TÉCNICO LISBOA

Towards a multi-agent dialogue system with contextual agents and tailored distractors

Leonor Machado Llansol

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisor: Prof. Maria Luísa Torres Ribeiro Marques da Silva Coheur

Examination Committee

Chairperson: Prof. Alberto Manuel Rodrigues da Silva
Supervisor: Prof. Maria Luísa Torres Ribeiro Marques da Silva Coheur
Member of the Committee: Prof. Ricardo Daniel Santos Faro Marques Ribeiro

January 2021

Acknowledgments

First, I would like to thank my parents for their love, support and encouragement, for teaching me that everything can be achieved with hard work and dedication, for giving me the foundations to who I am, and for all the opportunities.

To my advisor, Luísa, thank you for all your guidance, support and friendship through the last two years. To Professor Nuno Mamede, thank you for triggering my interest for Natural Language Processing, and thank you, Luísa, for giving me the privilege of working with you and deepening my interest for this area. I could not imagine this journey with anyone else.

To Miguel, thank you for your love, for embarrassing me and making me laugh, for always pushing me to work harder, even when I did not want to, and for being my constant support.

To Mariana and João, thank you for your support and for the most enjoyable meetings, which made Friday afternoons the most fun time of the week. It was a pleasure to support you through your thesis, and an honor to take both of your works to build mine.

To Vânia, Pedro, Hugo, Gonçalo and Ricardo, thank you for all your help, and for always being available to enlighten me.

To the rest of my family and friends, who always unconditionally supported me and believed in me, thank you.

I would also like to thank my Math tutor, António, for always believing in me, even when I did not, for making me believe in myself and my abilities, and for recovering my passion for mathematics. Without you, I would not have realized that this was my path, and, for that, I will always be grateful to you.

To the Fundação para Ciência e Tecnologia (FCT), for supporting this thesis through two research grants: through the INCoDe 2030 initiative, in the scope of the demonstration project AIA, “Apoio Inteligente a empreendedores (chatbots)”, and through MAIA (“MAIA: Multilingual AI Agent Assistants”), a collaborative research project funded by the Agência Nacional de Inovação (ANI), the FCT, and the CMU-Portugal partnership program. This research is filed under the P2020 program, under contract number 045909.

To each and every one of you – thank you.

Abstract

In this work, we combine two in-house dialogue multi-agent systems into a single (multi-agents) system. However, the latter does not handle linguistic phenomena as synonyms, and acronyms; in addition it does not handle context. Thus, we propose a retrieval-based agent that handles the first two – the General Agent – and an end-to-end dialogue system that takes context into consideration in order to perform response selection – the Contextual Agent. When training an end-to-end model to perform this task, most systems take advantage of a possible answer (gold reply) and one or more not possible answers (the distractors). The latter are randomly selected from the corpus, despite the fact that, in a real scenario, possible response candidates are usually similar to the gold reply. Therefore, in this work, we introduce the concept of tailored distractors, corresponding to different methods of selecting distractors that are closer to the gold reply. We show that these distractors have a positive impact in the response selection task, but also if we consider a generative dialogue system. We use the method that yields best results to fine-tune a model for response selection, used by our Contextual Agent.

Keywords

Response selection; Distractor selection; Multi-agent dialogue system;

Resumo

Neste trabalho, combinamos dois sistemas multi-agente de diálogo num único sistema (multi-agente). No entanto, o último não trata fenómenos linguísticos como sinónimos e acrónimos; além disso, não considera o contexto. Assim, propomos um agente baseado em recuperação que trata dos dois primeiros – o Agente Geral – e um sistema de diálogo que tem o contexto em conta para selecionar a resposta – o Agente Contextual. Ao treinar um modelo para realizar essa tarefa, a maioria dos sistemas tira proveito de uma resposta possível (resposta *gold*) e uma ou mais respostas impossíveis (os distratores). Estas últimas são selecionadas aleatoriamente do corpus, apesar do facto de que, num cenário real, as possíveis respostas candidatas geralmente são semelhantes à resposta *gold*. Portanto, neste trabalho, apresentamos o conceito de distratores sob medida, correspondendo a diferentes métodos de seleção de distratores que estão mais próximos da resposta *gold*. Mostramos que esses distratores têm um impacto positivo na tarefa de seleção de respostas, mas também se considerarmos um sistema de diálogo gerador. Usamos o método que produz os melhores resultados para ajustar um modelo para seleção de resposta, usado pelo nosso Agente Contextual.

