentiment labels, aims to predict the sentiment of an upcoming sentence.

Chapter 5

Guiding Dialogue Text Generation with Sentiment

Traditionally, state-of-the-art dialogue conversational agents rely on large amounts of conversational data to train text generation models, which are able to generate human-like responses, both regarding the context of the conversation and having lexical diversity (Zhang et al., 2020). In particular, Rashkin et al. (2019) show that state-of-the-art text generation models can be fine-tuned with appropriate dialogue empathetic data, and express emotions. However, when there is not enough data (Miao et al., 2020), or the data is not appropriate (Rashkin et al., 2019), these approaches might lead to models which are unable to implicitly generate text with the aforementioned characteristics. Our goal is to develop a system which is able to have those characteristics, in particular, by taking advantage of explicit sentiment information.

In Chapter 4, we proposed two models that can be part of a guided dialogue text generation pipeline, which allow to understand the sentiment of the context sentences, and predict the appropriate sentiment that should be expressed next. What is left is to describe the text generation model we are going to use in order to incorporate the sentiment information. During a dialogue, this model will receive the sentiment predicted by the reply sentiment prediction model. Then, it will use this information, as well as the previous context of the conversation, to generate a sentence that is not only appropriate given a context, but that expresses the predicted sentiment.

In this chapter, we will begin by describing the base model (Section 5.1). We will make use of pre-trained Transformer language models, such as the GPT-2 (Radford et al., 2019), which have been shown to provide strong baselines for text generation. Next, we will describe how we will improve upon the baseline. In particular, we are going to adapt the work by Wolf et al. (2019), which proposes a model that is able to leverage a set of sentences that describe a given persona in order to generate text that is coherent with the persona. We will adapt this concept and develop a sentiment lexicon knowledge base, which will be used to make the conversational agent sentiment-aware (Section 5.2). In addition to this, we will also describe how we are going to adapt the next sentence prediction approach, proposed by Devlin et al. (2019) and used to fine-tune the model proposed by Wolf et al. (2019), through the use of distractors, for our specific use-case (Section 5.2.2).

5.1 Base Model

In Section 2.4.3, we described one of the current state-of-the-art text generation architectures, the Generative Pre-trained Transformer (GPT), which has had three iterations: GPT-1 (Radford et al., 2018), GPT-2 (Radford et al., 2019), and GPT-3 (Brown et al., 2020). Since the original GPT-3 model is still not publicly available, and due to the computational costs of similar implementations (Black et al., 2021), we will use the GPT-2 model for our work. In this section, we will describe in more detail the GPT-2 architecture, and explain how it is used for dialogue text generation.

In Figure 5.1, we can observe the input and output of the GPT-2 model. In order to make a language model such as the GPT-2 suitable for the dialogue task, as described in Wolf et al. (2019), we concatenate previous context of the dialogue (history) to the input of the model. This is possible by introducing two extra special tokens to the model, `<speaker1>`, which indicates the beginning of a sequence from the user, and `<speaker2>`, which indicates the beginning of a sequence from the bot, in addition to the existing special tokens `<BOS>`, which indicates the beginning of a sequence, and `<EOS>`, which indicates the end of a sequence. Furthermore, in this figure, we can also observe the autoregressive property of this model. As described in Section 2.2.4, autoregressive language models use the left-wise context of a sentence in order to calculate the probability of the next word. This means that this model uses previously generated words as input to generate new ones. At each timestep t, the input of the model is $\mathrm{concat}(h; w_{0:t-1})$, with h representing the history, and $w_{0:t-1}$ representing a sequence of generated words from timestep 0 until $t-1$. The output is w_t, which is the word generated at timestep t.

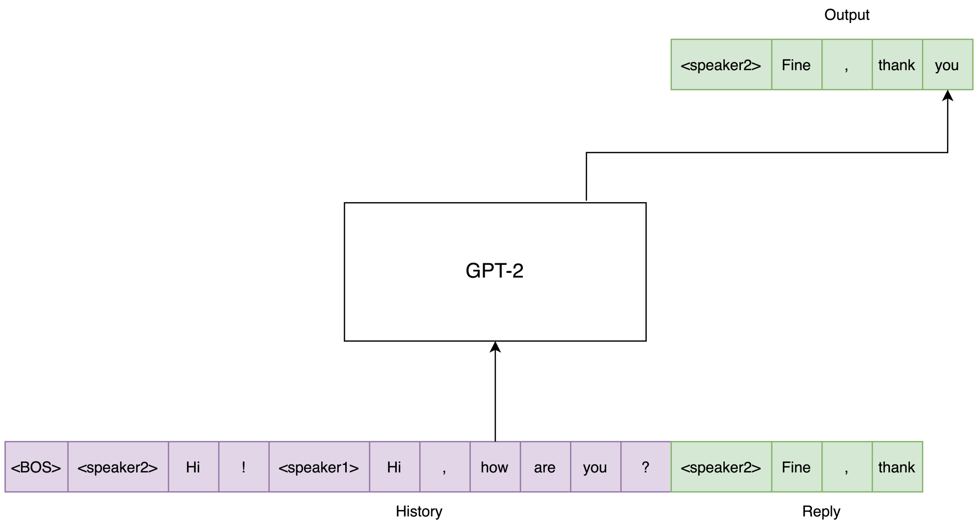


Figure 5.1: Example of the input and output of the base GPT-2 model adapted for the dialogue task.

5.2 Condition Model with Sentiment

In order to condition the text generation base model with a desired sentiment, we follow the work of Wolf et al. (2019) which proposes two increments to the base model: the first, at the input level, where sentences that describe a persona are concatenated to the input of the model; and the second, at the fine-tuning level, which uses a double-head Transformer model for multi-task learning that jointly fine-tunes the model for two tasks, language modelling (the original task) and next sentence prediction. The next sentence prediction task, proposed by Devlin et al. (2019), is implemented in this work through the use of distractors. To do so, between two and six randomly selected sentences (distractors) for each example in the corpus are concatenated to that input's history, creating two to six extra examples. The goal of the next sentence prediction head is to classify whether the sentence appended to the input is correct. In this section, we will describe how we are going to adapt these two approaches to build a generation model that is conditioned on a given sentiment.

5.2.1 Adding Sentiment Lexicon

We will start by improving the base model by introducing information from a knowledge base in order to guide the generation. Instead of a persona, we will input lexicon that represents the desired sentiment. This is done by concatenating the lexicon to the beginning of the input of the model. An advantage of this model is that we can experiment with various sentiment lexicons, which can consist of a single word for each sentiment (for example, the name of the sentiment), expressions, or full sentences. In Figure 5.2 we illustrate how concatenating sentiment lexicon to the input of the GPT-2 model might guide the generation model to output a sentence that expresses the desired sentiment. For instance, by adding the sentiment lexicon of the *anger* emotion, we expect the model to produce a more sentiment appropriate sentence ("I am annoyed"), than the base model ("Fine, thank you"), given the same history.

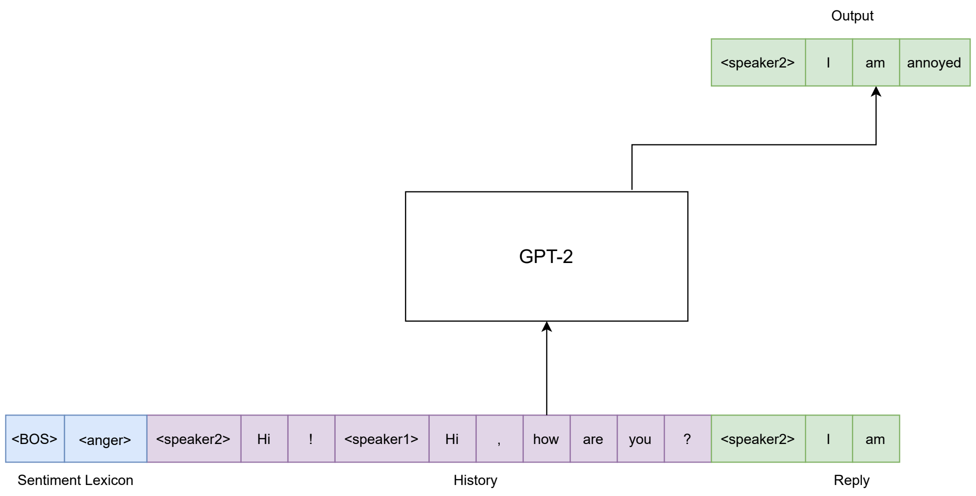


Figure 5.2: Example of the input and output of the sentiment controlled GPT-2 model through the concatenation of sentiment lexicon to the input of the model.

5.2.2 Next Sentence Prediction

The work by Wolf et al. (2019) also describes a training particularity regarding the loss that is calculated to update the weights of the model. The model used is called a double head Transformer and is trained for two different tasks. First, the language model head, is tasked with generating the tokens, from which a language model loss, $\mathcal{L}_{\text{LM}}$, is calculated. Second, the next sentence prediction head, task proposed in Devlin et al. (2019), is evaluating whether the reply is appropriate given a history and a sentiment lexicon. From this task, a next sentence prediction loss, $\mathcal{L}_{\text{NSP}}$, is calculated. Then, the model's weights are updated conditioned on the weighted sum of the language model loss, and the next sentence prediction loss, $\mathcal{L} = \alpha\mathcal{L}_{\text{LM}} + \beta\mathcal{L}_{\text{NSP}}$, with $\mathcal{L}$ representing the total loss, and α and β the weights given to each of the losses.

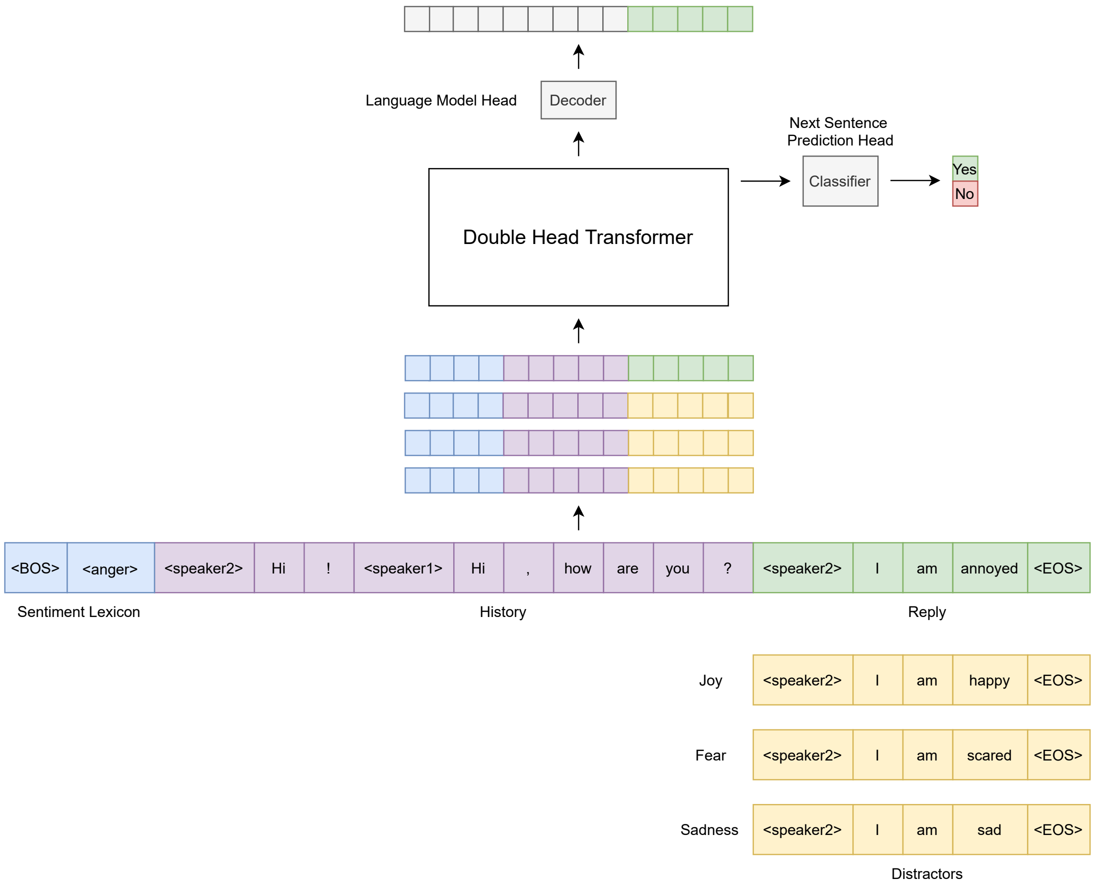


Figure 5.3: Example of the training of the double head transformer. The first head, the language model head, is tasked with generating a reply given a history and the sentiment lexicon. The second head, the next sentence prediction head, given four similar inputs but with different reply possibilities (the gold and the distractors) is tasked with classifying whether they are appropriate given the sentiment lexicon and the history.

The next sentence prediction task is trained through the use of distractors, as exemplified in Figure 5.3. During training, the model is not only learning to generate sentences, but also to classify whether the provided training sentences are appropriate given the history and the desired sentiment. The goal of this architecture is to teach the model which sentences are appropriate to answer, but in particular, which are not. To do so, in addition to the gold example, which includes the sentiment lexicon, history, and the gold reply, we also provide the model three additional examples, where instead of concatenating the gold reply, we concatenate a random sentence from the training set that is labelled with a different sentiment than the one the model is supposed to generate. For example, in Figure 5.3, the goal is to generate the sentence “I am annoyed”, which expresses the *anger* sentiment. As distractors, three sentences labelled with the sentiments *joy*, *fear*, and *sadness* are used.

5.3 Summary

Having the reply sentiment prediction model as a way to inform the text generation model of the appropriate reply emotion, we proposed a text generation model that is able to leverage this information to condition its output on the desired sentiment. In particular, we adapted the work by Wolf et al. (2019), that is able to express a given persona, to our use-case. We started by introducing the base model, which uses four new special tokens in order to make the text generation model suitable for the dialogue task. Next, we saw how we can concatenate sentiment lexicon to the input of the model in order to guide it towards the desired sentiment. Finally, we adapted the next sentence prediction task, with the use of sentiment distractors, as a way to teach the model which sentences are not appropriate to answer given the desired sentiment and a history.

Chapter 6

Experimental Analysis

In this chapter we will develop and present the results of the approaches proposed in Chapters 4 and 5 applied to the EmotionPush (Hsu et al., 2018) and the DailyDialog (Li et al., 2017) corpora. Throughout the experimental cycles, we will follow an incremental method comparing models' scores using the validation set. The first step of each model's experimental cycle will be to choose a baseline model, followed by exploring the impact of one increment to the model at a time. We will use the EmotionPush corpus to test the increments, and later apply our findings to the DailyDialog corpus.

We will start this chapter by introducing the datasets we are going to use, along with some useful statistics that are going to help us understand the results during the experimental cycles of each model. Next, we will explore the task of sentiment classification, in particular, we will take advantage of being in a dialogue scenario and use previous context sentences as extra input to our model. We will also experiment with retrieval augmentation methods, as a way to further improve the results. Similar techniques will be applied during the reply sentiment prediction experimental cycle. Then, we will present the experimental cycle of the text generation model, which consists on adapting the work by Wolf et al. (2019), described in Chapter 5, to our specific use-case. The best setups found for each model will be used in order to build the sentiment-aware conversational agent proposed in Chapter 1, and showcase its results. Furthermore, in Section 6.2.9 we compare the performance of the developed sentiment classifier versus other recent works that use the same datasets as we do. In particular, we only do so for the task of sentiment classification since, to the best of our knowledge, the datasets we are going to experiment with have only been used for that task out of the tasks we are working with.

6.1 Datasets

Transformer based models have been shown to perform well when trained and fine-tuned with large amounts of data (Devlin et al., 2019; Liu et al., 2019; Radford et al., 2019), but it is time and resource consuming to build quality corpora. Namely, for specific domains, such as customer support, there are constraints to retrieving data, such as clients' privacy and data protection. As mentioned in Section 3.1, we were not able to find any publicly available customer support corpora that was both in a dialogue setting and labelled on every sentence, thus, we will not be able to apply our work to the customer support scenario. Nonetheless, we found other dialogue corpora labelled with sentiments, but in an open-domain setting. In particular, we will focus the development of this work on small datasets that use larger emotion taxonomies, such as the Ekman emotions (Ekman, 1992). Such characteristics will allow us to simulate the difficulty of creating and labelling these kinds of datasets, which tend to be small and not well balanced, thus reflecting our goal: to build a quality sentiment-aware conversational agent, using Transformer-based models, when the amount of data is limited.

In order to evaluate our work, we will use the EmotionPush portion of the EmotionLines (Hsu et al., 2018) corpus, a corpus with 10733 training examples, to represent a small and unbalanced corpus, and the DailyDialog (Li et al., 2017) corpus, with 87170 training examples, to see how our models behave on a scenario where more data is available. Throughout the experimental analysis we will be developing and testing our models using the EmotionPush corpus, given that it is the closest corpus to the real scenario we are aiming for. Furthermore, given that this work has a very extensive experimental cycle, using a smaller corpus also results in faster training times, allowing us to make more experiments.

Both datasets are labelled with the six Ekman emotions (Ekman, 1992) (*fear*, *anger*, *joy*, *sadness*, *disgust*, and *surprise*). The EmotionPush corpus has two additional labels: *non-neutral*, to represent sentences the annotators could not find agreement on; and *neutral*. The DailyDialog corpus has only one additional label, *no emotion*. Furthermore, the *joy* sentiment is called *happiness* in this dataset.

	EmotionPush			**DailyDialog**		
	Train	**Dev**	**Test**	**Train**	**Dev**	**Test**
Number of dialogues	720	80	200	11118	1000	1000
Number of sentences	10733	1202	2807	87170	8069	7740

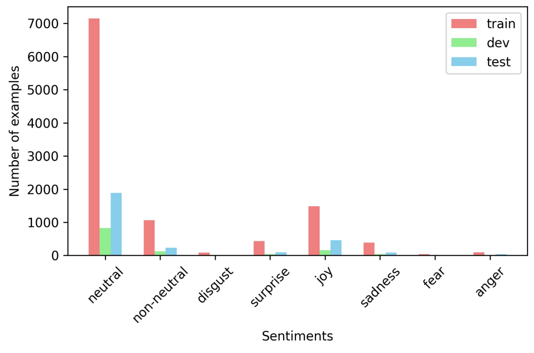
Table 6.1: Number of dialogues and sentences of the train/development/test splits of the EmotionPush and DailyDialog datasets.

In Table 6.1 we can observe some basic statistics about the EmotionPush and DailyDialog corpora. As mentioned before, we chose one small dataset and one larger to develop our work. The EmotionPush is composed of 1000 dialogues and 14742 sentences in total. The DailyDialog is larger: it has 13118 dialogues and 102979 sentences. In order to further simulate a customer support setting, in a given dialogue, we will always assume that the first speaker is the chatbot and the second speaker, the user.