Palavras Chave

Seleção de resposta; Seleção de distrator; Sistema de diálogo multi-agente;

Contents

List of Figures

List of Tables

List of Algorithms

Listings

Acronyms

NLP	Natural Language Processing
LSTM	Long-Short Term Memory
BiLSTM	Bidirectional Long-Short Term Memory (LSTM)
RNN	Recurrent Neural Networks
CNN	Convolutional Neural Networks
EWAF	Exponentially Weighted Average Forecaster
GRU	Gated Recurrent Unit
MLM	Masked Language Model
NSP	Next Sentence Prediction

1

Introduction

Contents

1.1 Motivation

Chatbots have been getting a great deal of attention lately, in a time when Natural Language Processing (NLP) is developing faster than ever. They are a type of dialogue systems, or conversational agents, designed to have extended conversations with the user, having a similar behavior to human interaction (Jurafsky and H. Martin, 2019).

In this context, two recent works from INESC-ID developed multi-agent systems (Santos, 2019) (Fernandes, 2019), both centered in the idea that all agents can potentially answer all questions. Here, an agent is an entity that receives one query and returns one or more responses. In this thesis, our first objective is to integrate those systems into a single multi-agent system.

However, the integrated system does not solve two limitations. The first is that none of the agents allowed to use paraphrases, synonyms and domain specific synonyms and acronyms, among others. Thus, our second objective is to propose a new agent architecture – the General Agent. The second limitation is that none of the agents takes the context of the conversation, defined by Sankar et al. (2019) as the **dialog history between the user and the agent**, into account. Just like in a human conversation, it is crucial to use the context in order to have plausible and high quality responses. The third objective of this thesis is to develop the Contextual Agent, that takes the context into account when selecting a response.

To select a response using the context, we fine-tune a state-of-the-art classifier to classify if a sentence follows a certain context. While fine-tuning, the model needs both positive and negative training examples. The latter are usually randomly selected (Lowe et al., 2015). However, in retrieval-based systems, a search engine is used to retrieve a number of candidates, from which the model selects a response. Thus, the candidates already have some degree of similarity between them, as proven by preliminary results that show that candidates retrieved by a search engine are, on average, *two times more similar* than randomly retrieved ones.

Therefore, training a model with random distractors may not be the best choice, when, in a real-world scenario, the model will have to distinguish a correct answer among a set of strong contenders. Thus, in this thesis, we also introduce the notion of **tailored** distractors, and study different ways of selecting them, rather than choosing them randomly, and study the impact of each method on response selection and generation systems.

Khandelwal et al. (2018) and Sankar et al. (2019) showed that current dialog systems do not properly use the context of the conversation, having a similar performance when the conversation history suffers perturbations, such as shuffling all sentences and words within each sentence. So, regardless of the context of the conversation, the bot gives the same answer. Taking their insights into account, we make a final experiment that studies if a more current language model is really using context when selecting an answer by introducing perturbations in it and observing if the system's performances changes.

1.2 Objectives

In the following we resume the main objectives of this thesis:

1. **Integrate two multi-agent systems** developed at INESC-ID (Fernandes, 2019) (Santos, 2019), and simplify the creation and addition of new agents to the system. Considering that part of this work will be developed as part of the project AIA – "Agente Inteligente para o Atendimento no Balcão do Empreendedor", the resulting system will work for the **Portuguese** language.
2. Develop an **agent architecture** with additional functionalities.
3. Develop an agent that uses the **context** of a conversation to select a response.
4. Study the hypothesis that selecting **tailored distractors** improves the performance of a retrieval and generative model.
5. Perform an ablation study to assess if a modern language model takes the context into account.