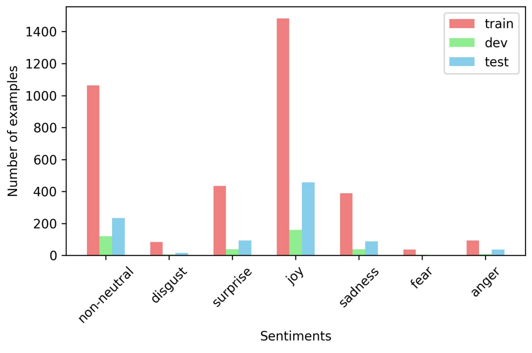
6.1.1 Splits and balancing

In Figures 6.1 and 6.2 we can observe the balancing of the EmotionPush and DailyDialog corpora, respectively. Similarly to Kim et al. (2020) and Hazarika et al. (2021), we will consider both datasets with and without the majority class (neutral/no emotion) for evaluation purposes.

The first observation we can make, by analysing Figures 6.1a and 6.2a, is that the majority class (*neutral*/*no emotion*) largely surpasses the remaining classes in number of examples in all sets. This can be further observed in Tables 6.2 and 6.3, where we show the datasets' balancing in both absolute and relative values. In the EmotionPush dataset, the *neutral* and *non-neutral* sentiments make up

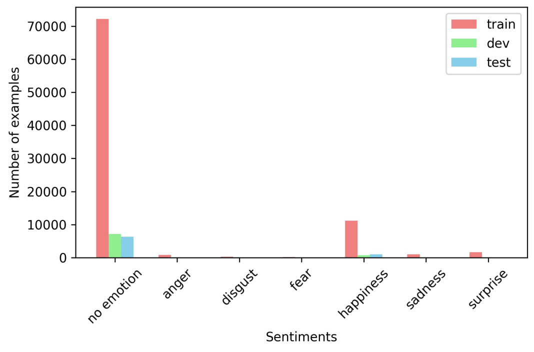


(a) With the majority class.

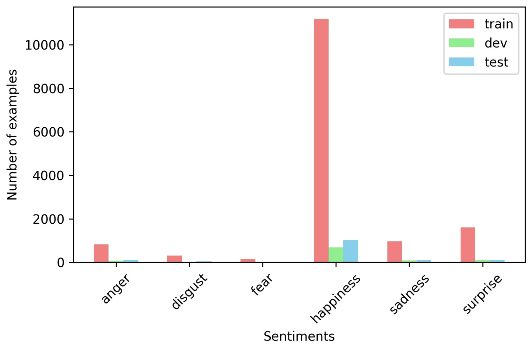


(b) Without the majority class.

Figure 6.1: Balancing of the EmotionPush corpus with and without the majority class (neutral).



(a) With the majority class.



(b) Without the majority class.

Figure 6.2: Balancing of the DailyDialog corpus with and without the majority class (no emotion).

almost 80% of the whole corpus. In the DailyDialog dataset, the sentences labelled with *no emotion* also represent around 80% of the corpus.

Furthermore, Figures 6.1b and 6.2b show us that, even among non majority classes, the datasets are unbalanced, with the *joy*/*happiness* sentiments having the most examples in all sets.

		Neutral	**Non-neutral**	**Disgust**	**Surprise**	**Joy**	**Sadness**	**Fear**	**Anger**
Train	**Abs.**	7148	1064	85	435	1482	389	36	94
	%	66.6	9.91	0.79	4.05	13.81	3.62	0.34	0.88
Dev	**Abs.**	825	121	6	39	160	38	4	9
	%	68.64	10.07	0.5	3.24	13.31	3.16	0.33	0.75
Test	**Abs.**	1882	233	15	93	458	87	2	37
	%	67.05	8.3	0.53	3.31	16.32	3.1	0.07	1.32

Table 6.2: EmotionPush corpus balancing in both absolute and relative values.

The information of the balancing of the corpora is very important to understand and correctly evaluate our models. In particular, for the sentiment classification and reply sentiment prediction tasks, the fact that some classes are poorly represented, will influence the results obtained. For instance, as observed in Table 6.2, on the EmotionPush corpus we only have two examples of the *fear* label in the test set. This means that if one model is able to classify both examples and another model is unable to classify one of them, the *fear*-F1 will be drastically different between both models. Because of this, the macro-F1,

		No emotion	Anger	Disgust	Fear	Happiness	Sadness	Surprise
Train	**Abs.**	72143	827	303	146	11182	969	1600
	%	82.76	0.95	0.35	0.17	12.83	1.11	1.83
Dev	**Abs.**	7108	77	3	11	684	79	107
	%	88.09	0.95	0.04	0.14	8.48	0.98	1.32
Test	**Abs.**	6321	118	47	17	1019	102	116
	%	81.67	1.52	0.61	0.22	13.17	1.32	1.49

Table 6.3: DailyDialog corpus balancing in both absolute and relative values.

which is obtained by averaging the F1 score of all labels, will be greatly impacted. This behavior impacts both datasets for other labels such as *disgust*, *sadness*, or *anger*. This is not an issue for micro-F1, since this metric considers all individual examples as equal.

It is important to notice that, despite the macro-F1 metric not being trustworthy to compare models because of the aforementioned problems, we still optimized the sentiment classification and reply sentiment prediction models to maximize it. This is because, for our use-case, it is important to have a model that is able to classify examples of all classes rather than to have one that classifies correctly the most examples, particularly because of the poor datasets' balancing.

Regarding the experimental analysis, in order to evaluate the classification tasks, and considering the fluctuation of the macro-F1, we will prioritize the micro-F1 metrics (with and without the majority class). What we will notice in our experimental analysis is that a high micro-F1 does not necessarily translate to a high quality model that classifies correctly examples of all classes. Therefore, we will also leverage the macro-F1 metrics (with and without the majority class) and F1 metrics of each sentiment, in order to take into consideration whether the model is being able to classify examples of all classes.

Text Generation Pre-processing

The EmotionPush dataset has a particularity that needs to be addressed for the task of text generation. The behavior of the dialogue chatbot we are developing only expects a message from the user at each turn, and only replies with a single message as well. In the EmotionPush dataset we noticed that some subsequent sentences are of the same speaker, which poses a problem for text generation. In order to deal with this, we performed the following changes:

- If the speakers on subsequent sentences are the same, join the sentences;
- If the labels of the sentences are the same, then the new example will have that label;
- If the labels of the sentences are different, we choose the least common label.

This process results on a smaller corpus of around 8000 examples in total. The EmotionPush balancing after the changes done can be observed in Table 6.4.

		Neutral	Non-neutral	Disgust	Surprise	Joy	Sadness	Fear	Anger
Train	**Abs.**	3555	432	80	370	1116	322	34	82
	%	59.34	7.21	1.34	6.18	18.63	5.37	0.57	1.37
Dev	**Abs.**	409	49	5	35	125	34	4	7
	%	61.23	7.34	0.75	5.24	18.71	5.09	0.6	1.05
Test	**Abs.**	969	97	15	78	355	71	2	26
	%	60.07	6.01	0.93	4.84	22.01	4.40	0.12	1.61

Table 6.4: EmotionPush corpus balancing after the changes necessary for text generation in both absolute and relative values.

6.2 Sentiment Classification

In this section we will focus on finding the best possible sentiment classification model for our corpora.

The default sentence embedding representation and classification layer we are going to use are the same as the ones used in Devlin et al. (2019). We will make use of the `[CLS]` token of the last hidden layer of the Transformer model as the embedding representation, and a linear layer as the classification model, both described in Section 4.1. Finally, we will perform an ablation study on the best model obtained in order to validate our results using the test set.

6.2.1 Metrics

In order to evaluate the sentiment analysis models, we will focus on the *F1* metric. In particular, because of the poor balancing of the datasets used, we will calculate the micro (*m*) and macro (*M*) averages, described in Section 2.5.2. The *macro-F1* evaluates whether the model is able to classify examples of all labels equally, while the *micro-F1* evaluates the number of examples classified correctly. Furthermore, given the distribution of examples labelled as *neutral* (the majority class) vs. all other labels (Section 6.1.1), we will also evaluate our models both with the majority class and without the majority class (micro/Macro-No Majority Class metric which we will refer to as *m/M-NMC*). This will allow us to have a better understanding of how the models are performing on less represented sentiments. Finally, if needed, we will also take into consideration the *F1* metric of each individual sentiment.

6.2.2 Training Details

The work described in Chapter 4 was implemented using PyTorch Lightning (Falcon and The PyTorch Lightning team, 2019) and the HuggingFace Transformers (Wolf et al., 2020) library. In particular, we modified the code base of the HLT-MAIA Emotion-Transformer repository[1].

The models were trained for a maximum of 40 epochs, using the cross entropy loss, with four validation steps per epoch, stopping the training after 10 consecutive validation steps without improvement. The checkpoint used to evaluate the model was the one that achieved the highest validation macro-F1 value. We follow the work by Howard and Ruder (2018) and use the Adam optimizer (Kingma and Ba,

[1]https://github.com/HLT-MAIA/Emotion-Transformer

2015) with a discriminative learning rate of 1×10^{-3}, except for the Transformer model that has a learning rate of 5×10^{-6}. For the Transformer model we apply a layer-wise learning rate decay of 0.95 after each training step. We apply a dropout (Srivastava et al., 2014) of 0.4 to the sentence embeddings during training. The checkpoint used to evaluate the model was the one that achieved the best *macro-F1* result on the development set. The models were trained on a GeForce GTX 1080 GPU, with a real batch size of 16 whenever the GPU's memory allowed it, but we used gradient accumulation to always simulate a batch size of 32. All the other hyperparameters were kept with their default values.

6.2.3 Baseline

In order to develop our classification models we used two pre-trained Transformer models, BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019) in two different configurations, base and large. We chose the BERT and the RoBERTa models because both have been shown to perform well when further fine-tuned to solve different classification tasks (Du et al., 2020; Chen et al., 2020). All models are cased, meaning, they can differentiate between upper and lowercase characters. The BERT model is trained for the tasks of masked language modelling, described in Section 2.2.5, and next sentence prediction, described in Section 2.3.3. The RoBERTa model differs from BERT by being trained with more data for longer, different hyperparameter values, and only for the task of masked language modelling. The models we are going to experiment with are obtained from the HuggingFace Transformers library and are the following: BERT-BASE-CASED; BERT-LARGE-CASED; ROBERTA-BASE; and ROBERTA-LARGE. Further details of the models' configurations can be observed in Table 6.5.

Model	Nr. of Layers	Hidden Size	Total Parameters
BERT-base-cased	12	768	110M
BERT-large-cased	24	1024	340M
RoBERTa-base	12	768	125M
RoBERTa-large	24	1024	355M

Table 6.5: Description of BERT and RoBERTa models used in two different configurations, base and large.

	F1				F1							
Model	m	m-NMC	M	M-NMC	ANG	DIS	FEA	JOY	NEU	NNEU	SAD	SUR
BERT-base-cased	76.4	49.5	39.0	32.1	19.2	0	0	61.3	86.8	31.1	53.5	59.0
BERT-large-cased	76.3	**52.9**	43.2	37.0	**52.9**	0	0	64.9	86.5	**31.9**	**53.9**	**62.3**
RoBERTa-base	75.9	51.4	**51.0**	**45.9**	35.3	**50.0**	**40.0**	65.7	86.7	31.4	44.5	54.6
RoBERTa-large	**76.8**	51.9	47.5	41.8	28.6	25.0	33.3	**66.5**	**87.2**	29.6	29.6	55.7

Table 6.6: Results obtained on four different architectures (BERT-base-cased, BERT-large-cased, RoBERTa-base, RoBERTa-large) on the development set of the EmotionPush dataset.

The results obtained for the baseline approach using the aforementioned Transformer models on the development set can be observed in Table 6.6. Regarding the *micro-F1* metric, the RoBERTa-large model achieves the highest score of 76.8. If we consider the same metric without the majority class, the model that achieved the highest score is the BERT-large-cased. Despite this, we can observe that both BERT models resulted in scores of 0 in both *disgust* and *fear*-F1, contrarily to the RoBERTa models, which were able to classify examples belonging to all labels. Due to that, we will not choose the BERT models. We are left with a choice between RoBERTa's base and large versions. We will use the latter, since it performs better on the *micro-F1* scores.

6.2.4 Sentence Embedding Representation

In order to classify an input sentence, we need to choose a suitable representation of the embeddings output by the Transformer model. Lets consider $E \in \mathbb{R}^{\text{hidden size} \times \text{sentence size} \times \text{number of layers}}$ as the total set of embeddings output by the model. As described in Section 4.1 the standard embedding representation technique was proposed in Devlin et al. (2019) and it consists on using the embedding corresponding to the [CLS] token of the last hidden layer, which we will refer to $E^{-1}_{\texttt{[CLS]}}$, as the representation of a given sentence. In addition to this, we will also experiment with the following options:

- **avg**: calculates an average of all embeddings of the last hidden layer of the model, from the [CLS] token to the last [SEP] token,
$$\text{avg}\left(E^{-1}_{\texttt{[CLS]:[SEP]}}\right) \tag{6.1}$$
- **cls+avg**: sums the [CLS] token with the average of all embeddings of the last hidden layer,
$$E^{-1}_{\texttt{[CLS]}} + \text{avg}\left(E^{-1}_{\texttt{[CLS]:[SEP]}}\right) \tag{6.2}$$
- **cls avg**: concatenates the [CLS] token and the average of all embeddings of the last hidden layer,
$$\text{concat}\left(E^{-1}_{\texttt{[CLS]}}; \text{avg}(E^{-1}_{\texttt{[CLS]:[SEP]}})\right) \tag{6.3}$$
- **biLSTM**: all the embeddings of the last hidden layer are input into a biLSTM,
$$\text{biLSTM}\left(E^{-1}_{\texttt{[CLS]:[SEP]}}\right) \tag{6.4}$$
- **concat** n: concatenates n [CLS] tokens belonging to the last n hidden layers,
$$\text{concat}(E^{[-1:-n]}_{\texttt{[CLS]}}). \tag{6.5}$$

  For this method we experimented with $n = \{2, 4\}$

The results obtained can be observed in Table 6.7. The pooling methods that achieve the best *micro-F1* scores are the *biLSTM* and the *cls avg*, but both achieve scores of 0 in the *anger*, *disgust*, and *fear*-F1, which affects the ability of the model to generalize to all sentiments. The next best performing models are the *cls+avg* and *concat4* models, achieving 77.4 and 77.2 *micro-F1* scores respectively, and 53.0 and 53.3 *micro-F1 without the majority class* scores. Given that the results are so similar, we will consider the *macro-F1* metrics. We can observe that the *concat4* model achieves the best scores in both

	F1				F1							
Pooling	**m**	**m-NMC**	**M**	**M-NMC**	**ANG**	**DIS**	**FEA**	**JOY**	**NEU**	**NNEU**	**SAD**	**SUR**
cls	76.8	51.9	47.5	41.8	28.6	25.0	33.3	66.5	87.2	29.6	29.6	55.7
avg	76.5	52.2	47.9	42.2	15.4	25.0	**57.1**	68.0	87.3	22.7	51.4	56.0
cls+avg	77.4	53.0	47.7	42.0	28.6	25.0	33.3	67.9	87.3	28.6	54.8	56.0
cls avg	**77.9**	54.1	37.4	30.2	0	0	0	66.3	**87.7**	37.9	54.8	52.2
biLSTM	77.3	**55.2**	38.3	31.2	0	0	0	68.2	87.5	**39.5**	**55.0**	**56.1**
concat2	75.3	52.3	48.7	43.4	**42.9**	26.7	33.3	67.2	86.3	30.0	51.0	52.5
concat4	77.2	53.3	**49.2**	**43.8**	30.7	**28.6**	44.5	**69.3**	87.4	27.6	53.1	52.5

Table 6.7: Results obtained with the RoBERTa large model using seven different pooling techniques.

macro-F1 metrics out of all variants, and also improves all metrics when compared to the *cls* version, so we will choose the *concat4* as our preferred pooling method.

6.2.5 Contextual Sentiment Classification

As described in Section 4.2, we will take advantage of being in a dialogue setting to also use context sentences, belonging to both the user and the conversational agent, as input to our model in order to give it additional information about the dialogue. To do so, we will experiment adding 1, 2, 3 and 4 sentences as context to the previous model (RoBERTa-large with *concat4* pooling). We did not experiment with more than four sentences as context due to computational constraints.

	F1				F1							
Context	**m**	**m-NMC**	**M**	**M-NMC**	**ANG**	**DIS**	**FEA**	**JOY**	**NEU**	**NNEU**	**SAD**	**SUR**
$x = 0$	77.2	**53.3**	49.2	43.8	30.8	28.6	**44.5**	**69.3**	87.4	27.6	53.1	52.5
$x = 1$	75.8	53.1	**53.1**	**48.3**	40.0	54.6	40.0	67.5	86.6	**31.0**	55.0	50.0
$x = 2$	**78.5**	52.4	51.0	45.6	**53.3**	**66.7**	0	65.8	**88.2**	25.3	53.8	54.6
$x = 3$	75.3	51.8	51.3	46.2	40.0	50.0	33.3	67.1	86.5	28.3	54.4	54.5
$x = 4$	77.5	53.2	48.7	43.2	41.7	54.6	0	67.4	87.4	21.7	**59.7**	**57.1**

Table 6.8: Results obtained with the RoBERTa large model, with the concat4 pooling technique, using between zero and four sentences as context.

The results can be observed in Table 6.8. The models that achieve the best *micro-F1* metrics are the ones with no context ($x = 0$), with two sentences as context ($x = 2$), and with four sentences as context ($x = 4$). Regarding the $x = 2$ and $x = 4$ models, both have a score of 0 in the *fear*-F1 metric, which is not desirable. Next, we will compare the $x = 0$ model with the next best performing model, which is the one with one sentence as context ($x = 1$). Even though the *micro-F1* metric is 1.4 points lower than the obtained on the $x = 0$ model, if we consider the *m-NMC* metric, then the difference is only 0.2 points, which makes it hard to choose a model only considering the *micro-F1* metrics. If we look at the *macro-F1* metrics, the difference is more expressive, with the $x = 1$ model scoring 3.9 points above on the *M* metric and 4.5 on the same metric if we do not consider the majority class. Given the improvement on these metrics, plus the fact that this model is able to get non-zero scores for all sentiment-level F1, we will choose the $x = 1$ as the best model.

6.2.6 Retrieval Augmentation

The final method we experimented was to use retrieval augmentation to aid the classification. This method is described in Section 4.3, and is based on the work by Ramos et al. (2021), where information from the nearest training example is used to initialize the memory state of an LSTM. Our approach uses this idea and applies it to the Transformer model. We incorporate the nearest training example of a given sentence to aid its classification, using two approaches: **sum**, where the embedding corresponding to the label of the nearest training example is added to the input embeddings of the Transformer model; and **concat**, where the same sentiment embedding is concatenated to the output of the Transformer model, to be used only on the classification step. Furthermore, we also experiment both approaches with and without the initialization of the sentiment embeddings. The initialization is done using the average sentence embedding representation for a given sentiment, using the training data.

Retrieved Labels Evaluation

Before using the nearest neighbors information in our models we are first going to evaluate the quality of the bias we are passing to the model. This means that we first want to assess how good the labels we retrieve are.

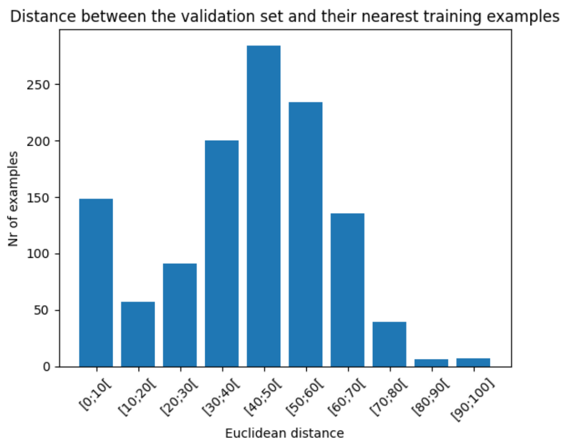
The retrieval was done using a top-n method: using the FAISS index, described in Section 4.3.1, for each train/development/test example we retrieve the n most similar sentences from the training set, and assign the target example the majority voting among the n examples. If is it not possible to reach a quorum, we use $n = 1$, which means we assign it the label of the most similar example from the training set. In our experiments we used $n = \{1, 3, 5\}$.

	F1				F1							
Retrieval	**m**	**m-NMC**	**M**	**M-NMC**	**ANG**	**DIS**	**FEA**	**JOY**	**NEU**	**NNEU**	**SAD**	**SUR**
Best model	75.8	53.1	53.1	48.3	40.0	54.6	40.0	67.5	86.8	31.0	55.0	50.0
$n = 1$	61.5	31.6	27.8	20.9	**21.1**	0	0	45.0	76.2	15.3	36.4	28.3
$n = 3$	67.5	34.4	**31.4**	**24.3**	19.1	**16.7**	0	45.0	81.0	11.6	**40.0**	37.8
$n = 5$	**71.5**	**38.0**	30.9	23.4	18.2	0	0	**50.5**	**83.4**	**17.2**	33.9	**43.8**

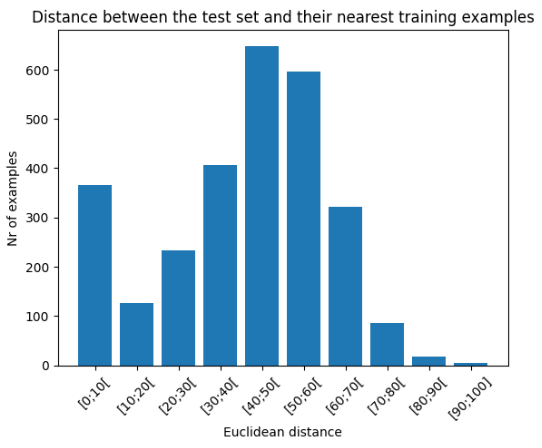
Table 6.9: Retrieval augmentation results using top-n retrieval method, with $n = \{1, 3, 5\}$.

In Table 6.9, we can observe the results of the retrieval using the different methods. We can start by observing that the majority voting methods are superior to the $n = 1$ method in all metrics reported. If we only consider the majority voting methods, the *m* metrics are superior in the $n = 5$, but the *M* metrics are superior in the $n = 3$. Furthermore, the $n = 3$ method is able to classify examples from more classes than the $n = 5$ method. For that matter, we will use the $n = 3$ majority voting as our retrieval method.

In Table 6.9 we can also observe the results of our best model thus far (RoBERTa large + *concat4* + *context1*). All reported metrics have a lower performance using only retrieval, with the *neutral*-F1 having similar results, but in the remaining classes having a lower performance. This is particularly noticeable in the *M/m-NMC* metrics, where the results are substantially lower. Given that the goal of the retrieval is to aid the classification model with bias towards the correct label, this could be an indication that using this approach might not work well towards our goal. Furthermore, in Figure 6.3 we can observe the



(a) Validation set.



(b) Test set.

Figure 6.3: Distribution of the euclidean distances between the closest retrieved training example for each example in the validation and test sets.

distribution of the euclidean distances between the closest retrieved training example (with $n = 1$) and the examples on the development and test sets. Intuitively, closer examples would be more relevant to find correct labels. In both sets we can observe how most examples are not part of the first bins, which could also indicate uncertainty in finding the correct retrieved label for the examples. Regardless, we are not able to conclude this only from these results. For that matter, we will next incorporate the information of the retrieval using the $n = 3$ retrieval majority voting.

Results

In this section we will analyze the results of the methods previously described. The models we are going to experiment with are the following:

- **sum + init**: *sum* method with the sentiment embedding matrix initialized.
- **concat + no init**: *concat* method with the sentiment embedding matrix not initialized. In this method, the embeddings in the sentiment embedding matrix can have size $= \{64, 128\}$.
- **concat + init**: *concat* method with the sentiment embedding matrix initialized. As previously mentioned the sentiment embedding matrix is initialized with the average of the sentence embeddings of each sentiment. The RoBERTa large model embeddings' size is 1024. In this method we will have two options: either concatenate the embedding of size 1024, or resize it to a smaller dimension of size 128 using a linear layer, before concatenating.

We can see the results obtained in Table 6.10. First, we can observe that the *sum* method performs very poorly, and classifies all sentences as neutral. For that matter, we did not do anymore experiments using this method and focused solely on the concat method.

Regarding the remaining models using the concat method, the ones that performed best based on the *micro* scores are the *no init (128)* and the *init (1024)*. The *micro-F1* metrics are similar between them, with the *no init (128)* performing 0.8 points above on the *micro-F1* and 0.2 on the *m-NMC* metric.

	F1				F1							
Ret Aug.	**m**	**m-NMC**	**M**	**M-NMC**	**ANG**	**DIS**	**FEA**	**JOY**	**NEU**	**NNEU**	**SAD**	**SUR**
None	75.8	53.1	**53.1**	**48.3**	40.0	54.6	**40.0**	**67.5**	86.8	31.0	55.0	50.0
sum + init (128)	68.6	0	10.2	0	0	0	0	0	81.4	0	0	0
concat + no init (64)	76.9	52.8	52.0	47.0	38.1	54.6	33.3	66.7	87.4	27.9	54.6	53.8
concat + no init (128)	**77.8**	**53.8**	48.6	43.0	40.0	44.5	0	65.9	87.6	**32.1**	**57.9**	**60.9**
concat + init (1024)	77.0	53.6	52.8	47.9	38.1	**66.7**	25.9	66.1	87.2	28.9	54.1	56.4
concat + init (128)	76.8	51.8	48.6	43.0	**52.2**	44.5	0	65.2	**87.7**	31.8	52.5	55.0

Table 6.10: Retrieval augmentation results using three different methods: sum + init, concat + no init, and concat + init, with different resizings of the sentiment embeddings (64, 128 and 1024).

If we observe the *macro-F1* metrics, the *init (1024)* achieves a score of 52.8 on the *macro-F1* (4.2 points above the *no init (128)*) and 47.9 on the *M-NMC* (4.9 points above the *no init (128)*), mainly due to the fact that this model has a non-zero score on the *fear*-F1 metric. Furthermore, the *init (1024)* model improves the *micro-F1* metrics, when compared to the version with no retrieval augmentation, scoring 1.2 points above on the *micro-F1* and 0.5 on the *m-NMC* metric. Regarding the *macro* metrics, both models with and without retrieval augmentation are very similar. For that matter, we will choose the *concat + init (1024)* model as the best model.

6.2.7 EmotionPush Results

Having developed our approach to the sentiment classification problem, we will start by comparing the performance of the final model (SA Model) with the baseline, for both development and test sets. The results can be seen in Table 6.11. It can be observed that our model improves all major metrics, except the *micro-F1* on the test set where it maintains the same value. More notably, it is able to improve the *macro-F1* metric by 5.3 points on the development set, and 8.7 points on the test set. These improvements are also noticeable on the *M-NMC* metric, where our model improves 6.1 points on the development set and 10 points on the test set. On both sets, our model is also able to classify examples belonging to all classes, unlike the baseline.

		F1				F1							
		m	**m-NMC**	**M**	**M-NMC**	**ANG**	**DIS**	**FEA**	**JOY**	**NEU**	**NNEU**	**SAD**	**SUR**
Dev	Baseline	76.8	51.9	47.5	41.8	28.6	25.0	**33.3**	**66.5**	**87.2**	**29.6**	29.6	55.7
	SA Model	**77.0**	**53.6**	**52.8**	**47.9**	**38.1**	**66.7**	25.9	66.1	**87.2**	28.9	**54.1**	**56.4**
Test	Baseline	**78.9**	57.6	45.4	39.2	36.0	21.1	0	**73.1**	**88.5**	**30.1**	**58.7**	55.5
	SA Model	**78.9**	**58.2**	**54.1**	**49.2**	**45.6**	**27.3**	**66.7**	72.5	88.4	21.3	55.6	**55.8**

Table 6.11: Comparison between the baseline and our best sentiment analysis model on the development and test sets.

In order to further validate our results we will perform an ablation study on the best model (RoBERTa-large + *concat4* pooling + one sentence as context + linear classification layer + retrieval augmentation initialized with no resizing of the embeddings) using the test set. We will remove/substitute each method one at a time with the ones used in the baseline. We performed four experiments in the ablation study. They are defined as follows:

- **+ RoBERTa-base**: We replace RoBERTa-large by RoBERTa-base;

- **+ CLS**: we replace the *concat4* pooling by the CLS pooling;
- **- Context 1**: we no longer use context as input to our model;
- **- Ret. Aug.**: we remove the retrieval augmentation from the model.

	F1				F1							
	m	**m-NMC**	**M**	**M-NMC**	**ANG**	**DIS**	**FEA**	**JOY**	**NEU**	**NNEU**	**SAD**	**SUR**
SA Model	**78.9**	58.2	54.1	49.2	45.6	27.3	**66.7**	72.5	**88.4**	21.3	55.6	**55.8**
+ RoBERTa base	-0.7	-0.8	-4.2	-4.7	-4.9	-16.8	-16.7	-0.3	-0.4	+6.4	+2.2	-3.5
+ CLS	-0.9	-0.9	-1.6	-1.7	-6.3	+4.3	-16.7	+0.8	-0.3	+6.7	-1.5	**0**
- Context 1	-2	-2.6	-8.8	-9.9	-16.4	+9.1	-66.7	-1.6	-1.3	**+8.2**	+0.7	-2.7
- Ret. Aug.	-0.7	**+0.3**	**+3.1**	**+3.6**	**+6**	**+12.7**	**0**	**+1.4**	-0.6	+5.6	**+3.4**	-4.2

Table 6.12: Ablation study on the test set.

The results obtained on the ablation study can be observed in Table 6.12. Removing the context sentences from the input is what impacts the model the most, which tells us it was the most significant addition to our model. Furthermore, replacing the RoBERTa-large by the RoBERTa-base and the *concat4* by the CLS pooling option, also worsens all major metrics. Interestingly, using the retrieval augmentation methods worsened our results on the test set. Another interesting remark is that, in the model with no retrieval augmentation, the *fear*-F1 metric is the only one that remained the same when compared to the SA Model. If we look at Table 6.9, where we evaluated the quality of the bias passed to the model, this was the only label that was not captured by retrieval in the development set. We also computed the bias results for the test set, and the *fear* label is also not captured, which might help explain this observation.

6.2.8 DailyDialog Results

In this section we will evaluate how the setup behaves for the DailyDialog corpus. To do so, we will compare the baseline (RoBERTa-large) with the best setup found on the EmotionPush corpus (RoBERTa-large + *concat4* pooling + one sentence as context + linear classification layer + retrieval augmentation initialized with no resizing of the embeddings). Since on the ablation study done on the EmotionPush corpus (Section 6.2.7) we found that removing context and retrieval augmentation had the most impact on the model, we will only focus on those changes.

A summary of the results obtained on the development set can be seen in Table 6.13. First, we can start by noting that our model does not improve the baseline on most major metrics, except the *micro-F1* metric that improves 0.3 points. Secondly, we can observe that removing the context or retrieval augmentation, has a minimal impact on the metrics.

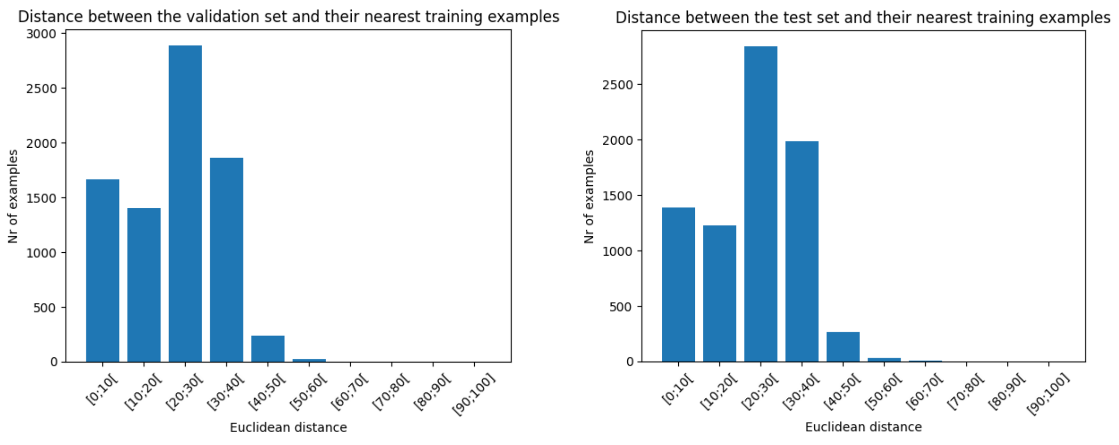
In Table 6.14, we have the results on the test set. Contrarily to the results obtained on the development set, on the test set, all major metrics results' improve when compared to the baseline. The *micro-F1* metric improves by 0.5 points, and the *m-NMC* by 1.6 points. Regarding the *macro-F1* metric, it improves by 2.9 points and the *M-NMC* by 3.4 points. Interestingly, on the test set both removing the context and retrieval augmentation worsen the results, which tells us that both methods are helping in the classification.

	F1				F1						
Model	**m**	**m-NMC**	**M**	**M-NMC**	**ANG**	**DIS**	**FEA**	**HAP**	**NO-EMO**	**SAD**	**SUR**
Baseline	89.3	59.4	47.5	39.8	53.6	0	28.6	63.2	94.1	40.5	52.9
SA Model	**89.6**	59.3	47.4	39.5	51.8	0	**30.8**	63.5	**94.3**	42.1	49.1
- Context 1	-0.3	0	0	+0.1	-3.7	0	-2.2	+0.2	-0.2	**+2.0**	**+4.3**
- Ret. Aug.	**0**	**+0.2**	**+0.3**	**+0.4**	**+3.0**	0	**0**	**+0.8**	**0**	-0.4	-1.1

Table 6.13: Results obtained on the DailyDialog development set. The results on the ablation study are a comparison with the results obtained for the SA Model.

	F1				F1						
Model	**m**	**m-NMC**	**M**	**M-NMC**	**ANG**	**DIS**	**FEA**	**HAP**	**NO-EMO**	**SAD**	**SUR**
Baseline	84.5	56.5	48.1	41.0	**41.6**	18.2	33.3	61.7	91.0	38.1	53.0
SA Model	**85.0**	**58.1**	**51.0**	**44.4**	41.5	31.3	34.3	**62.8**	**91.1**	41.0	**55.2**
- Context 1	-0.6	-1.6	-2.5	-3.0	-2.0	-11.0	**+2.7**	-0.3	-0.2	-4.8	-2.2
- Ret. Aug.	-0.6	-1.3	-1.0	-1.2	-5.3	**+3.5**	-4.7	-0.9	-0.3	**+1.2**	-0.6

Table 6.14: Results obtained on the DailyDialog test set. The results on the ablation study are a comparison with the results obtained for the SA Model.



(a) Development set.

(b) Test set.

Figure 6.4: Distribution of the euclidean distances between the closest retrieved training example for each example in the development and test sets.

In Section 6.2.6 we mentioned how our intuition was that in retrieval augmentation methods, finding closer examples would be more relevant to finding correct labels. Contrarily to the results obtained on the EmotionPush corpus, on the DailyDialog the retrieval augmentation methods seemed to have a more positive impact on the results. In Figure 6.4 we have the distribution of the euclidean distances between the closest retrieved training example and the examples in the development and test sets from the DailyDialog corpus. When compared to the same analysis done on the EmotionPush corpus, in Figure 6.3, we can observe how in the DailyDialog more examples are part of the first bins, which indicates a higher certainty when retrieving examples.

6.2.9 Results Comparison

In this section we will compare our results with other published works. Regarding EmotionPush, we will use the most recent result we have found that used the same train/development/test splits as we did. In particular, we will compare our models' performance with the best model of the SocialNLP 2018 EmotionX Challenge shared task (Hsu et al., 2018). The metric reported for this shared task is the Unweighted Accuracy (UWA), which we calculated for comparison purposes. Additionally, due to the low number of examples for some classes, Hsu et al. (2018) proposes to use only four classes during evaluation: *joy*, *anger*, *sadness*, and *neutral*. Therefore, for this dataset, we also report the score considering this constraint. Regarding the DailyDialog corpus, we compare the micro-F1 without the majority class and the macro-F1 metrics.

In Table 6.15 we can observe our models' performance versus other works found in literature. We can start by observing that our model is able to improve the performance on the EmotionPush corpus by 8.5 points. We hypothesise this is due to the fact that Khosla (2018) makes use of GloVe embeddings, which have a subpar performance when compared to the Transformer-based contextual embeddings we use. On the DailyDialog corpus, our model outperforms all approaches except Ghosal et al. (2020), even though the performance is similar. However, we would like to highlight that Ghosal et al. (2020) makes use of an additional large knowledge base that captures certain aspects such as personality, or emotion interactions. We do not make use of such attributes, which albeit useful, are not readily available for most practical use-cases.

	EmotionPush	**DailyDialog**	
Model	**UWA**	**m-NMC**	**M**
CNN-BiLSTM (Khosla, 2018)	62.2	-	-
KET (Zhong et al., 2019b)	-	53.4	-
ELECTRA with Contextual Augmentation (Kim et al., 2020)	-	57.9	-
COSMIC (Ghosal et al., 2020)	-	**58.5**	**51.1**
SA Model (*ours*)	**70.7**	58.1	51.0

Table 6.15: Comparison of results found in literature against our model.