1.3 Contributions

The contributions of this thesis are the following:

- Integrate two multi-agent dialogue systems;
- Propose the General Agent architecture;
- Instantiate the General Agent with two different corpora:
  - With an AMA corpus, obtaining the AMA Agent (project AIA);
  - With a COVID corpus, obtaining the COVID Agent;
- Create the Contextual Agent;
- Introduce three ways of selecting tailored distractors;
- Evaluate our tailored distractors on a customer support dataset (project MAIA);
- Show that a current neural dialogue system is using the context of the conversation.

The resulting multi-agent dialogue system, containing the General and Contextual agents, can be consulted at `https://github.com/leonorllansol/multi-agent-dialogue`.

1.4 Organization of the Document

This thesis is organized as follows: Section 2 describes concepts and systems used in the development of this thesis. Section 3 studies works related to this thesis, particularly focused on the representation and use of context in dialogue systems. Section 4 describes the integration of the two multi-agent systems, and the creation and addition of new agents to it. Section 5 describes the introduced methods to select tailored distractors. Section 6 shows how to easily integrate new agents into the multi-agent system, how the impact of tailored distractors was evaluated, and the ablation study performed to assess if context is taken into account in a modern language model. Finally, Section 7 points the conclusions of our work and suggestions for future work.

2

Background

Contents

In this Section, we describe existing systems that were used on the development of this thesis. Particularly, we study how neural networks evolved until the Transformer architecture (Section 2.1) and explain the concept of Transfer Learning (Section 2.2), for a better understanding of the language models we will use (Section 2.3). Finally, we describe the in-house developed systems that will be used in this thesis (Section 2.4).

2.1 From RNNs to Transformers

A few years ago, the state-of-the-art approaches for language problems, such as machine translation and speech recognition, were **Recurrent Neural Networks (RNN)** (Anderson and Rosenfeld, 1988), which are neural networks with memory (they can remember past inputs), where the relation between input words is taken into account due to its sequential nature, encoding the whole sentence first and then decoding it. While they have many advantages over basic neural networks, due to their ability of persisting information, they have some problems, in particular: they fail at modeling long-range dependencies, since their hidden states contain information about the whole sentence and the context vector has a fixed size, which difficults the storage of input information for longer inputs; and, due to their sequential nature, they do not allow for parallelization, which leads to performance issues for longer sequence lengths.

A solution to model long-range dependencies is a **Long-Short Term Memory (LSTM)** (Hochreiter and Schmidhuber, 1997) which is a particular type of RNN whose basic unit is composed of a cell and three gates: input, forget and output gate. The forget gate allows the LSTM to forget things which are not important, and the input and output gates moderate what goes in and out of each cell. This solves the RNN problem of modeling long-range dependencies. However, it still does not perform well when the input sentence is long, since it still has a sequential nature.

A first solution to the problem of non parallelization is **Convolutional Neural Networks (CNN)**. They parallelize – each word on the input can be processed at the same time, not depending on other input words to be processed. However, they do not model dependencies between words, which is the problem that the Transformer aims to solve. The **Transformer** (Vaswani et al., 2017) is a transduction model which avoids recurrence by relying entirely on an attention mechanism, with the goal of drawing global dependencies between input and output. Without the sequential nature of a RNN, it allows for parallelization, reducing the time necessary to train the model, and reaches state-of-the-art results for translation tasks.

To understand the Transformer architecture, one must first understand the **attention** mechanism. Similar to human attention, where the mechanism gets its name, it allows the decoder to go back and focus on particular parts of the input.

The Transformer relies on a particular type of attention – **self-attention** – where a sentence attends to itself, in order to compute a representation of the sentence. Its architecture consists on an encoder and a decoder, each with N layers (6 in the original model). The layers of the encoder have two sublayers: a multi-head self-attention mechanism and a feed-forward neural network. The multi-head self-attention mechanism consists on *h* layers of self-attention running in parallel. The model was tested on machine translation (English to German and English to French) and outperformed state-of-the-art models, further showing that training was faster. It is now the main reference in machine translation and other language problems.

Since neural approaches will be used in this work, and being the Transformer architecture the main neural approach today, and the basis of BERT and GPT-2 that will be used in this work and are described in Sections 2.3 and 2.4, this section introduced the concepts of Transformer and attention, which are important to understand some of the systems we will study in section Related Work.

2.2 Transfer Learning

One of the main problems of machine learning is gathering enough data to train a model – *insufficient training data*. This problem is aggravated in current deep learning models, which need even more training data than traditional machine learning models (Tan et al., 2018).

The concept of **transfer learning** comes from the idea that, if a person generalizes her experience, she can apply it in different domains. For example, someone with experience in playing the violin can learn how to play the piano faster than those without experience with musical instrument, by generalizing his violin knowledge and applying it to the piano (Zhuang et al., 2019). Thus, transfer learning solves the aforementioned problem by using the same knowledge across domains, allowing a model to be **pre-trained** on data from a source domain (more general) and then **fine-tuned** on data from a target domain (more specific). Hence, a big amount of data is needed to pre-train the model, but, to train a model on a specific task, only a small amount is needed.

2.3 Language models: BERT and GPT-2

Language modeling consists on assigning a probability to a sequence of words. Pre-training language models and applying them to specific tasks can be done through two approaches: feature-based, where task-specific architectures are used as additional features, or fine-tuning, where the model is trained on the specific task by fine-tuning all the pre-trained parameters.

Previous language models assigned a probability to a word, considering the words to its left. **BERT** (Bidirectional Encoder Representations From Transformers) (Devlin et al., 2019) obtained state-of-the-

art results in the task of language modeling, having a masked language modeling objective that does it bidirectionally, that is, looking at a sentence both from left to right and from right to left. It is a fine-tuning approach and an example of transfer learning: it is **pre-trained** on a large amount of data and can be **fine-tuned** to specific tasks, using less data. It introduces two tasks to the pre-training:

- Masked Language Model (MLM) – random tokens are masked, using special token [MASK], and then predicted by the model, using both the left and right context.

- Next Sentence Prediction (NSP) – model learns relationships between two sentences: when training, given two sentences `A` and `B`, 50% of the time `B` is the actual sentence that follows `A`, and is labeled as so (`IsNext`), and 50% of the time it is a random sentence from the corpus (`NotNext`).

While the Transformer has encoders and decoders, BERT only encodes a sentence, being composed of Transformer *encoder* layers. So, when a sentence is given as input to BERT, the output is a representation of that sentence, which consists of the word embedding of each token, plus an embedding for an extra token, [CLS], which represents the whole sentence and is used for sentence classification purposes.

GPT-2 (Radford et al., 2019) is also a language model, and a fine-tuning approach. It uses the auto-regression mechanism, as RNNs do, using the output of the previous layer as input of the next layer. While BERT takes a sentence as input and returns a vector representation of it, GPT-2's output is a complete sentence. Unlike BERT, it is only composed of Transformer *decoder* layers: it uses the context to its left to predict the next word.

While pre-training these models requires a large amount of data, fine-tuning them only requires a small amount of task specific data, which we will take advantage of, as we will see further.

This Section introduced two models based on which we develop this work, since we will use BERT For Next Sentence Prediction, which is a BERT model with a Next Sentence Prediction layer on top, and DialoGPT-2 (Zhang et al., 2019c), a model fine-tuned from GPT-2 for dialog.

2.4 In-house tools

Here, we describe three in-house tools that will be used. Two of them are multi-agent systems that were developed as Master thesis at INESC-ID (Section 2.4.1), and the other is a GPT-2-based ranking and generative framework which was developed under the scope of project MAIA: Multilingual AI Agent Assistants[1], whose goal is to develop a platform where AI agents perform customer support (Section 2.4.2).

[1] https://resources.unbabel.com/maia-unbabel-research (Last accessed on: 23/11/2020)

2.4.1 Multi-agent systems

On most dialogue systems, there is either a single agent behind each virtual assistant or multiple agents working separately, with, for example, each being delegated a different type of question. However, it may be useful to have multiple agents working together, which is the motivation for multi-agent systems.