6.2.10 Conclusion

Throughout this section we have explored the problem of sentiment classification applied to two datasets, EmotionPush and DailyDialog, in a dialogue context. In particular, we resorted to an extensive experimentation cycle to show the impact, not only of different model/architecture choices, but also the addition of dialogue-specific inputs, such as the context of a sentence, and the usage of labeled examples to improve performance. In particular, we have shown the impact of using context, which highlights the importance of considering the previous context when classifying the sentiment of a sentence within a dialogue. With regard to the retrieval-based approaches, the results were not as conclusive, which we attribute to the different degrees of similarity between development/test splits and the corresponding training splits (used as support data) of different datasets.

Considering the results obtained, for our final system we will be using the full model (RoBERTa-large + *concat4* pooling + one sentence as context + linear classification layer + retrieval augmentation initialized and with no resizing of the embeddings) on the DailyDialog corpus, and the same model but without retrieval augmentation on the EmotionPush corpus, given that we showed that retrieval worsened our test results.

6.3 Reply Sentiment Prediction

In this section we will develop the reply sentiment prediction model. As defined in Section 4.4, the default input of the reply sentiment prediction model is to use two sentences, one from the conversational agent, and one from the user. The goal is to predict the label of the upcoming sentence. Given that the BERT and RoBERTa model proved to be strong baselines for the task of sentiment classification, we will use them as a starting point for reply sentiment prediction model as well. Furthermore, we will make use the `[CLS]` token of the last hidden layer as the sentiment embedding representation, and a linear layer as the classification model for the baseline.

6.3.1 Metrics

Given that reply sentiment prediction is a classification task and our goal is to evaluate the number of examples being classified correctly, we will be using the same metrics used in the sentiment classification evaluation (Section 6.2), which are the following: micro-F1 (*m*); micro-F1 with no majority class (*m-NMC*); macro-F1 (*M*); macro-F1 with no majority class (*M-NMC*); and sentiment-F1 metrics.

6.3.2 Training Details

The training of the reply sentiment prediction models follows the same details described in Section 6.2.2.

6.3.3 Baseline

The first step is to choose a baseline model. Given that the BERT and RoBERTa models performed well for the task of sentiment classification, we will perform baseline experiments with the same models, in two different sizes, base and large. A description of the models can be found in Section 6.2.2.

	F1				F1							
Model	**m**	**m-NMC**	**M**	**M-NMC**	**ANG**	**DIS**	**FEA**	**JOY**	**NEU**	**NNEU**	**SAD**	**SUR**
BERT-base-cased	67.8	14.9	15.0	5.5	0	0	0	25.1	81.4	3.2	**10.0**	0
BERT-large-cased	66.5	17.4	**15.4**	**6.1**	0	0	0	27.9	80.8	**6.1**	8.5	0
RoBERTa-base	68.5	13.3	13.6	3.8	0	0	0	25.2	**81.8**	1.6	0	0
RoBERTa-large	**69.0**	**18.0**	15.0	5.5	0	0	0	**31.8**	81.7	1.8	5.1	0

Table 6.16: Results obtained on four different architectures (BERT-base-cased, BERT-large-cased, RoBERTa-base, RoBERTa-large) on the development set of the EmotionPush dataset.

The results obtained are in Table 6.16. We can start by observing how this task is harder for the baseline models to perform well in. In particular, for the sentiments *anger*, *disgust*, *fear*, and *surprise*, none of the models is able to classify any example. Nonetheless, the RoBERTa-large model is the model that performs better in the *micro* scores, and has similar results to the best performing model in the *macro* scores (BERT-large-cased). So, we will choose the RoBERTa-large as our baseline model.

6.3.4 Sentence Embedding Representation

We will explore the same sentence embedding representations we used for the sentiment classification model. The methods are described in Section 6.2.4 and are: *cls*; *cls+avg*; *biLSTM*; *concat2*; *concat4*; and *cls avg*.

	F1				**F1**							
Pooling	**m**	**m-NMC**	**M**	**M-NMC**	**ANG**	**DIS**	**FEA**	**JOY**	**NEU**	**NNEU**	**SAD**	**SUR**
cls	**69.0**	18.0	15.0	5.5	0	0	0	31.8	**81.7**	1.8	5.1	0
avg	67.1	17.6	15.2	5.9	0	0	0	29.2	80.6	7.6	4.6	0
cls+avg	63.8	18.2	16.1	7.1	0	0	0	27.3	78.9	**15.2**	9.5	0
biLSTM	67.1	18.7	15.1	5.7	0	0	0	28.9	81.1	11.1	0	0
concat2	65.6	18.3	**16.3**	**7.2**	0	0	0	28.6	80.1	12.1	9.5	0
concat4	67.6	**19.7**	15.9	6.6	0	0	0	**33.1**	80.9	3.1	**10.3**	0
cls avg	68.8	0	10.2	0	0	0	0	0	81.5	0	0	0

Table 6.17: Results obtained with the RoBERTa large model using seven different pooling techniques.

The results can be seen in Table 6.17. The *cls* representation achieves the best result on the *micro-F1* metric, but the *concat4* performs 1.7 points above the *cls* on the *m-NMC* metric. Furthermore, when compared with the *cls* method, the *concat4* also improves the *macro* metrics, scoring 0.9 points above on the *M* metric and 1.1 points on the *M-NMC* metric. Therefore, we will use the *concat4* sentence embedding representation for our model.

6.3.5 Contextual Reply Sentiment Prediction

As mentioned in Section 4.4.2, the default input for the task of reply sentiment prediction is having two sentences as context, one from the conversational agent and one from the user, in order to predict the sentiment of the upcoming sentence to be generated by the conversational agent. In addition to this, we experimented with using between one and four turns as context to see if the model benefited from it.

	F1				**F1**							
Context	**m**	**m-NMC**	**M**	**M-NMC**	**ANG**	**DIS**	**FEA**	**JOY**	**NEU**	**NNEU**	**SAD**	**SUR**
$x = 1$	67.5	12.8	14.5	5.1	0	0	0	21.3	80.8	3.3	5.3	**5.4**
$x = 2$	**67.6**	**19.7**	15.9	6.6	0	0	0	**33.1**	**80.9**	3.1	10.3	0
$x = 3$	67.2	18.9	16.2	7.0	0	0	0	30.0	80.6	**8.8**	5.0	5.0
$x = 4$	66.5	19.6	**17.8**	**8.8**	0	0	0	29.2	80.7	8.5	**18.9**	4.8

Table 6.18: Results obtained for the reply sentiment prediction task using between one and four sentences as context.

The results of the experiments can be observed in Table 6.18. Firstly, for 1, 3 and 4 turns of context we are able to classify examples of a label we were not previously able to (*surprise*), so we will focus on these models. Out of these three setups, the one that performs best on the *micro*-F1 score is the $x = 1$, but its *m-NMC* is the lowest out of the three. The $x = 4$ setup achieved the highest *macro-F1* scores, 1.9 points above the default setup ($x = 2$) on the *M* score and 2.2 points above on the *M-NMC* score. Furthermore, it performs only 0.1 points lower on the *m-NMC* metric than the standard $x = 2$ and 6.8 above than the $x = 1$ model. Therefore, we will use four context sentences as input for the model.

Adding Sentiment Context

In addition to adding extra context sentences to the input of the model, as described in Section 4.4.2, we can also add sentiment representations of the emotions expressed in the context sentences. To do so, we will use an approach similar to the *concat* method described in Section 4.3.2.

The sentiment embeddings (SE) created for this method have a dimension of 64 and are initialized at random following a normal distribution, $\mathcal{N}(0, 1)$. Given that in the previous section we saw that we achieved the best results using four input sentences, we concatenate four sentiment vectors to the sentence representations, each corresponding to the labels of the input sentences.

	F1				F1							
Context	**m**	**m-NMC**	**M**	**M-NMC**	**ANG**	**DIS**	**FEA**	**JOY**	**NEU**	**NNEU**	**SAD**	**SUR**
$x = 4$	66.5	**19.6**	**17.8**	**8.8**	0	0	0	**29.2**	80.7	**8.5**	**18.9**	**4.8**
$x = 4$ + *SE*	**68.5**	15.0	15.8	6.4	0	0	0	24.9	**81.5**	6.0	14.0	0

Table 6.19: Results obtained for the response sentiment prediction task with and without adding the sentiment embeddings to the input of the model.

The results can be observed in Table 6.19. Using context sentiment embeddings improves the *micro-F1* score, but all other major metrics lower significantly. In particular, after an informal inspection to the data, we have observed that most sentences are preceded or followed by sentences labeled as *neutral*. Given that observation, our hypothesis is that for this particular dataset it is not possible to obtain useful information about the next sentence label based solely on the labels of the previous sentences. The obtained results are interesting, given that the model with the sentiment embeddings actually improved the *neutral*-F1 by 0.8 points and lowered all other *sentiment*-F1 metrics. This could indicate that in a better balanced corpus with more significant sequences this approach could work. Nevertheless, given the poor results on this dataset, we will not be using the labels of the context sentences.

An important aspect to notice about this decision is that, since we will not be using the labels of the context sentences, this means that the proposed sentiment-aware conversational agent will not include the sentiment classification as a first step. Thus, the sentiment classification model will only be used as a metric to evaluate the text generation.

6.3.6 Retrieval Augmentation

Given the seemingly hard task to classify the appropriate sentiment of an upcoming sentence, we will be using the same retrieval augmentation techniques described in Section 4.3 to give some more context to the model in order to improve performance.

Retrieved Labels Evaluation

In the task of sentiment classification the retrieval was done by finding the most similar training example to the one we were going to classify. In this task, our inputs are composed of previous sentences. For that matter, in order to retrieve the nearest neighbors information, for each example we retrieved the most similar training context and used the label of the upcoming sentence as the retrieved label. For the retrieval we used two sentences, for computational efficiency reasons.

	F1				**F1**							
Retrieval	**m**	**m-NMC**	**M**	**M-NMC**	**ANG**	**DIS**	**FEA**	**JOY**	**NEU**	**NNEU**	**SAD**	**SUR**
Best model	66.5	19.6	17.8	8.8	0	0	0	29.2	80.7	8.5	18.9	4.8
$n = 1$	51.3	**10.5**	**12.9**	**5.0**	0	0	0	14.0	69.2	**13.1**	**2.5**	**4.7**
$n = 3$	57.9	9.5	11.8	2.9	0	0	0	**15.6**	74.4	4.5	0	0
$n = 5$	**63.4**	4.4	10.9	1.2	0	0	0	8.6	**78.4**	0	0	0

Table 6.20: Retrieval augmentation signal results using top-n retrieval method, with $n = \{1, 3, 5\}$.

Again, we will start by considering the quality of the retrieved labels using the top-n method described in Section 6.2.6. The results of using only the retrieval methods can be observed in Table 6.20. The retrieval method that achieves the best *micro*-F1 is the $n = 5$, but it is only able to retrieve examples of the *joy* and *neutral* sentiments. When comparing the $n = 1$ and $n = 5$ retrieval methods, we can observe that the $n = 1$ approach achieves the best overall results, but still lower than the results obtained by the best model so far. Similarly to what we observed when using this method for sentiment classification in Section 6.2.6, this can be an indication that adding the retrieval augmentation information will not yield better results, given that the quality of the retrieval is much lower than the results achieved by the model.

Results

In this section we will show the results of the experiments done by adding the retrieval information to the model. We will use the same methods described in Section 6.2.6: *sum + init*, *concat + no init*, and *concat + init*.

	F1				**F1**							
Ret Aug.	**m**	**m-NMC**	**M**	**M-NMC**	**ANG**	**DIS**	**FEA**	**JOY**	**NEU**	**NNEU**	**SAD**	**SUR**
None	66.5	**19.6**	**17.8**	**8.8**	0	0	0	**29.2**	80.7	**8.5**	**18.9**	**4.8**
sum + init (128)	67.0	0	10.0	0	0	0	0	0	80.3	0	0	0
concat + no init (64)	68.5	14.5	14.3	4.7	0	0	0	26.3	**81.8**	1.7	5.0	0
concat + no init (128)	67.2	15.6	15.2	5.9	0	0	0	27.3	80.6	4.4	9.3	0
concat + init (1024)	**68.6**	17.1	15.8	6.3	0	0	0	28.8	81.7	6.2	9.3	0
concat + init (128)	68.1	13.0	13.8	4.1	0	0	0	24.0	81.3	0	5.0	0

Table 6.21: Retrieval augmentation results using three different methods: sum + init, concat + no init, and concat + init, with different resizings of the sentiment embeddings (64, 128 and 1024).

The results obtained can be observed in Table 6.21. Contrarily to the results we obtained in the development set for sentiment classification, for the reply sentiment prediction, using retrieval augmentation significantly lowers the performance of most metrics. Out of the models that use retrieval, the setup with the better performance is the *concat + init (1024)*, which actually achieves a higher *micro*-F1 and *neutral*-F1 than the model with no retrieval augmentation, but has a lower performance in all other metrics. For this reason, we will not be using retrieval augmentation.

6.3.7 EmotionPush Results

Now that we have a final reply sentiment prediction (RSP) model setup (RoBERTa-large + *concat4* pooling + four sentences as context + linear classification layer), in Table 6.22 we can observe the comparison of our model versus the baseline in both the development and test sets. In the development set we can see how the *micro-F1* does not improve when compared to the baseline. This can be explained by our baseline model having a higher bias towards the *joy* and the *neutral* sentiments, as can be observed by their higher F1 values. Nonetheless, our model improves all other metrics (*m-NMC* by 1.6 points, *M* by 2.8 points, and *M-NMC* by 3.3 points), which shows how our model is better at generalizing for less represented sentiments. Regarding the performance on the test set, the conclusions are similar. The *micro-F1* is higher by 1.8 points on the baseline, which can also be explained by the higher *neutral-F1*. Despite this, our model also outperforms the baseline in all other metrics (*m-NMC* by 6.7 points, *M* by 2.4 points, and *M-NMC* by 2.9 points), which validates the improvements when compared to the baseline.

		F1				**F1**							
		m	**m-NMC**	**M**	**M-NMC**	**ANG**	**DIS**	**FEA**	**JOY**	**NEU**	**NNEU**	**SAD**	**SUR**
Dev	Baseline	**69.0**	18.0	15.0	5.5	0	0	0	**31.8**	**81.7**	1.8	5.1	0
	RSP Model	66.5	**19.6**	**17.8**	**8.8**	0	0	0	29.2	80.7	**8.5**	**18.9**	**4.8**
Test	Baseline	**66.0**	14.4	13.3	3.8	0	0	0	24.3	**79.6**	0	2.3	0
	RSP Model	64.2	**21.1**	**15.7**	**6.7**	0	0	0	**32.0**	78.8	**8.6**	**2.1**	**4.1**

Table 6.22: Comparison between the baseline and our best sentiment analysis model on the development and test sets.

In order to further validate our results, we will perform an ablation study on the test set using a similar method to the one defined in Section 6.2.7. The ablation study experiments are defined as follows:

- **+ RoBERTa-base**: we replace RoBERTa-large by RoBERTa-base;
- **+ CLS**: we replace the *concat4* sentence representation by the CLS;
- **+ Context 2**: we replace the four context sentences by the default two context sentences;

Given that for the sentiment classification task the retrieval augmentation showed different behaviors in the development and test sets, we will also be showing the results of the best performing retrieval augmentation setup (*concat + init (1024)*) on the test set (*+ Ret. Aug.*).

In Table 6.23, we can observe the ablation study done on the test set. We can start by observing how all models have a higher *micro*-F1 than the RSP model. Again, this can be explained by the higher *neutral*-F1 values. On the remaining major metrics, all models perform worse than the RSP model. These results show how our model is doing a trade-off between a lower F1 in the *neutral* label and a higher F1 on the less represented sentiments. Another important aspect is the performance of the

	F1				F1							
	m	m-NMC	M	M-NMC	ANG	DIS	FEA	JOY	NEU	NNEU	SAD	SUR
RSP Model	64.2	**21.1**	15.7	**6.7**	0	0	0	**32.0**	78.8	**8.6**	2.1	**4.1**
+ RoBERTa base	**+3.4**	-6.8	**+2.5**	-3.1	0	0	0	-7	**+1.7**	-8.6	-2.1	-4.1
+ CLS	+2.6	-4.9	-1.6	-2.1	0	0	0	-6.3	+1.4	-4.2	+0.2	-4.1
+ Context 2	+0.2	-3.4	-1.3	-1.5	0	0	0	-4.5	0	-4.1	+2.3	-4.1
+ Ret. Aug.	+0.6	-3.2	-0.3	-0.3	0	0	0	-5.3	+0.3	-0.8	**+7.8**	-4.1

Table 6.23: Ablation study on the test set.

model with retrieval augmentation. Similarly to the evaluation done on the development set, adding the retrieval augmentation methods is not helping our model achieve better results.

6.3.8 DailyDialog Results

In this section we will evaluate how our reply sentiment prediction setup behaves for the DailyDialog corpus. In particular, we will compare the baseline (RoBERTa-large) with the best setup found on the EmotionPush corpus (RoBERTa-large + concat4 sentence embedding representation + four context sentences).

The ablation study performed on the EmotionPush showed that all introduced changes were impacting the final model. For that matter, in addition to showing the baseline and the best model results, we will also perform the same ablation study done in the previous section to the DailyDialog corpus, for both development and test sets.

	F1				F1						
Model	m	m-NMC	M	M-NMC	ANG	DIS	FEA	HAP	NO-EMO	SAD	SUR
Baseline	86.6	**40.9**	28.0	17.2	**34.1**	0	0	**45.6**	92.6	5.6	**18.2**
RSP Model	85.4	39.9	27.6	16.9	30.1	0	0	44.7	91.9	**13.0**	13.5
+ RoBERTa base	-0.6	-3	-3.7	-4.3	-12.4	0	0	-2.8	-0.3	-7.2	-3.2
+ CLS	+0.8	-0.4	-1.4	-1.7	-3.6	0	0	0	-0.5	-7.4	+0.7
+ Context 2	-0.2	-2.5	-0.9	-1.1	+2.5	0	0	-2.8	0	-5.9	-0.3
+ Ret. Aug.	**+2**	-5.9	**+1.2**	**+1.5**	-10.1	0	**+36.4**	-5.5	**+1.3**	-13.0	+1.4

Table 6.24: Comparison between the Baseline and RSP Model results obtained, and ablation study performed on the DailyDialog development set. The results on the ablation study are a comparison with the results obtained for the RSP Model.

In Table 6.24 we can observe the results obtained on the development set for both the baseline and our model (RSP Model), as well as the ablation study. Firstly, we can observe that even in a corpus with a larger number of training examples, the reply sentiment prediction task is still a hard task to perform well in. Neither the baseline nor the RSP Model were able to classify examples belonging to two sentiments. Comparing the baseline with the RSP Model, we can observe that this setup did not improve the base model on most metrics. Furthermore, if we consider the ablation study, we can see that none of the changes had a very strong impact on the performance either. Interestingly, for this corpus, the retrieval augmentation improved some of the metrics. As we saw in Sections 6.2.6 and 6.2.8, the distance between examples on the DailyDialog corpus tends to be lower than on the EmotionPush, which might explain why the retrieval augmentation improves the results.

	F1				F1						
Model	**m**	**m-NMC**	**M**	**M-NMC**	**ANG**	**DIS**	**FEA**	**HAP**	**NO-EMO**	**SAD**	**SUR**
Baseline	80.7	40.1	33.8	24.6	30.1	9.3	25.0	45.0	88.8	13.8	24.7
RSP Model	80.4	**42.8**	35.0	26.1	**34.6**	8.9	33.3	**48.5**	88.6	10.6	20.5
+ RoBERTa base	-1.2	-1.5	-2.9	-3.2	-16	+0.4	**+2**	-1.6	-0.8	-5.1	+1.1
+ CLS	+0.7	-1.3	-0.4	-0.6	-5.4	+0.4	-2	-1.3	+0.5	+0.1	+1.1
+ Context 2	-0.2	-1.7	**+0.5**	**+0.5**	-5.5	+8.1	-9.8	-2.6	-0.2	**+7.6**	**+5.9**
+ Ret. Aug.	**+1.6**	-6.1	-1.9	-2.4	-8.7	**+16.6**	-11.1	-6.9	**+1.2**	-8.5	+4.2

Table 6.25: Comparison between the Baseline and RSP Model results obtained, and ablation study performed on the DailyDialog test set. The results on the ablation study are a comparison with the results obtained for the RSP Model.

The results obtained on the test set, which can be observed in Table 6.25, allow for a different set of conclusions. First of all, our introduced changes improve the baseline. Secondly, on the test set, both the baseline and the RSP Model are able to classify examples belonging to all classes. Finally, contrarily to the evaluation performed on the development set, the retrieval augmentation does not improve the results on most metrics.

In summary, the results obtained for this dataset are inconclusive and lead us to believe that this corpus required a study of its own to understand which increments are more favorable, which for time and computational resources constraints we were unable to do. Nonetheless, we were able to observe that the task of reply sentiment prediction is hard to perform well in even when trained on a larger corpus.

6.3.9 Conclusion

During this section we explored the task of reply sentiment prediction applied to two datasets, EmotionPush and DailyDialog. We started this experimental cycle by observing that despite being a classification task, the baseline Transformer-based approaches (Devlin et al., 2019; Liu et al., 2019) do not perform well, which highlights the need for models pre-trained in a different way or for better balanced and more appropriate data. Furthermore, we saw that we were unable to make a connection between the sequences of sentiments that precede a given sentiment. Additionally, we saw that on this task, using retrieval augmentation did not provide conclusive results.

Considering the results obtained, we will use the *RoBERTa-large + concat4 sentence embedding representation + four context sentences* setup for both models in the final system. It is important to notice that this setup does not use previous labels' knowledge to make predictions. Thus, the proposed conversational agent will only include the reply sentiment prediction and the text generation models.

Despite the low scores for reply sentiment prediction, we are still missing the text generation experiments, which we will present next. We will see that if we succeed in generating text with sentiment, then we prove the need for a reply sentiment prediction model, which is key to guide the dialogue conditioned text generation model.

6.4 Conditioned Text Generation with Sentiment

Having developed both the sentiment classification and reply sentiment prediction models, what is left to do is to develop a sentiment-conditioned text generation model. To do so, as described in Chapter 5, we will adapt the model proposed by Wolf et al. (2019), a dialogue conversational agent that is able to express a given persona, to our specific use-case. We will start by fine-tuning different pre-trained text generation models in order to build a baseline. Then, we will explore different sets of sentiment lexicons to be concatenated to the input of the model, as a way of conditioning generation with the desired sentiment. Finally, we will explore the task of next sentence prediction (Devlin et al., 2019), by using sentiment distractors during the fine-tuning of the model. Similarly to the previously described tasks (Sections 6.2 and 6.3), we will perform these experiments on the EmotionPush corpus, and later apply our findings to the DailyDialog dataset.

6.4.1 Metrics

The goal of our work is to develop a sentiment-aware conversational agent without compromising the quality of the generated text. Therefore, our choice of automatic evaluation metrics aims to measure these two aspects. First, regarding the quality of the generated text, we will focus on two metrics:

- **Perplexity (PPL)**: described in Section 2.5.2, it is a metric used to compare language models. The model with the lowest **PPL** has a higher probability of correctly generating an unseen example from a test set.
- **Sentence Embedding Similarity (SES)**: calculates the cosine similarity between the embeddings of the generated and the gold examples. Similarly to Xie and Pu (2021), we use the Sentence Transformer (Reimers and Gurevych, 2019) library to create sentence representations of both sentences, using the `paraphrase-distilroberta-base-v1` model. Then, we calculate the cosine similarity between the two representations. The **SES** is the average of the cosine similarities obtained for all examples in the test set.

Second, to evaluate if the generated text is expressing the appropriate sentiment, we will use the sentiment classification model described in Section 4 and developed in Section 6.2. In particular, we will use the best setup we found for each of the datasets. This model will classify the generated sentences and evaluate them using the micro-F1 (*m*), micro-F1 with no majority class (*m-NMC*), macro-F1 (*M*), and macro-F1 with no majority class (*M-NMC*) metrics, described in Section 6.2.1.

It is important to mention that commonly used automatic metrics such as BLEU (Papineni et al., 2002) or METEOR (Banerjee and Lavie, 2005), can be used for dialogue. However, as mentioned in Liu et al. (2016), such metrics rely on the word overlap between the gold and the generated sentence, which is not ideal in a dialogue scenario given the multitude of possible correct answers for a given context. In particular, given that we will be conditioning the generated text with sentiment, this might lead to even more diversity in the model's replies. For the aforementioned reasons, we will not be using these metrics in the evaluation.

6.4.2 Training and Decoding Details

The work described in Chapter 5 was implemented using PyTorch Lightning (Falcon and The PyTorch Lightning team, 2019) and the HuggingFace Transformers (Wolf et al., 2020) library. Particularly, we modified the code base of the HLT-MAIA lightning-convai repository[2].

The models were trained for a maximum of 40 epochs, using the negative log-likelihood loss, with four validation steps per epoch, stopping the training after 12 consecutive validation steps without improvement. We use the Adam optimizer (Kingma and Ba, 2015) with a learning rate of 5×10^{-6}. The checkpoint used to evaluate the model was the one that achieved the lowest validation negative log-likelihood loss value. Given the dimension of the models, and due to computational constraints, we always use the two most recent context sentences from the dialogue as input to the model. The models were trained on a GeForce GTX 1080 GPU, with a real batch size of 4 whenever the GPU's memory allowed it, but we used gradient accumulation to always simulate a batch size of 16. All the other hyperparameters were kept with their default values.

Decoding was performed using top-p (nucleus) sampling (Holtzman et al., 2019), with $p = 0.9$, and a maximum sequence length of 200.

6.4.3 Baseline

We will start by building a baseline model that will be the baseline for both the text generation model, developed in this section, and the sentiment-aware conversational agent, which will be developed in Section 6.5. We will experiment with two different pre-trained Transformer models, GPT-2 (Radford et al., 2019) and DialoGPT (Zhang et al., 2020). We will experiment with two different configurations from the HuggingFace Transformers (Wolf et al., 2020) library: GPT2; and GPT2-MEDIUM. Regarding the DialoGPT model, it is a model built upon the GPT-2, but trained with conversation-like exchanges from Reddit[3]. We will experiment with two different configurations, also from the HuggingFace Transformers library: DIALOGPT-SMALL; and DIALOGPT-MEDIUM. Further details of the GPT-2 and DialoGPT configurations can be observed in Table 6.26.

Model	**Nr. of Layers**	**Hidden Size**	**Total Parameters**
GPT2	12	768	117M
GPT2-medium	24	1024	345M
DialoGPT-small	12	768	117M
DialoGPT-medium	24	1024	345M

Table 6.26: Description of the GPT-2 and DialoGPT models. The GPT-2 is used in two different configurations: GPT2, and GPT2-medium. The DialoGPT is used in two different configurations, small and medium.

As mentioned in Section 5.1, the baseline architecture we are going to use is appropriate for dialogue. It receives as input the previous context of the conversation, and outputs an answer. The results obtained for the baseline approach can be observed in Table 6.27. First, we can start by observing the high

[2]https://github.com/HLT-MAIA/lightning-convai
[3]https://www.reddit.com

perplexity values, which could be related with the small size of the corpus we are using. As described in 2.5.2, a high *perplexity* value means that the model is not very sure of the text it is generating. The model that achieves the lowest *perplexity* is the *DialoGPT-medium*. This model also outperforms the others in all metrics, which would make it the ideal choice among the evaluated models. Unfortunately, due to computational constraints, we are unable to use the medium sized models when we add sentiment knowledge or the distractors mechanism. For that reason, we will use the next best performing model, the *DialoGPT-small*, as a baseline for our experiments.

			F1			
Models	**PPL**	**SES**	**m**	**m-NMC**	**M**	**M-NMC**
GPT2	107.7	16.6	45.6	16.0	13.4	6.2
GPT2-medium	81.8	16.5	44.3	15.7	13.7	6.2
DialoGPT-small	92.4	16.8	42.5	15.6	13.2	6.4
DialoGPT-medium	**62.2**	**17.3**	**45.9**	**20.5**	**18.1**	**11.5**

Table 6.27: Results obtained on four different architectures (GPT2, GPT2-medium, DialoGPT-small, DialoGPT-medium) on the development set of the EmotionPush corpus.

6.4.4 Guide Model With Sentiment

In order to guide the model towards an appropriate sentiment, as described in 5.2, we will concatenate the desired sentiment's lexicon to the input of the model. During this set of experiments, the provided sentiment for the generation step will be the same as the sentiment of the gold sentence. We will build sentiment lexicon for each sentiment using the following techniques:

- **Tag**: use the sentiment's name. We will use this model as a baseline for the sentiment conditioned text generation model;
- **TF**: retrieve the 40 most frequent n-grams from the set of training sentences of each sentiment. We will assume $1 \leq n \leq 3$;
- **TFU**: the same as the **TF** approach, but removing n-grams that are not unique to each sentiment. We will assume $1 \leq n \leq 3$;
- **TF-IDF**: retrieve 40 n-grams with the highest TF-IDF score. We will assume $1 \leq n \leq 3$;
- **Random Sample**: select at random a sentence from the training set labelled with the sentiment we want to generate;
- **Sentiment Sentences**: use a pre-defined set of sentences representative of each sentiment. The set we are going to use for the EmotionPush corpus can be observed in Table 6.28. This set of short sentences was created using always the same sentence structure ("I am..." and "That is..."). For the DailyDialog corpus the same set is used, except we do not include the *non-neutral* label.

The results of concatenating the different types of sentiment lexicon can be observed in Table 6.29. Regarding the perplexity, we can observe that all approaches, except the *TF-IDF* and the *Sentiment*

Sentiment	**Sentence 1**	**Sentence 2**
Anger	I am angry.	That is so annoying!
Disgust	I am disgusted.	That is repulsive!
Fear	I am frightened.	That is scary!
Joy	I am happy.	That is delightful!
Neutral	I am ok.	That is ok.
Non-neutral	I am not ok.	That is not ok.
Sadness	I am sad.	That is so upsetting.
Surprise	I am surprised.	That is so amazing!

Table 6.28: Sentences used to represent each sentiment in the EmotionPush corpus.

			F1			
Sentiment Lexicon	**PPL**	**SES**	**m**	**m-NMC**	**M**	**M-NMC**
None	92.4	16.8	42.5	15.6	13.2	6.4
Tag	90.1	17.7	45.6	16.8	14.2	7.2
TF	91.2	16.4	45.5	14.4	12.6	5.6
TFU	90.7	16.3	45.0	14.7	13.8	6.7
TF-IDF	120.9	16.4	41.7	13.6	12.6	5.7
Random Sample	94.3	16.8	44.7	20.1	15.0	8.3
Sentiment Sentences	**86.3**	**18.0**	**62.7**	**42.3**	**29.1**	**22.4**

Table 6.29: Results of concatenating sentiment lexicon to the input of the DialoGPT-small text generation model.

Sentences, perform similarly to the baseline (*None*). When compared to the baseline, the *TF-IDF* approach worsens the *perplexity* by 28.5 points, while the *Sentiment Sentences* approach improves this metric by 6.1, which is considerable. Furthermore, the *Sentiment Sentences* approach also improves the *SES* metric by 1.2 points. More interestingly, this approach improves significantly the metrics that are evaluating the expressed sentiment. When compared to the baseline, the *m* metric improves by 20.2 points, the *m-NMC* by 26.7 points, the *M* by 15.9 points, and the *M-NMC* by 16 points. These results show us how this model is generating better sentences, due to the improvements on the *perplexity* and the *sentence embedding similarity*, and expressing the correct sentiment more times, which can be concluded by the improvements on the metrics without the majority class (*m/M-NMC*).

Generation Analysis

Given the major improvements on the metrics, particularly on the sentiment metrics, achieved with the *Sentiment Sentences* approach, we will analyse whether the model is simply copying the sentences provided and using them in the generated reply, and if that is what is causing the improvement on the sentiment metrics. To do so, we will evaluate each generated sentence from the development set and check whether any of the keywords on the set of sentences is present on the generated sentence. We perform stemming on each token, both from the keywords and from the generated sentences, using the NLTK (Bird et al., 2009) implementation of the Porter stemming algorithm (Porter, 1980).

The outcome of this experiment can be observed in Table 6.30. The **Gold** columns show the results if we consider the sentiment of each sentence to be the corresponding gold label. The **Predicted** columns show the results if we consider the predicted labels by the sentiment classification model. We can start

	Gold		Predicted	
Sentiment	**Nr. of Sentences**	**% with keywords**	**Nr. of Sentences**	**% with keywords**
Anger	7	0	2	0
Disgust	5	0	1	0
Fear	4	0.25	0	0
Joy	103	0.03	128	0.02
Neutral	362	0.10	341	0.11
Non-neutral	47	0	30	0
Sadness	28	0.14	54	0.07
Surprise	29	0.03	29	0.03

Table 6.30: Percentage of sentences using the gold and predicted sentiment that include at least one keyword part of the pre-defined set of sentences sentiment lexicon building technique.

by observing that the percentage of sentences with keywords is very low across all sentiments which leads us to believe that the model is not learning to copy the keywords of the provided set of sentiment sentences. Furthermore, in Table 6.31 we gather the number of generated sentences by the *Baseline*, *Tag*, and *Sentiment Sentences* models, that were correctly classified by the sentiment classifier. Since the number sentences that include at least one keyword adds up to 45 (calculated from Table 6.30), and that the *Sentiment Sentences* approach has 366 sentences that were classified correctly, which corresponds to an improvement of 118 sentences regarding the *Baseline* model, and 100 sentences regarding the *Tag* model, we can conclude that the increased sentiment scores shown in Table 6.29 are not exclusively explained by the presence of keywords.

Model	**Nr. of sentences classified correctly**
Baseline	248
Tag	266
Sentiment Sentences	366

Table 6.31: Number of sentences generated by the baseline, Tag and sentiment sentences approaches that were correctly classified by the sentiment classifier.

6.4.5 Next Sentence Prediction

The final experiment performed was to use a double head transformer model and apply the next sentence prediction task through the use of sentiment distractors, as described in Section 5.2.2. Ideally we would use sentences labelled with every other sentiment as distractors. Due to computational constraints this was not possible, therefore, for each gold reply labelled with a given sentiment, we choose three other sentiments at random, and sampled random sentences labelled with those sentiments from the training set to use as distractors. As mentioned in Section 5.2.2, the model's weights are updated conditioned on the weighted sum of the language model loss, and the next sentence prediction loss, $\mathcal{L} = \alpha\mathcal{L}_{\text{LM}} + \beta\mathcal{L}_{\text{NSP}}$, with $\mathcal{L}$ representing the total loss, $\mathcal{L}_{\text{LM}}$, the loss of the language model, $\mathcal{L}_{\text{NSP}}$, the loss of the next sentence prediction model, and α and β the weights given to each of the losses. We will consider $\alpha = 2$ and $\beta = 1$.

The results of this experiment can be observed in Table 6.32. Training the model for the next sentence prediction task worsens all metrics considerably. We hypothesize this is due to the amount of data being

Models	PPL	SES	F1 m	F1 m-NMC	F1 M	F1 M-NMC
Sentiment Sentences	**86.3**	**18.0**	**62.7**	**42.3**	**29.1**	**22.4**
Sentiment Sentences + Distractors	135.0	16.9	48.1	26.5	19.9	13.6

Table 6.32: Results obtained with the use of a double head Transformer model trained for the tasks of language modelling and next sentence prediction.

used to fine-tune the model, given that this approach worked in Devlin et al. (2019) and Wolf et al. (2019), which used much larger datasets than ours. Given the poor results, we opted for not using the double head transformer model.

6.4.6 EmotionPush Results

In order to evaluate the improvements achieved by the best developed model (DialoGPT-small + pre-defined set of sentiment sentences) we will consider two baseline models: the baseline (DialoGPT-small); and the tag model, as a sentiment conditioned baseline (DialoGPT-small + sentiment tag). The results obtained by the three models, for both development and test sets, can be observed in Table 6.33. As we saw in Section 6.4.4, the *Sentiment Sentences* model performs exceptionally well on the development set when compared to both the *Baseline* and the *Tag* approaches. Regarding the performance on the test set, despite the improvements not being as expressive, in particular, we achieve a better perplexity score on the *Tag* model, nonetheless, the *Sentiment Sentences* is the model that performs better on this set as well, showing clear improvements on the sentiment related metrics.

		PPL	SES	F1 m	F1 m-NMC	F1 M	F1 M-NMC
Dev	Baseline	92.4	16.8	42.5	15.6	13.2	6.4
	Tag	90.1	17.7	45.6	16.8	14.2	7.2
	Sentiment Sentences	**86.3**	**18.0**	**62.7**	**42.3**	**29.1**	**22.4**
Test	Baseline	85.0	17.3	47.6	22.6	17.1	10.3
	Tag	**78.9**	16.7	61.4	43.9	24.2	17.4
	Sentiment Sentences	79.4	**18.0**	**64.3**	**45.5**	**33.0**	**26.6**

Table 6.33: Results obtained on the development and test sets of the EmotionPush corpus with three different models: DialoGPT-small (Baseline); DialoGPT-small + sentiment tag (Tag); and DialoGPT-small + pre-defined set of sentiment sentences (Sentiment Sentences).

6.4.7 DailyDialog Results

Following what was done in the previous section, we will analyse the results obtained on the *Baseline*, *Tag*, and *Sentiment Sentences* models applied to the development and test sets of the DailyDialog corpus. The results can be observed in Table 6.34. Contrarily to the results obtained for the EmotionPush corpus, the *Tag* and the *Sentiment Sentences* models perform more similarly on this dataset. On the development set, it is relevant to mention that the *Tag* model outperforms the *Sentiment Sentences* model in the *macro-F1* score by 4.5 points, and in the *macro-F1 without the majority class* by 6.8 points. On the test set, the *Sentiment Sentences* model outperforms the *Tag* model on the sentiment related metrics, more significantly on the *macro* metrics, but scores worse results on the generation metrics.