As previously said, this work will be built upon two multi-agent frameworks developed as Master thesis at INESC-ID: "Chattuga – A meta-chatbot for the Portuguese language" (Fernandes, 2019) and "Say Something Smart 3.0: A Multi-Agent Chatbot in Open Domain" (Santos, 2019), from now on, MULTI-SSS. Both of these works focus on developing plug and play architectures to integrate conversational agents, and both take the same assumption: all agents can potentially answer all questions. This is because agents, like humans, have strengths and weaknesses, so it may be useful for them to work together.

These systems have two main components: **agents**, that given one query, select one or more responses from a set of candidate responses that can be either retrieved by a search engine, such as Whoosh[2], when the corpus is large enough, or locally retrieved by each agent; and **decision making strategies**, that given a set of responses, return one of them, according to their heuristic. They also have a component responsible for managing the conversation and interaction between the system and the agents and strategies, which was called *Coordinator* in MULTI-SSS, but which we will call the *Dialogue Manager* (Section 3.1).

[2]https://whoosh.readthedocs.io/en/latest/intro.html (Last Accessed on 28/01/2021)

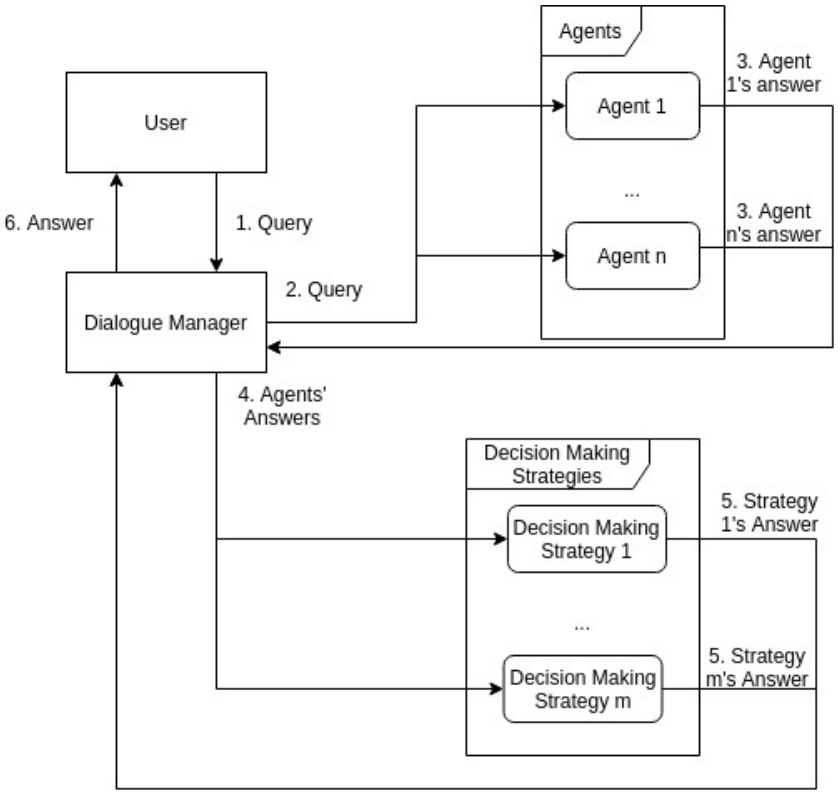

Figure 2.1: CHATTUGA's architecture

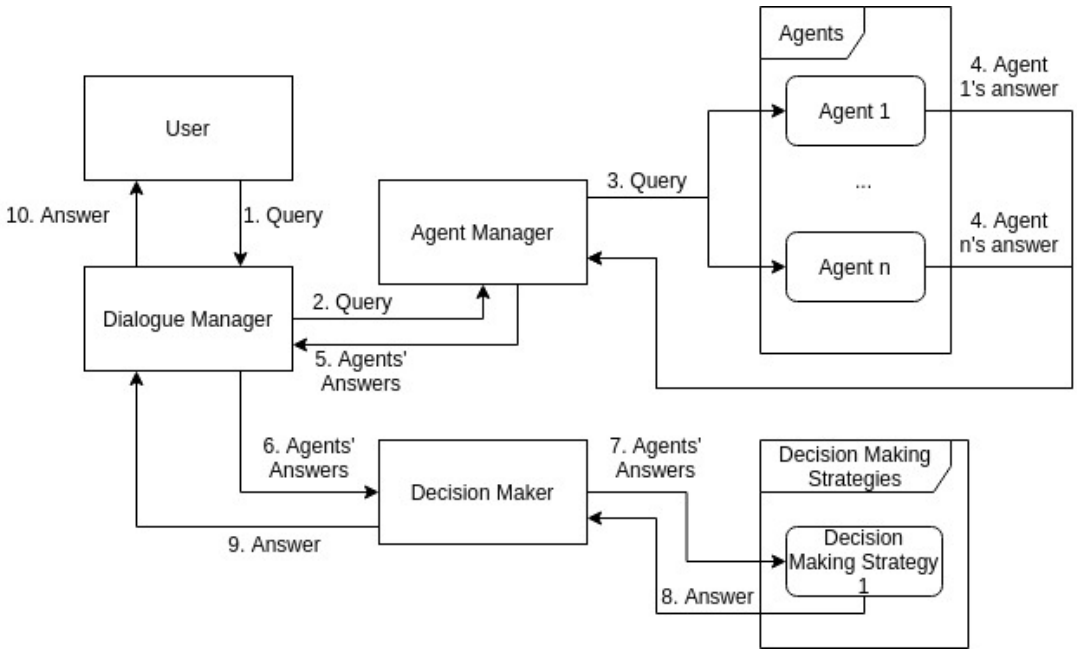

Figure 2.2: MULTI-SSS's architecture

Both systems architectures, in what regards agents and decision making strategies, are shown in

Figures 2.1 and 2.2.

Chattuga's *Dialogue Manager* has a classifier that, upon receiving a user query, classifies it using a set of labels, such as the type of the query, i.e. if it is a question and, if so, the type of the question, such as WHAT question, WHO question, or WHEN question). Regarding the **agents** used in CHATTUGA, already existing agents were integrated into a multi-agent system:

- Talkpedia (Mota, 2015) – uses Wikipedia to retrieve an answer.
- SSS (Ameixa, 2015) – uses data from movie subtitles to retrieve an answer.
- Say Something Smart AMA – uses SSS, using data from a corpus built with Agência para a Modernização Administrativa (AMA)[3] instead of movie subtitles.
- Edgar (Fialho et al., 2013) – specialist in answering questions about the Monserrate Palace