				F1			
		PPL	**SES**	**m**	**m-NMC**	**M**	**M-NMC**
Dev	Baseline	9.7	28.5	82.7	25.7	20.9	9.4
	Tag	**9.3**	29.2	88.4	50.5	**42.5**	**34.0**
	Sentiment Sentences	**9.3**	**29.5**	**88.5**	**50.6**	38.0	28.8
Test	Baseline	9.9	26.7	77.9	30.6	24.9	14.5
	Tag	**9.5**	**28.5**	83.6	51.2	42.6	34.6
	Sentiment Sentences	9.6	27.3	**84.6**	**53.1**	**48.5**	**41.4**

Table 6.34: Results obtained on the development and test sets of the DailyDialog corpus with three different models: DialoGPT-small (baseline); DialoGPT-small + sentiment tag (Tag); and DialoGPT-small + pre-defined set of sentences (Sentiment Sentences).

As mentioned in Section 3.1, the EmotionPush corpus is retrieved in an online chat context, which means the text is very informal, while the DailyDialog corpus was built from websites that are used to practice English, which makes the corpus more formal and fluent. This aspect could influence the quality of the generation models. In particular, the fact that the *Tag* and *Sentiment Sentences* models fine-tuned with the DailyDialog corpus perform similarly could be an indication that the quality of the data used makes the models fine-tuned for this corpus not as dependent on the provided set of sentences, and a more simple option, such as a sentiment tag, is enough to guide the models. In contrast, the lower text quality of the EmotionPush corpus could be making the models fine-tuned for this dataset more reliant on full sentiment sentences in order to generate text conditioned on a sentiment.

6.4.8 Conclusion

Throughout the sentiment conditioned dialogue text generation experimentation cycle we showed the impact of dialogue appropriate pre-trained Transformer models on our problem, concluding that models that are pre-trained for the dialogue task in fact perform better. Unfortunately, for computational constraints, we were not able to use the best performing model, and had to resort to the smaller one. Next, we explored various methods to build a sentiment lexicon knowledge base as a way to guide the text generation model towards a generated sentence that better expresses a desired sentiment. Given the good results obtained when compared to the baseline model, we performed an experiment to show that the model is not simply copying the lexicon concatenated to the input. Furthermore, it was noticeable how adding sentiment information improves the performance substantially in terms of generating the desired sentiment, without compromising the quality of the generated text. Finally, we adapted the task of next sentence prediction for our use-case, through the use of sentiment distractors, which worsened all metrics considerably.

Considering the results obtained, we concluded that for both corpora the best text generation setup is the DialoGPT-small + pre-defined set of sentiment sentences, which is what we will be using on the sentiment-aware conversational agent.

6.5 Sentiment-aware Conversational Agent

With the experiment cycles of the sentiment classification (Section 6.2), reply sentiment prediction (Section 6.3), and text generation (Section 6.4) models complete, we are left with the task of joining the models into the proposed sentiment-aware conversational agent, first described in Chapter 1. In this section we will describe the setup of each model we are going to use, along with the results obtained on the development and test sets of the EmotionPush and DailyDialog corpora.

6.5.1 Setup

As mentioned in Section 6.3.5, given that using the sentiment of context sentences did not improve the reply sentiment prediction model's performance, the sentiment classification model is not part of the conversational agent, and it is only used as a metric to evaluate whether the generated sentences express the desired sentiment. The proposed sentiment-aware conversational agent system can be observed in Figure 6.5. The full setup includes: the reply sentiment prediction model that receives the previous context of a conversation (including the new user message) and outputs the appropriate sentiment for the conversational agent to express; and the text generation model that, given the predicted sentiment and the dialogue context, outputs a suitable reply. For each model, the setups we are going to use are the following:

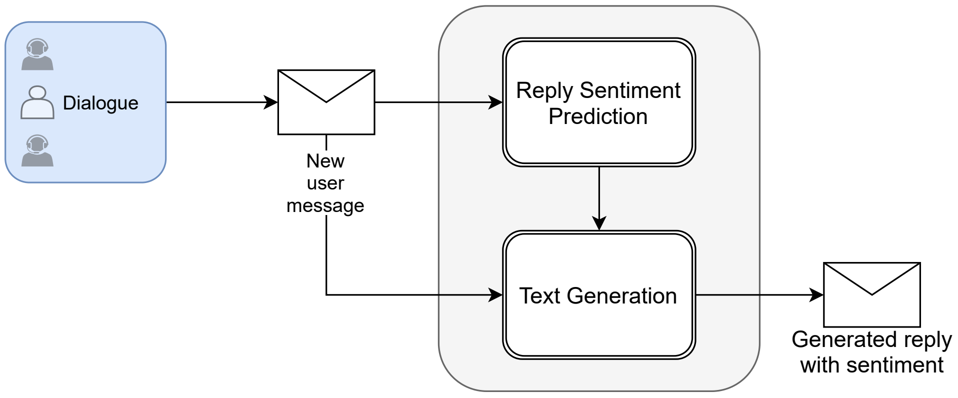


Figure 6.5: Proposed sentiment-aware conversational agent.

- **Sentiment Classifier**: RoBERTa-large + *concat4* pooling + one sentence as context + linear classification layer + retrieval augmentation initialized and with no resizing of the embeddings (no retrieval augmentation for the EmotionPush corpus);
- **Reply Sentiment Prediction**: RoBERTa-large + *concat4* sentence embedding representation + four input sentences;
- **Text Generation**: DialoGPT-small + set of sentiment sentences.

In order to evaluate the sentiment-aware conversational agent, we will consider three systems:

- **Baseline**: the DialoGPT-small model developed in Section 6.4.3, which is not conditioned on sentiment;

- **Sentiment Sentences**: the DialoGPT-small + Sentiment Sentences model developed in Section 6.4.4. Since this model is conditioned on the gold sentiment label, it represents the proposed sentiment-aware conversational agent if the reply sentiment prediction model was perfect;
- **Full System**: the proposed sentiment-aware conversation agent which includes the reply sentiment prediction and text generation models.

During the development of this work, Xie and Pu (2021) proposed an approach with a similar goal to ours. It also uses a reply sentiment prediction model, built with a Transformer Encoder, but the text generation model used is based on the original Transformer (Vaswani et al., 2017), which uses the sequence-to-sequence architecture. The predictions output by the reply sentiment prediction model are added to the input of the decoder part of the Seq2Seq model. However, contrarily to us, Xie and Pu (2021) focuses on larger datasets than the ones we used. Interestingly, their analysis on the reply sentiment prediction model, reaches similar conclusions to ours, which shows that the answer to achieve a better performance in the reply sentiment prediction task might not be directly related to the amount of data used to train the models, and other methods should be explored.

6.5.2 Results

The results obtained for the development and test sets of the EmotionPush and DailyDialog corpora, can be observed in Tables 6.35 and 6.36, respectively. For both datasets, the introduction of the reply sentiment prediction on the full system seems to be the bottleneck, given the lower performance achieved, namely on the sentiment related metrics. This tells us that the sentiment being predicted by the reply sentiment prediction model is steering the model towards the wrong emotion, as expected by the results obtained in Section 6.3. This proves the need for a better reply sentiment prediction model, which is key for the proper functioning of the proposed conversational agent. However, the full system still improves the *perplexity* when compared to the baseline, which shows that it is the set of sentences that is being concatenated to the input that is improving the *perplexity*, and not whether the model is receiving the correct sentiment or not. Regarding the *SES*, on the EmotionPush corpus, it is noticeable that the *SES* is better when the sentiment related metrics and the *perplexity* achieve better results. On the DailyDialog corpus this is not as perceptive, given the low fluctuation of the metrics between the evaluated models.

				F1			
		PPL	**SES**	**m**	**m-NMC**	**M**	**M-NMC**
Dev	Baseline	92.4	16.8	42.5	15.6	13.2	6.4
	Sentiment Sentences	**86.3**	**18.0**	**62.7**	**42.3**	**29.1**	**22.4**
	Full System	88.1	16.0	49.2	21.1	15.6	8.2
Test	Baseline	85.0	17.3	47.6	22.6	17.1	10.3
	Sentiment Sentences	**79.4**	**18.0**	**64.3**	**45.5**	**33.0**	**26.6**
	Full System	80.3	16.6	49.8	21.6	15.7	8.4

Table 6.35: Results obtained on the development and test sets of the EmotionPush corpus.

				F1			
		PPL	SES	m	m-NMC	M	M-NMC
Dev	Baseline	9.7	28.5	82.7	25.7	20.9	9.4
	Sentiment Sentences	**9.3**	**29.5**	**88.5**	**50.6**	**38.0**	**28.8**
	Full System	9.5	28.4	83.7	25.6	20.4	8.7
Test	Baseline	9.9	26.7	77.9	30.6	24.9	14.5
	Sentiment Sentences	**9.6**	27.3	**84.6**	**53.1**	**48.5**	**41.4**
	Full System	9.7	**27.6**	77.9	30.8	26.0	15.8

Table 6.36: Results obtained on the development and test sets of the DailyDialog corpus.

An example of the sentences generated by the developed sentiment-conditioned text generation model (*Sentiment Sentences*) can be observed in Table B.1. For the same context sentences (in Example 1 “How do you feel?” and in Example 2 “What do you think of this shirt?”) we present the sentences generated by the model conditioned on each of the target sentiments. Additionally, we also show the sentence generated by the baseline, which is not conditioned on any sentiment. We can observe that the model is capable of generating fluent and adequate answers for all sentiments. Furthermore, most sentences generated express the desired sentiment. Only on Example 2, the *fear* sentiment does not seem to be represented, however, the model was able to generate a sentence with a negative emotion. Another interesting aspect about the generated sentences is that the model only repeated keywords from the pre-defined set of sentiment sentences (presented in Section 6.4.4) twice on a sentence generated conditioned on the *fear* emotion. Other examples are available in Appendix B.

Example 1	
Context	How do you feel?
Baseline	I feel great, thanks.
Anger	I’m sick of this city.
Disgust	I’m very nauseous. I’ve never felt so nauseous.
Fear	I’m scared. I’m scared of the future.
Joy	I like it very much.
Sadness	I’m really sorry, I’m really not feeling well.
Surprise	Oh, my god! My eyes are burning!
Example 2	
Context	What do you think of this shirt?
Baseline	It’s beautiful.
Anger	Oh, it’s hideous.
Disgust	I don’t like it. It’s too plain.
Fear	I don’t like it. It’s too plain. How about a darker one?
Joy	I love it. I have the same shirt in black.
Sadness	I’ve never seen a shirt like this before. I don’t like it.
Surprise	What a terrible idea!

Table 6.37: Examples of sentences generated by the sentiment-conditioned text generation model using the pre-defined set of sentences given the same context. The words in red represent words that are present in the pre-defined set of sentences.

An example of a generated dialogue between the proposed sentiment-aware conversational agent and a user can be observed in Table 6.38. On all sentences generated in this example, the reply sentiment prediction model gave a higher probability to the *neutral* sentiment. In order to display a more interesting conversation, we chose the second best sentiment to condition the text generation model.

The text generation model seems to be able to generate the sentiments it is conditioned to. Furthermore, choosing the second best sentiment also resulted in a sentiment-appropriate conversation. The generated sentences also seem to consider the past context, with the only mistake being on the second chatbot sentence, where the model confused who called the police. Again, the sentence generated with the *fear* emotion repeated a keyword from the pre-defined set of sentiment sentences. Interestingly, as shown in Section 6.1, the *fear* label is the most underrepresented sentiment in the corpus, which might have made the model rely more on the provided set.

Dialogue Example		
User:		I just called the police, I think someone broke into our home while we were at work
Chatbot:	`[FEAR]`	I am really scared.
User:		Me too! What do we do now?
Chatbot:	`[JOY]`	I'm really glad that I called the police. I really wanted to go home.
User:		I just want this nightmare to be over.
Chatbot:	`[ANGER]`	Oh, it's not over. We are going to have to go home now.

Table 6.38: Example of a dialogue between a user and the proposed sentiment-aware conversational agent. `[SENTIMENT]` represents the sentiment predicted by the reply sentiment prediction model and that is used to condition the reply of the conversational agent on that utterance. The words in red represent words that are present in the pre-defined set of sentences.

6.6 Summary

Throughout this chapter we explored the tasks of sentiment classification (Section 6.2), reply sentiment prediction (Section 6.3), and dialogue text generation (Section 6.4) applied to the EmotionPush and DailyDialog corpora, in order to build a sentiment-aware conversational agent. During the sentiment classification experimental cycle we took advantage of being in a dialogue setting and used past dialogue context to enrich the input of the model, which improved the performance of our model. We also experimented with retrieval augmentation, and concluded that in the datasets we are working with, there are not enough similar sentences for this approach to be impactful. Regarding the reply sentiment prediction model, our results highlighted the difficulty of performing well in this task, and how due to that, it is the bottleneck of the conversational agent. In particular, we have concluded that using information about the labels of the context sentences for reply sentiment prediction has a negative impact on the model's performance. This means that the sentiment classification model is not part of the final pipeline of sentiment-aware conversational agent, and is only used as a metric to evaluate the sentiment of the sentences generated by the text generation models. Finally, our proposed conditioned dialogue text generation model achieved results that outperformed significantly the baseline model, through the use of a pre-defined set of sentences that represent each sentiment. The final results obtained for the sentiment-aware conversational agent fell short of our expectations, mainly due to the low results obtained on the reply sentiment prediction model. Nevertheless, the high scores obtained by the text generation model when compared to the baseline, both on the generation and the sentiment metrics, motivate the use of sentiment-guided text generation as a way of building a sentiment-aware conversational agent.

Chapter 7

Human Evaluation

As mentioned in Xie and Pu (2021), automatically evaluating empathetic conversational agents is a challenging task given the limitations of automatic metrics. In particular, the most common text generation metrics evaluate the word/lexical overlap between the gold and generated sentences, and in a dialogue setting there can be many correct answers. Furthermore, the experiment reported in Section 6.4.7 further motivates the challenge of relying on these metrics to evaluate sentiment-aware conversational agents: giving to the model the exact same context and conditioning it on different sentiments drastically changes the outcome of the generated sentences. For that matter, human evaluation became a popular method to evaluate conversational agents. The limitation of the human evaluation is that it is a time consuming and expensive evaluation. For this evaluation we were able to gather answers from seven annotators with a proficient English level.

7.1 Evaluation Method

The goal of this work is to build a sentiment-aware conversational agent without compromising the quality of the generated text. With this in mind, the most important aspects to evaluate regarding the generated text were the fluency, the adequacy given the previous context, and whether it expressed the desired sentiment.

7.1.1 Sampling

This evaluation was done by sampling at random 40 inputs from the test set of each corpus and retrieving the corresponding replies generated by four of the developed architectures:

- **Baseline**: which consists on the DialoGPT-small model;
- **Sentiment Baseline**: which consists on the DialoGPT-small + *tag* setup. This model allows us to have a baseline that is conditioned on a sentiment;
- **Sentiment Sentences**: which consists on the DialogGPT-small + pre-defined set of sentences

defined in Section 6.4.4. This model represents the sentiment-aware conversational agent if the reply sentiment prediction model was perfect.

- **Full System**: which consists on the proposed sentiment-aware conversational agent, with the reply sentiment prediction and text generation models.

One relevant detail about the chosen architectures is that both the sentiment baseline and the set of sentences approaches use the gold sentiment label to condition the sentiment of the generated sentence. In that sense, these approaches can be considered as being in the sentiment-aware conversational agent scenario, where the reply sentiment prediction model works perfectly.

The number of examples sampled of each sentiment can be observed in Table 7.1. The sampling was done with the goal of balancing the number of examples of each sentiment. For the EmotionPush corpus this was not possible given that for some sentiments we did not have enough examples available.

	Nr. of Examples					
Corpus	**Anger**	**Disgust**	**Fear**	**Joy**	**Sadness**	**Surprise**
EmotionPush	7	6	1	9	9	8
DailyDialog	7	7	7	6	7	6

Table 7.1: Number of examples sampled of each sentiment.

7.1.2 Evaluation Metrics

As mentioned before, the metrics we found interesting to evaluate were the fluency, the adequacy of the reply given the previous context, and whether the generated text expressed the desired sentiment.

We considered a sentence to not be fluent if it had grammatical errors or repetitions. Through an informal inspection of the sampled sentences, there were no such cases on the DailyDialog corpus sample. On the EmotionPush sample we found three replies with repetitions and one reply with a grammatical error out of the 160 replies retrieved (40 generated by each considered model). In order to make the evaluation easier and less time consuming for the annotators, and considering that through the informal inspection we did not find a considerable number of examples with fluency problems, we did not ask the annotators to evaluate the fluency of the generated texts.

In order to evaluate the adequacy of the reply we asked the annotators the following question: "*Do the replies sound appropriate considering the context of the dialogue?*". We use a 2-point Likert scale for this question. A similar process was followed to evaluate the sentiment of the sentences. Our goal with this evaluation was to assess if the model was able to generate sentences with a desired sentiment. The question asked to the annotators was "*Do the replies represent the* `<sentiment_name>` *emotion?*". In this evaluation we asked the annotators specifically not to consider the previous context of the replies. Additionally, given the multitude of sentiments present that often can be interchanged, we also asked them to consider whether the sentence being evaluated could express the asked sentiment. For example, "*What?*" could be used to express *anger* or *surprise*, depending on the tone used. We also use a 2-point Likert scale of for this question.

An important aspect to notice regarding the questions made is our choice of evaluation scale. Van Der Lee et al. (2019) performs an analysis on how human evaluations were conducted in recent works. This work concludes that the 5-point Likert scale is the most common choice, followed by preference ratings and the 2-point Likert scale. We chose a 2-point Likert scale in our work due to the following reasons: each annotator has to analyze 40 dialogues for each dataset, which makes using a larger scale cumbersome; we were concerned that using a larger scale could make the annotators compelled to choose middle-of-the-scale values, making it harder to take conclusions from this evaluation.

Another interesting detail to consider was the feedback we got from the annotators regarding the sentiment metrics. Different annotators considered some expressions differently. An example of this is the expression "*lol*". Some annotators considered this expression as *joy*, while others, who use this expression sarcastically, considered it as "anger". This aspect shows some of the difficulties of annotating corpora with sentiment, particularly on a more fine-grained level.

7.2 Results

In this section, we will present the results obtained for the human evaluation. We will start by showing the scores obtained on the evaluated metrics, followed by a study on the annotators' agreement. Finally, we will analyze how the human evaluation metrics correlate with the automatic metrics reported in Chapter 6.

7.2.1 Scores

The results obtained for the human evaluation performed on the dialogues sampled from the EmotionPush corpus can be observed in Table 7.2. Given that we are using a Likert scale of 2-points the reported scores correspond to the ratio of positive answers to a given question. E.g., an *adequacy* score of 0.6 for the *Baseline* model means that 60% of the generated sentences by this model were considered adequate. The *sentiments* score corresponds to the ratio of positive answers considering all sentiments. It is clear that the model that achieves the best performance was the *Sentiment Sentences* model. In particular, we highlight how this model improved the *adequacy* of the replies when compared to all other models. It also achieves the highest *sentiment* scores, except on *disgust* sentences, where it is outperformed by the *Sentiment Baseline* model. It is also interesting to observe that, despite the accumulated error of the *Full System* due to the reply sentiment prediction model, the *Full System* still outperforms the *Baseline* by 0.0292 points on the *sentiments* metric. Nonetheless, it achieves a worse performance on the *adequacy* metric. There also seems to exist a correlation between the *sentiments* metric and how adequate the replies are. The models that achieved a higher sentiment metric also tend to be more adequate.

The results obtained on the evaluation done on the DailyDialog corpus are presented in Table 7.3. Both *Sentiment Baseline* and *Sentiment Sentences* approaches perform very similarly, with the *Sentiment Baseline* model scoring higher on the *adequacy* metric, and the *Sentiment Sentences* model

Model	Adequacy	Sentiments	Anger	Disgust	Fear	Joy	Sadness	Surprise
Baseline	0.4292	0.325	0.3095	0.1667	0.1667	0.2778	0.4074	0.4375
Sentiment Baseline	0.5042	0.5167	0.2381	**0.5278**	0	0.7963	0.5926	0.4167
Sentiment Sentences	**0.6167**	**0.6583**	**0.4762**	0.4167	**0.8334**	**0.8889**	**0.6481**	**0.7292**
Full System	0.3917	0.3542	0.3810	0.4167	0.1667	0.5556	0.2223	0.2292

Table 7.2: Human evaluation average EmotionPush Scores.

scoring higher on the *sentiments* metric. The improvements of both models when compared to the *Baseline* are considerable on all metrics.

Model	Adequacy	Sentiments	Anger	Disgust	Fear	Joy	Sadness	Surprise
Baseline	0.4958	0.3708	0.3095	0.2381	0.3571	0.6945	0.3571	0.3056
Sentiment Baseline	**0.6542**	0.7084	0.5714	**0.6190**	0.4762	**0.9723**	0.7381	0.9445
Sentiment Sentences	0.6292	**0.7625**	**0.7619**	0.4762	**0.6667**	0.9167	**0.8334**	**0.9723**
Full System	0.5667	0.3292	0.3334	0.1905	0.2857	0.5556	0.3810	0.2500

Table 7.3: Human evaluation average DailyDialog Scores.

It is relevant to mention that a common complaint among the annotators was that the dialogues' context present in the EmotionPush corpus evaluation were often times confusing and hard to understand. As mentioned in Section 6.1, this corpus was retrieved in an online chat setting which made some of the dialogues very informal.

7.2.2 Inter Annotator Agreement

In order to evaluate the agreement among the annotators we chose the Fleiss' Kappa (Fleiss and Cohen, 1973) statistical test, which allows to measure the agreement between two or more annotators.

The agreement results obtained can be observed in Table 7.4. According to the Kappa's interpretation scale proposed by Viera et al. (2005), the Fleiss' Kappa can assume values between 0 (poor agreement) and 1 (almost perfect agreement). In our study there is either a slight (0.01-0.2), fair (0.2-0.4), moderate (0.41-0.60), or substantial (0.61-0.80) agreement. In particular, we can observe that, even though the agreement is generally higher on the DailyDialog corpus, in both models the annotators achieve a moderate agreement on the sentiments and a fair agreement on the adequacy.

Metric	EmotionPush	DailyDialog
Adequacy	0.2877	0.3617
Avg. Sentiments	0.4378	0.5257
Anger	0.1901	0.3190
Disgust	0.3234	0.4043
Fear	0.3950	0.3786
Joy	0.4918	0.3998
Sadness	0.5797	0.7122
Surprise	0.3905	0.7470

Table 7.4: Fleiss' Kappa agreement among the annotators.

7.2.3 Correlation with Automatic Metrics

The automatic evaluation performed on these models in Sections 6.4 and 6.5 showed that on the EmotionPush corpus the model that performed better was the *Sentiment Sentences* model. Regarding the DailyDialog, we observed that the *Sentiment Sentences* model also achieved the best performance. However, for the DailyDialog corpus, the *Sentiment Baseline* model had a similar performance to it, which led us to believe that a simpler approach already achieved interesting results.

We use the Pearson correlation to measure the linear relationship between the automatic and the human evaluation metrics. The Pearson correlation ranges from -1 to 1. A correlation -1 indicates a perfect negative correlation, 1 a positive correlation, and 0 no correlation. In particular, we correlate four points, corresponding to pairs of the automatic and human metrics obtained for each evaluated model. It is important to mention that we should take into consideration that a correlation using four points is not ideal, and might not lead to statistically significant results.

We can observe the correlation between the automatic metrics (*perplexity*, *sentence embedding similarity*, and *micro/macro-F1 without the majority class*) and the human evaluation metrics (*adequacy* and average of the sentiments) for the EmotionPush and DailyDialog corpora in Tables 7.5 and 7.6, respectively. As previously mentioned, the annotators had some complaints regarding the sentences of the dialogues on the EmotionPush corpus. This uncertainty can be seen in the correlation between the human evaluation metrics and the *perplexity* which is very low. Regarding the same correlation on the DailyDialog corpus, we can observe that the correlation between the *perplexity* and the *sentiments* metric is high, while between the *perplexity* and the *adequacy* is close to perfect. This highlights the difference in the quality of the text in both datasets. Regarding the correlation between the human evaluation metrics and the *sentence embedding similarity*, on the EmotionPush corpus is close to perfect on both metrics, while on the DailyDialog the correlation between the *SES* and the *adequacy* is high, but between the *SES* and the *sentiments* metric is lower. This could be related to the fact that on this evaluation the *Sentiment Sentences* performs better on the sentiments metric, while on the automatic evaluation performed in Section 5.2 we saw that the *Sentiment Baseline* model performed better on the *SES* metric. Finally, we can observe that the correlation between the sentiment automatic metrics and the human evaluation metrics is also high for both corpora.

It is relevant to highlight that, as discussed in Section 6.4.7, models fine-tuned with formal and fluent data, such as the DailyDialog corpus, seem to perform well with simpler sentiment-conditioning approaches. In contrast, models fine-tuned with informal data, such as the EmotionPush corpus, seem to rely more on full sentiment sentences in order to perform well. This hypothesis is further validated with these experiments, since we have seen that the annotators gave higher scores to the *Sentiment Sentences* model fine-tuned for the EmotionPush corpus, while the annotations gathered for the DailyDialog corpus showed similar results for the *Sentiment Sentences* and *Sentiment Baseline* approaches.

We can conclude that the correlation between the automatic metrics and the human evaluation metrics is generally high and the results obtained on the automatic metrics are trustworthy given that we achieve similar results on the human evaluation. This encourages the use of these automatic metrics during the development of these models.

	PPL	SES	m-NMC	M-NMC
Adequacy	0.18	0.9524	0.8015	0.8522
Sentiments	0.01	0.9519	0.7992	0.8328

Table 7.5: Pearson correlation between the automatic metrics (perplexity, sentence embedding similarity, micro/macro-F1 without the majority class) and the human evaluation metrics (adequacy and sentiments) applied to the EmotionPush dataset.

	PPL	SES	m-NMC	M-NMC
Adequacy	-0.981	0.7992	0.8852	0.8589
Sentiments	-0.7345	0.4475	0.9966	0.9839

Table 7.6: Pearson correlation between the automatic metrics (perplexity, sentence embedding similarity, micro/macro-F1 without the majority class) and the human evaluation metrics (adequacy and sentiments) applied to the DailyDialog dataset.

7.3 Summary

Throughout this chapter we described the human evaluation performed on the generated sentences by four developed models: a baseline, a sentiment-aware baseline, the proposed sentiment-aware conversational agent, and the same agent if the reply sentiment prediction model was perfect. Similarly to what was concluded in Sections 6.4 and 6.5, we saw that, on the EmotionPush corpus, the model that achieves the best performance is the text generation model that uses a pre-defined set of sentiment sentences, which corresponds to the sentiment-aware conversational agent if the reply sentiment prediction model was perfect. Once again, this serves as motivation to work on the reply sentiment prediction model and try to improve its performance as much as possible. On the DailyDialog we achieved similar conclusions, and could observe that similarly to the results obtained on the automatic metrics, the sentiment-aware baseline already achieved interesting results. In particular, as discussed in Section 6.4.7, and further validated in the human evaluation experiments, models fine-tuned with corpora with formal and fluent data, such as the DailyDialog corpus, seem to perform well with simpler sentiment-conditioning approaches. In contrast, more informal corpora, such as the EmotionPush corpus, seem to rely more on full sentiment sentences in order to perform well. Finally, we observed the correlation of the human experiments with the automatic metrics and concluded that for most metrics the correlation is high, which means that the results obtained on the automatic metrics are trustworthy and validated by this human experiment. However, we should take into consideration that a low number of points was used for the correlation, which is not ideal.

Chapter 8

Conclusions and Future Work

8.1 Contributions

In Section 1.1 we proposed an end-to-end sentiment-aware conversational agent to help mitigate the constraints of current text generation models, which rely on large volumes of data in order to implicitly learn specific text characteristics, such as sentiment. This system consists on three models: a sentiment classifier, a reply sentiment predictor, and a conditioned text generation model. We applied the proposed system to the EmotionPush (Hsu et al., 2018) and the DailyDialog (Li et al., 2017) corpora.

We started by describing the state-of-the-art for sentiment classification and reply sentiment prediction models, in particular, we explored contextual sentiment classification, as a way to give previous dialogue context to the classification model. We found that for sentiment classification, using one previous context sentence resulted in considerable improvements on the evaluated metrics. For reply sentiment prediction, since the goal of the model was not to classify any sentence in the input, but instead, predict an appropriate reply sentiment after a given dialogue context, using four previous context sentences as input displayed the best results. Through experimentation we also found that, for the task of reply sentiment prediction, using information regarding the labels of past context sentences did not improve the results. We hypothesize that this might be due to the label unbalancing of the used corpora, and the fact that there are not significant patterns when it comes to sequences of sentiments. For that matter, we removed the sentiment classification model from the originally proposed pipeline, and only used this model to classify the sentences generated by the text generation model in order to obtain sentiment-related metrics. Additionally, we also explored retrieval augmentation techniques to aid both the sentiment classifier and the reply sentiment prediction models. In particular, we adapted the work of Ramos et al. (2021), that explores a mechanism based on finding the nearest neighbors among the training set of the evaluated development/test splits to aid classification using LSTM models, to the Transformer model architecture. We concluded that this approach worsened the sentiment classification and the reply sentiment prediction models fine-tuned for the EmotionPush corpus, and slightly improved the models fine-tuned for the DailyDialog corpus. We attributed this to the different degrees of similarity between the evaluated development/test splits and the training splits. For the EmotionPush corpus, we found that the similarities between the validation/test splits and the nearest training examples were generally lower than the similarities observed for the DailyDialog corpus.

Regarding the conditioned text generation model, we followed Wolf et al. (2019), that develops a text generation model able to express a given persona, and adapted it to our use-case. In particular, conditioning the text generation model with sentiment by concatenating a pre-defined set of sentences related to the desired sentiment, resulted in significant improvements when compared to the baseline, both in the quality of the generated text and the expressed sentiment. We also found that for corpora with a smaller number of training instances, such as the EmotionPush, this approach showed expressive improvements, while for larger corpora, such as the DailyDialog, a simpler approach, such as concatenating a sentiment tag, already exhibits interesting results. In order to justify these improvements, we performed extra experiments that aimed to verify if the text generation model was simply copying the provided sentences, and if that was what was causing the improvement in the sentiment metrics. We found that, even though some keywords were present in the generated sentences, they did not fully explain the sentiment metrics' improvements.

With the three models developed, we were able to build the proposed sentiment-aware conversational agent. The results using the full system fell short of our expectations, given the low results obtained by the reply sentiment prediction model, which we identified as the bottleneck of the proposed system. However, when we compare the text generation model that makes use of a "perfect" reply sentiment prediction model, with an end-to-end baseline that uses only a text generation model, our approach shows significant improvements both in the quality of the generated text, and the expressed sentiment. Moreover, we empathize the ability of our conditioned text generation model of generating sentences conditioned on the different target sentiments, using same context. Furthermore, we also showed examples of dialogues between a user and the proposed system and verified that the generated sentences by the system were both context aware and sentiment appropriate.

We finalized this work by conducting a human evaluation on the developed models. To do so, we asked annotators to evaluate sentences generated by four text generation models: a baseline that is not conditioned on sentiment; a baseline that is conditioned on sentiment, through the concatenation of a sentiment tag; a conditioned text generation approach, through the concatenation of a pre-defined set of sentences that represent each label; and the proposed sentiment-aware conversational agent. We highlight that both models conditioned on sentiment represent the proposed agent if the reply sentiment prediction model was perfect. We concluded that the results obtained on the human evaluation are well correlated with the automatic results, and therefore the conclusions are similar.

8.2 Future Work

In this section we will describe some future directions for the developed work. In particular, given that the proposed system is composed of three models, we will first describe possible improvements for the sentiment classification and reply sentiment prediction models, and then for the sentiment-conditioned text generation model.

8.2.1 Sentiment Classification and Reply Sentiment Prediction

During this work, the techniques applied to the sentiment classification and the reply sentiment prediction tasks were similar. The reason behind this is that there is little research on the task of reply sentiment prediction as a data-driven task and, therefore, we used the sentiment classification approaches as a starting point. Even though the results obtained for the reply sentiment prediction task fell short of expectations, we believe that a data-driven approach for this task should be further explored. In that sense, future approaches that could be investigated for both tasks could further explore retrieval augmentation as a way to better contextualize the embeddings for classification/prediction. In particular, instead of using the labels as a hint to the classification model, we could use similar contexts, similarly to what Wang et al. (2021) does for token-level classification tasks. Another approach that has shown promising results for text classification is prompt-based learning. Prompt-based learning consists on augmenting the dataset's examples with a given template, as showcased in Han et al. (2021). Instead of classifying the example directly, the example is augmented with a template that includes a masked entry. Then, a masked language model is tasked with predicting the masked entry, which is then classified with a given sentiment, for instance. Such approaches allow the classification model to focus on specific parts of the sentence, thus, aiding the classification. Additionally, it is possible to explore different templates which can be more appropriate depending on the task.

Regarding specifically the reply sentiment prediction model, it could be interesting to explore hybrid approaches, with both data-driven models and pre-defined rules. In particular, some heuristics applied to the prediction's confidence could be adopted in order to improve the model.

8.2.2 Sentiment-Conditioned Text Generation

Adapting the work by Wolf et al. (2019) for our specific problem proved to perform well in conditioning the text generation model to express sentiment. Nonetheless, further improvements could be explored on the decoding side. An advantage of exploring the decoding part of the text generation model is that it does not require any further training. In particular, several works have used retrieval augmentation in the decoding to improve tasks such as machine translation (Khandelwal et al., 2020), and text generation (Lewis et al., 2020; Khandelwal et al., 2019). The goal of such an approach would be to interpolate the probability distribution yielded by text generation model with information from a knowledge base, which can include example sentences of multiple emotions.

Bibliography

Joseph Weizenbaum. Eliza—a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 9(1):36–45, 1966.

Anton Leuski, Ronakkumar Patel, David Traum, and Brandon Kennedy. Building effective question answering characters. In *Proceedings of the 7th SIGdial Workshop on Discourse and Dialogue*, pages 18–27, 2006.

Anton Leuski and David Traum. Npceditor: Creating virtual human dialogue using information retrieval techniques. *Ai Magazine*, 32(2):42–56, 2011.

Zongcheng Ji, Zhengdong Lu, and Hang Li. An information retrieval approach to short text conversation. *arXiv preprint arXiv:1408.6988*, 2014.

Daniel Jurafsky and James H. Martin. *Speech and Language Processing (2nd Edition)*. Prentice-Hall, Inc., USA, 2009. ISBN 0131873210.

Maria João Pereira, Luisa Coheur, Pedro Fialho, Ricardo Ribeiro, et al. Chatbots' greetings to human-computer communication. In *CEUR Workshop Proceedings*, volume 2390, pages 61–66, 2019.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

Oriol Vinyals and Quoc Le. A neural conversational model, 2015.

Lifeng Shang, Zhengdong Lu, and Hang Li. Neural responding machine for short-text conversation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1577–1586, 2015.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations, 2018.

Hannah Rashkin, Eric Michael Smith, Margaret Li, and Y-Lan Boureau. Towards empathetic open-domain conversation models: A new benchmark and dataset. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5370–5381, 2019.

Ning Miao, Yuxuan Song, Hao Zhou, and Lei Li. Do you have the right scissors? tailoring pre-trained language models via monte-carlo methods. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3436–3441, 2020.

Hao Zhou, Minlie Huang, Tianyang Zhang, Xiaoyan Zhu, and Bing Liu. Emotional chatting machine: Emotional conversation generation with internal and external memory. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Chenyang Huang, Osmar R Zaiane, Amine Trabelsi, and Nouha Dziri. Automatic dialogue generation with expressed emotions. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 49–54, 2018.

Thomas Wolf, Victor Sanh, Julien Chaumond, and Clement Delangue. Transfertransfo: A transfer learning approach for neural network based conversational agents. *arXiv preprint arXiv:1901.08149*, 2019.

Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. Plug and play language models: a simple approach to controlled text generation. *arXiv preprint arXiv:1912.02164*, 2019.

Nitish Shirish Keskar, Bryan McCann, Lav R Varshney, Caiming Xiong, and Richard Socher. Ctrl: A conditional transformer language model for controllable generation. *arXiv preprint arXiv:1909.05858*, 2019.

Yubo Xie and Pearl Pu. Generating empathetic responses with a large scale dialog dataset. *arXiv preprint arXiv:2105.06829*, 2021.

Moshe Davidow. Organizational responses to customer complaints: What works and what doesn't. *Journal of service research*, 5(3):225–250, 2003.

Tianran Hu, Anbang Xu, Zhe Liu, Quanzeng You, Yufan Guo, Vibha Sinha, Jiebo Luo, and Rama Akkiraju. Touch your heart: A tone-aware chatbot for customer care on social media. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, pages 1–12, 2018.

Xiang Li and Ming Zhang. Emotion analysis for the upcoming response in open-domain human-computer conversation. In *Asia-Pacific Web (APWeb) and Web-Age Information Management (WAIM) Joint International Conference on Web and Big Data*, pages 352–367. Springer, 2018.

Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323, 2011.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

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

Paul J Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318, 2013.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681, 1997.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation, 2014.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642, 2013.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27:3104–3112, 2014.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate, 2016.

Peter J Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. Generating wikipedia by summarizing long sequences. *arXiv preprint arXiv:1801.10198*, 2018.

Yoav Goldberg. Neural network methods for natural language processing. *Synthesis Lectures on Human Language Technologies*, 10(1):1–309, 2017.