- Cheat – rule-based agent that only answers salutations, yes no questions and or questions.

In MULTI-SSS (Santos, 2019), the above mentioned agent SSS was turned into a multi-agent system, using, for example, similarity measures as agents, as the Cosine Similarity and the Levenshtein Distance. It also has an Edgar agent, as CHATTUGA. Note that none of the mentioned agents handles synonyms or acronyms, or uses context to select a response.

Regarding the **decision making strategies**, in CHATTUGA there are four strategies, all based in classification modules:

- Query Agent – the query is classified and its labels are compared with each agent's area of expertise, giving each agent a score; the answer given by the agent with highest matching is returned.
- Query Answer – the query is classified, as are the answers given by each agent. The query's labels are compared to each answer's labels, and the answer with highest matching is returned.
- Answer Impersonal – each answer is scored using an impersonal answers module that compares it to the query, when the query is labelled as IMPERSONAL.
- Answer Personal – each answer is scored using a personal answers module that compares it to the query, when the query is labelled as PERSONAL.

Each strategy returns one answer, and the final answer to return to the user is chosen by the *Dialogue Manager*, based on the weights of each strategy.

In MULTI-SSS, there are two decision making strategies available:

- Voting Model – the most common answer is returned.

[3] https://www.ama.gov.pt/ (Last accessed on 28/08/2020).

- Priority System – an answer is prioritized according to a predefined score given to each agent, so the answer returned will be that of the agent with the highest score. If this agent does not return an answer, the returned answer is chosen using the Voting Model over the answers given by the remaining agents.

Unlike in CHATTUGA, where all strategies can be available, in MULTI-SSS only one strategy is available, so the answer returned by the active strategy is returned to the user.

MULTI-SSS has two additional modules – the **Agent Manager**, which is an interface point between the system and the agents and is responsible for initializing the agents, and the **Decision Maker**, that bridges the system and the decision making strategies.