George James Lidstone. Note on the general case of the bayes-laplace formula for inductive or a posteriori probabilities. *Transactions of the Faculty of Actuaries*, 8(182-192):13, 1920.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155, 2003.

Stefan Kombrink, Tomáš Mikolov, Martin Karafiát, and Lukáš Burget. Recurrent neural network based language modeling in meeting recognition. In *Twelfth annual conference of the international speech communication association*, 2011.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training, 2018.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

Wilson L Taylor. "cloze procedure": A new tool for measuring readability. *Journalism quarterly*, 30(4): 415–433, 1953.

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

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.

Ronald J Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280, 1989.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

Chao-Chun Hsu, Sheng-Yeh Chen, Chuan-Chun Kuo, Ting-Hao Huang, and Lun-Wei Ku. EmotionLines: An emotion corpus of multi-party conversations. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan, May 2018. European Language Resources Association (ELRA). URL `https://www.aclweb.org/anthology/L18-1252`.

Dorottya Demszky, Dana Movshovitz-Attias, Jeongwoo Ko, Alan Cowen, Gaurav Nemade, and Sujith Ravi. Goemotions: A dataset of fine-grained emotions, 2020.

Robert Plutchik. A general psychoevolutionary theory of emotion. In *Theories of emotion*, pages 3–33. Elsevier, 1980.

Paul Ekman. An argument for basic emotions. *Cognition & emotion*, 6(3-4):169–200, 1992.

Lei Zhang, Shuai Wang, and Bing Liu. Deep learning for sentiment analysis: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(4):e1253, 2018.

Yanran Li, Hui Su, Xiaoyu Shen, Wenjie Li, Ziqiang Cao, and Shuzi Niu. Dailydialog: A manually labelled multi-turn dialogue dataset, 2017.

Y. Zhang, Z. Zhao, P. Wang, X. Li, L. Rong, and D. Song. Scenariosa: A dyadic conversational database for interactive sentiment analysis. *IEEE Access*, 8:90652–90664, 2020. doi: 10.1109/ACCESS.2020.2994147.

Jonathan Herzig, Guy Feigenblat, Michal Shmueli-Scheuer, David Konopnicki, Anat Rafaeli, Daniel Altman, and David Spivak. Classifying emotions in customer support dialogues in social media. In *Proceedings of the 17th annual meeting of the special interest group on discourse and dialogue*, pages 64–73, 2016.

Katja Gelbrich. Anger, frustration, and helplessness after service failure: coping strategies and effective informational support. *Journal of the Academy of Marketing Science*, 38(5):567–585, 2010.

Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. How to fine-tune bert for text classification?, 2020.

Matthew E. Peters, Sebastian Ruder, and Noah A. Smith. To tune or not to tune? adapting pretrained representations to diverse tasks, 2019.

Jonggu Kim, Hyeonmok Ko, Seoha Song, Saebom Jang, and Jiyeon Hong. Contextual augmentation of pretrained language models for emotion recognition in conversations. In *Proceedings of the Third Workshop on Computational Modeling of People's Opinions, Personality, and Emotion's in Social Media*, pages 64–73, 2020.

Yen-Hao Huang, Ssu-Rui Lee, Mau-Yun Ma, Yi-Hsin Chen, Ya-Wen Yu, and Yi-Shin Chen. Emotionx-idea: Emotion bert–an affectional model for conversation. *arXiv preprint arXiv:1908.06264*, 2019.

Weizhou Shen, Siyue Wu, Yunyi Yang, and Xiaojun Quan. Directed acyclic graph network for conversational emotion recognition. *arXiv preprint arXiv:2105.12907*, 2021.

Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

Chandrakant Bothe, Sven Magg, Cornelius Weber, and Stefan Wermter. Dialogue-based neural learning to estimate the sentiment of a next upcoming utterance. In *International Conference on Artificial Neural Networks*, pages 477–485. Springer, 2017.

Zhiyuan Wen, Jiannong Cao, Ruosong Yang, Shuaiqi Liu, and Jiaxing Shen. Automatically select emotion for response via personality-affected emotion transition. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 5010–5020, 2021.

Albert Mehrabian. Pleasure-arousal-dominance: A general framework for describing and measuring individual differences in temperament. *Current Psychology*, 14(4):261–292, 1996.

Xianda Zhou and William Yang Wang. Mojitalk: Generating emotional responses at scale. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1128–1137, 2018.

Pierre Colombo, Wojciech Witon, Ashutosh Modi, James Kennedy, and Mubbasir Kapadia. Affect-driven dialog generation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3734–3743, 2019.

Sashank Santhanam and Samira Shaikh. Emotional neural language generation grounded in situational contexts. In *Proceedings of the 4th Workshop on Computational Creativity in Language Generation*, pages 22–27, 2019.

Nabiha Asghar, Pascal Poupart, Jesse Hoey, Xin Jiang, and Lili Mou. Affective neural response generation. In *European Conference on Information Retrieval*, pages 154–166. Springer, 2018.

Ali Yadollahi, Ameneh Gholipour Shahraki, and Osmar R Zaiane. Current state of text sentiment analysis from opinion to emotion mining. *ACM Computing Surveys (CSUR)*, 50(2):1–33, 2017.

Jingyuan Li and Xiao Sun. A syntactically constrained bidirectional-asynchronous approach for emotional conversation generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 678–683, 2018.

Linhong Xu, Hongfei Lin, Yu Pan, Hui Ren, and Jianmei Chen. Constructing the affective lexicon ontology. *Journal of the China society for scientific and technical information*, 27(2):180–185, 2008.

Peixiang Zhong, Di Wang, and Chunyan Miao. An affect-rich neural conversational model with biased attention and weighted cross-entropy loss. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7492–7500, 2019a.

Nurul Lubis, Sakriani Sakti, Koichiro Yoshino, and Satoshi Nakamura. Eliciting positive emotion through affect-sensitive dialogue response generation: A neural network approach. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Hannah Rashkin, Eric Michael Smith, Margaret Li, and Y-Lan Boureau. I know the feeling: Learning to converse with empathy. 2018.

Emily Dinan, Stephen Roller, Kurt Shuster, Angela Fan, Michael Auli, and Jason Weston. Wizard of wikipedia: Knowledge-powered conversational agents. *arXiv preprint arXiv:1811.01241*, 2018.

Nikita Moghe, Siddhartha Arora, Suman Banerjee, and Mitesh M Khapra. Towards exploiting background knowledge for building conversation systems. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *arXiv preprint arXiv:2005.11401*, 2020.

Rita Parada Ramos, Patrícia Pereira, Helena Moniz, Joao Paulo Carvalho, and Bruno Martins. Retrieval augmentation for deep neural networks. *arXiv preprint arXiv:2102.13030*, 2021.

Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019. URL `https://arxiv.org/abs/1908.10084`.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 2019.

Yizhe Zhang, Siqi Sun, Michel Galley, Yen-Chun Chen, Chris Brockett, Xiang Gao, Jianfeng Gao, Jingjing Liu, and William B Dolan. Dialogpt: Large-scale generative pre-training for conversational response generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 270–278, 2020.

Sid Black, Leo Gao, Phil Wang, Connor Leahy, and Stella Biderman. GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow, March 2021. URL `https://doi.org/10.5281/zenodo.5297715`.

Devamanyu Hazarika, Soujanya Poria, Roger Zimmermann, and Rada Mihalcea. Conversational transfer learning for emotion recognition. *Information Fusion*, 65:1–12, 2021.

William Falcon and The PyTorch Lightning team. PyTorch Lightning, 3 2019. URL `https://github.com/PyTorchLightning/pytorch-lightning`.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Perric Cistac, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-Art Natural Language Processing. pages 38–45. Association for Computational Linguistics, 10 2020. URL `https://www.aclweb.org/anthology/2020.emnlp-demos.6`.

Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339, 2018.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

Chunning Du, Haifeng Sun, Jingyu Wang, Qi Qi, and Jianxin Liao. Adversarial and domain-aware bert for cross-domain sentiment analysis. In *Proceedings of the 58th annual meeting of the Association for Computational Linguistics*, pages 4019–4028, 2020.

Xiao Chen, Changlong Sun, Jingjing Wang, Shoushan Li, Luo Si, Min Zhang, and Guodong Zhou. Aspect sentiment classification with document-level sentiment preference modeling. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3667–3677, 2020.

Sopan Khosla. Emotionx-ar: Cnn-dcnn autoencoder based emotion classifier. In *Proceedings of the sixth international workshop on natural language processing for social media*, pages 37–44, 2018.

Deepanway Ghosal, Navonil Majumder, Alexander Gelbukh, Rada Mihalcea, and Soujanya Poria. Cosmic: Commonsense knowledge for emotion identification in conversations. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 2470–2481, 2020.

Peixiang Zhong, Di Wang, and Chunyan Miao. Knowledge-enriched transformer for emotion detection in textual conversations. *arXiv preprint arXiv:1909.10681*, 2019b.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318, 2002.

Satanjeev Banerjee and Alon 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*, pages 65–72, 2005.

Chia-Wei Liu, Ryan Lowe, Iulian Vlad Serban, Mike Noseworthy, Laurent Charlin, and Joelle Pineau. How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2122–2132, 2016.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. In *International Conference on Learning Representations*, 2019.

Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python: analyzing text with the natural language toolkit*. "O'Reilly Media, Inc.", 2009.

Martin F Porter. An algorithm for suffix stripping. *Program*, 1980.

Chris Van Der Lee, Albert Gatt, Emiel Van Miltenburg, Sander Wubben, and Emiel Krahmer. Best practices for the human evaluation of automatically generated text. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 355–368, 2019.

Joseph L Fleiss and Jacob Cohen. The equivalence of weighted kappa and the intraclass correlation coefficient as measures of reliability. *Educational and psychological measurement*, 33(3):613–619, 1973.

Anthony J Viera, Joanne M Garrett, et al. Understanding interobserver agreement: the kappa statistic. *Fam med*, 37(5):360–363, 2005.

Xinyu Wang, Yong Jiang, Nguyen Bach, Tao Wang, Zhongqiang Huang, Fei Huang, and Kewei Tu. Improving named entity recognition by external context retrieving and cooperative learning. *arXiv preprint arXiv:2105.03654*, 2021.

Xu Han, Weilin Zhao, Ning Ding, Zhiyuan Liu, and Maosong Sun. Ptr: Prompt tuning with rules for text classification. *arXiv preprint arXiv:2105.11259*, 2021.

Urvashi Khandelwal, Angela Fan, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. Nearest neighbor machine translation. *arXiv preprint arXiv:2010.00710*, 2020.

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. Generalization through memorization: Nearest neighbor language models. *arXiv preprint arXiv:1911.00172*, 2019.

Appendix A

Annotation Guidelines

You will be provided with one dialogue between a user and a chatbot in an open-domain context, as well as four possible replies to the previous message. Your task is to compare the answers and evaluate their reply **adequacy** and whether they are expressing the **supposed sentiment**.

Some dialogues may present terms like person_46 or organization_9. You are not supposed to have any knowledge of these terms and should only consider their broad meaning for your evaluation. In the end of this document you can find an example of what you will be asked to evaluate. When considering the context, the grey balloons represent the user, while the blue balloons represent the chatbot.

The evaluation has an estimated duration of **40 to 60 minutes**.

During this evaluation you will be asked the same two questions per example displayed:

Do the replies represent the "x" emotion? In this question you will be provided with an emotion and are asked to consider all replies and answer whether or not they represent the desired emotion. In the end of this document you can find a description of each sentiment. **Please notice that the same expression could be used to express different emotions depending on the tone given to it.** E.g., "What?" could be used to express anger or surprise depending on the tone.

Do the replies sound appropriate considering the context of the dialogue? This question aims to evaluate if the reply is **appropriate** considering the provided context. To answer it, you must carefully analyse the context and the replies, and answer whether they are appropriate or not. In the example below you can find examples of inappropriate and appropriate sentences given a dialogue. **The reply options are the possible answers to the last sentence in the dialogue.**

Please notice that all metrics are independent and whether the chatbot sounds adequate should not influence the sentiment evaluation, and vice-versa. Furthermore, all possible replies are independent of each other, i.e., multiple options can be marked *Yes* or *No*.

Please look through these instructions carefully and feel free to reference back to them throughout the evaluation.

Sentiment

Anger: This sentiment is expressed by being frustrated/upset. Some expressions/words that represent it are: “*hate*”, “*is frustrating*”, etc.

Disgust: This sentiment is expressed by being repulsed/annoyed. Some expressions/words that represent it are: “*gross*”, “*disgusting*”, “*so annoying*”, etc.

Fear: This sentiment is expressed by being scared/fearful. Some expressions/words that represent it are: “*scared*”, “*acting desperate*”, etc.

Joy: This sentiment is expressed by happiness/enjoyment/gratefulness. Some expressions/words that represent it are: “*love you*”, “*awesome*”, etc

Sadness: This sentiment is expressed by sorrow/loss/being apologetic. Some expressions/words that represent it are: “*miss you*”, “*I’m sorry*”, etc.

Surprise: This sentiment is expressed by reacting to something unexpected. This can be positive or negative. Some expressions/words that represent it are: “*wait what?*”, “*holy*”, “*oh really?*”, etc

Appendix B

Generation Examples

	Example 1
Context	I can't believe the bus is late again today.
Baseline	It's just about time.
Anger	Why? What's wrong?
Disgust	I am sick of this bus. The traffic is so bad. Every time I get on the bus I feel ill.
Fear	Why?
Joy	Yes, I'm glad I got off the bus. I thought it was going to be late.
Sadness	I am sorry to hear that. But it's been a long time since we last saw each other. If you have any suggestions, I would like to see them. It's very kind of you to give me some information. We're always here for you.
Surprise	Why? Are you serious?
	Example 2
Context	I got a new job.
Baseline	What do you think about it?
Anger	Why are you angry?
Disgust	You are not a good employee. You have to work hard. You should give the boss a chance.
Fear	I think it's very good. But the problem is, I can't sleep well. But I feel very tired. So I have to go to the office. It's very stressful. But I feel very good.
Joy	You are a real winner.
Sadness	I am sorry. But my work is not easy.
Surprise	I am so happy.

Table B.1: Examples of sentences generated by the sentiment-conditioned text generation model using the pre-defined set of sentences, fine-tuned on the DailyDialog corpus, given the same context. The words in red represent words that are present in the pre-defined set of sentences.

	Example 1 - Anger
Context	The girl told her to collect it from neville apt. Smart decision.
Baseline	That's what she said.
Sentiment Baseline	I'm in the same situation. It's a great decision and you should feel proud.
Sentiment Sentences	I feel like you should be mad at her for having to collect it.
Full System	Ok, thanks for the offer.
	Example 2 - Disgust
Context	What? How so? link_6.
Baseline	lol.
Sentiment Baseline	Hmm I am not sure how to do that. I'm not sure how to.
Sentiment Sentences	I'm so confused.
Full System	I just want to get it up and running I don't think I have a clue.
	Example 3 - Fear
Context	person_53... eheh. I'm not, sorry. What's up?
Baseline	I'm just asking for the link of the person_54. A pdf. I don't want to link it, just the text.
Sentiment Baseline	heh heh heh hehe.
Sentiment Sentences	I'm scared.
Full System	Oh yeah.
	Example 4 - Joy
Context	I'm aware. Me too! I'm so happy for person_9!
Baseline	Ok I'll be here soon.
Sentiment Baseline	lol.
Sentiment Sentences	I'm so glad you're happy.
Full System	Yeah that's what I was thinking of the other time I talked to him.
	Example 5 - Sadness
Context	Ahh I'm talking to my prof. Give me a min I'll try to run. I gotta go I'm sorry I'm late for class
Baseline	I think I can hear you.
Sentiment Baseline	I'm sorry.
Sentiment Sentences	I'm sorry.
Full System	Ohhh ok I'll try to find you. I'm in class right now. You're not in class?
	Example 6 - Surprise
Context	Oh so the entire staff??! wowwww In AAA maybe 20%
Baseline	oh no what?
Sentiment Baseline	And the whole staff?
Sentiment Sentences	Oh wow, that's crazy.
Full System	So the whole organization is just a bunch of guys with a bunch of different people.

Table B.2: Examples of sentences generated by the Baseline (DialoGPT-small), Sentiment Baseline (DialoGPT-small + sentiment tag), Sentiment Sentences (DialoGPT-small + pre-defined set of sentences), and Full System (proposed sentiment-aware conversational agent, with the reply sentiment prediction model and the sentiment sentences model), fine-tuned on the EmotionPush corpus. For the same context from the test set, it is displayed the generated reply by each of the aforementioned models.

	Example 1 - Anger
Context	OK, I promise. What is it? It's a kitty.
Baseline	What is it?
Sentiment Baseline	I'm sorry, but i can't accept it.
Sentiment Sentences	It's hideous.
Full System	Is it white?
	Example 2 - Disgust
Context	Now what are you doing? Putting on mascara.
Baseline	What's that?
Sentiment Baseline	What a terrible idea!
Sentiment Sentences	That's disgusting!
Full System	I'm going to have a look.
	Example 3 - Fear
Context	Tired and stressed. This wedding is giving me a headache. Just relax. With me as your maid of honor, everything will be fine!
Baseline	I hope so. I will take you on a date.
Sentiment Baseline	I know, but I'm just not in the mood.
Sentiment Sentences	I am terrified. I have to go to the doctor. I'm scared.
Full System	You're right. I will not forget this.
	Example 4 - Joy
Context	Great, Mr. Sandals. Your reservation is confirmed. Now all I need is your phone number. Of course! It's area code 626-555-1739.
Baseline	I'm sorry, but we have to cancel your reservation.
Sentiment Baseline	Thank you for calling, Mr. Sandals. You've been very helpful.
Sentiment Sentences	Thank you. That's very kind of you.
Full System	Thank you.
	Example 5 - Sadness
Context	Can I borrow this magazine from you? It's really interesting and I can't put it down.
Baseline	Sure. Just take it to the library.
Sentiment Baseline	Sorry, I'm afraid I can't. I've already lent it to someone else.
Sentiment Sentences	I'm sorry, but I can't. I don't have it anymore.
Full System	Sure. It's the best one. I think I'll buy it.
	Example 6 - Surprise
Context	Yes, do I have to work a whole year before I can take a vacation? Yes, after you've been here a year, you qualify for a three-day vacation.
Baseline	Is that right?
Sentiment Baseline	Really? That's great.
Sentiment Sentences	Really? I thought you had to take a three-week vacation every year.
Full System	What about for vacation?

Table B.3: Examples of sentences generated by the Baseline (DialoGPT-small), Sentiment Baseline (DialoGPT-small + sentiment tag), Sentiment Sentences (DialoGPT-small + pre-defined set of sentences), and Full System (proposed sentiment-aware conversational agent, with the reply sentiment prediction model and the sentiment sentences model), fine-tuned on the DailyDialog corpus. For the same context from the test set, it is displayed the generated reply by each of the aforementioned models.