To better understand these systems' flow, we present a running example each system, whose steps match those in Figure 2.1, for CHATTUGA, and 2.2, for MULTI-SSS:

- CHATTUGA

  1. The user poses the query *q* to DM: "Quantos anos tens?" ("How old are you?")
  2. The DM classifies *q* with labels QUESTION, WH_QUESTION, PERSONAL, OTHER, NUM, COUNT and AGENT_LIFE, and forwards q to all active agents: Edgar, Talkpedia and Cheat
  3. The agents return their selected responses to the DM:
     - a_{Cheat}: "Essa não é a minha área de especialidade." ("This is not my area of expertise.")
     - a_{Edgar}: "Tenho 65 anos." ("I'm 65 years old.")
     - $a_{Talkpedia}$: "Não sou um grande perito nesse tema, mas sobre isso sei que em agosto de 2019, os Wet Bed Gang actuaram no festival de música MEO Sudoeste." ("I am not a great expert on this subject, but I know that in August 2019, Wet Bed Gang performed at the MEO Sudoeste music festival.")
  4. The DM forwards the agents' answers, [a_{Cheat}, a_{Edgar}, $a_{Talkpedia}$], to the active Decision Making Strategies (query agent, answer impersonal and query answer, all with the same weight), except for the answer impersonal one, since this one is only chosen when IMPERSONAL is in the query labels.
  5. Each of the decision making strategies selects one of the agents' answers and returns it to the DM:
     - query agent – a_{Edgar}, "Tenho 65 anos." ("I'm 65 years old.")
     - query answer – a_{Edgar}, "Tenho 65 anos." ("I'm 65 years old.")
  6. Since all decision making strategies returned the same answer, a_{Edgar}, the DM returns that answer to the user: "Tenho 65 anos." ("I'm 65 years old.")

- MULTI-SSS

  1. The user poses the query *q* to DM: “Quantos anos tens?” (“How old are you?”)
  2. The DM forwards *q* to the Agent Manager.
  3. The Agent Manager sends *q* to each active agent: Cosine, Edgar and Levenshtein.
  4. The agents return their selected responses to the Agent manager:
     - a_{Cosine}: “Ela é do ano do galo.” (“She is from the year of the rooster.”)
     - a_{Edgar}: “Tenho 65 anos.” (“I’m 65 years old.”)
     - $a_{Levenshtein}$: “Trinta e quatro.” (“Thirty four.”)
  5. The Agent Manager returns the agents’ answers, [a_{Cosine}, a_{Edgar}, $a_{Levenshtein}$], to the DM.
  6. The DM sends the agents’ answers to the Decision Maker.
  7. The Decision Maker sends the agents’ answers to its active decision making strategy, Simple Majority.
  8. Since all the answers are different, the Simple Majority returns the first one to the Decision Maker, a_{Cosine}.
  9. The Decision Maker sends the answer to the DM.
  10. The DM returns the selected answer, a_{Cosine}, to the user: “Ela é do ano do galo.” (“She is from the year of the rooster.”).

Besides turning SSS into a multi-agent system, Santos (2019) also developed a learning module to take user feedback into account (**online learning**). The user evaluates the response, which leads to the weight update of the agent’s features (text similarity with input, response frequency, answer similarity with input). This approach was proposed by Mendonça et al. (2017) and uses the Exponentially Weighted Average Forecaster (EWAF) algorithm to update the weights.

Neither of the aforementioned works use context when selecting an answer, which is why, in this work, they will be integrated into a single multi-agent system and a new agent that uses context when selecting a response will be created, along with a new agent that will allow instances with different corpora and the use of user-defined synonyms and acronyms.

2.4.2 DialoGP3T

To study the impact of tailored distractors, we use **DialoGP3T**, a model[4] developed under the scope of project MAIA, which is an adaptation of TransferTransfo (Wolf et al., 2019). It is built on top of a

[4]https://github.com/huggingface/transfer-learning-conv-ai (Last accessed on: 19/10/2020)

Listing 2.1: Persona example

```
'personality'=[
                "i like to remodel homes .",
                "i like to go hunting .",
                "i like to shoot a bow .",
                "my favorite holiday is halloween ."
              ]
```

pre-trained GPT-2 model. Besides taking the context of the conversation into account, it also considers another type of context: the *persona* of the agent, which is a set of sentences describing the agent.

While most models try to minimize a single loss, this model is trained on a multi-task setting with two goals:

- minimize the *language modeling* loss, in order to generate plausible responses.
- minimize the *next sentence prediction* loss, in order to correctly classify a gold response among a set of *random* distractors.

At each iteration, the model loss is computed as a weighted sum of both losses, where the weights assigned to each one are 2 to the first and 1 to the last, and can be changed in a configuration file.

Therefore, this model is both retrieval and generative. As the name suggests, the model is a transfer learning approach, where its training consists of pre-training it and fine-tuning it for a specific task.

As fine-tuning data, it uses the *PersonaChat* dataset[5], which contains 17898 entries, where each entry contains a *persona*, and a set of utterances, with each containing a set of candidates, where the last one is the gold reply and the others are distractors, and a conversation history. An example of a persona from the *PersonaChat* dataset is shown in Listing 2.1.

Other corpora can be used to fine-tune the model, as long as they have the same *json* structure as the *PersonaChat*.

For our experiments to study the impact of our tailored distractors in retrieval and generative models, we will use DialoGP3T, since its multi-task training setting allows it to do both. We use the *PersonaChat* and the pre-trained model `microsoft/DialoGPT-small`[6] (Zhang et al., 2019c). It is a neural model for response generation, trained on top of GPT-2 using Reddit dialogue data. To solve the problem of uninformative responses, that can answer many questions, they implement a maximum mutual information (MMI) scoring function, which is trained to predict a query (*source*), given a response (*target*), to maximize the probability of generating a response that uniquely answers that query, and is, thus, more informative.

[5]https://s3.amazonaws.com/datasets.huggingface.co/personachat/personachat_self_original.json (Last accessed on: 19/10/2020)

[6]https://huggingface.co/microsoft/DialoGPT-small(Last accessed on: 27/11/2020)

3

Related Work

Contents

In this thesis, we join two multi-agent dialogue systems, and add a context agent to the resulting system. In this section, we start by studying current dialogue systems' architectures (Section 3.1). We also study how context is represented and used in current dialogue systems (Section 3.2). Then, we introduce the task of Next Sentence Prediction, and study recent systems which use it (Section 3.3). That leads us to the task of distractor selection, which we explain and see in which tasks it is used, and how (Section 3.4). Finally, we explore how response selection systems and the usage of context are currently evaluated (Section 3.5).

3.1 Dialogue Systems Architecture

Our first objective is to join two multi-agent systems, each with different components. To better accomplish it, and try to standardize our system as much as possible, we present a brief study of the components that make a dialogue system.

A multi-agent dialogue system is a system composed of multiple agents. Each agent receives a query and returns a response, whether retrieved or generated. Most of these systems' agents are cooperative, working together to achieve a certain goal. Fum et al. (1988) motivated a distributed architecture where a set of autonomous agents cooperated to solve a NLP assignment. More recently, Papangelis et al. (2019) train two conversational agents that learn by interacting with each other. In some systems, there are multiple agents available, as in ours, but each of them focuses on a particular skill, such as Facts, Movies, and Thoughts. These are usually called *miniskills*, and, based on the dialogue state and user's intent, *only one* of them is selected to answer each user's request (Fang et al., 2018) (Chen et al., 2018). Others, as MACA (Truong et al., 2017), are frameworks that allow to easily integrate different agents, as in ours, but only allow to have one agent implemented at a time, while, in our system, multiple agents can be active and respond simultaneously.

Regarding dialogue systems' architectures, the main components are:

1. **Natural Language Understanding** (NLU) – preprocessing operations are applied to the user input, such as POS tagging, stopwords removal (Truong et al., 2017), intention, sentiment and topic extraction (Fang et al., 2018), dialogue act extraction and coreference resolution (Chen et al., 2018).

2. **Dialogue Manager** (DM) – manages the conversation, using the user's intent and the dialogue history to redirect the user's request to one of multiple miniskills (Fang et al., 2018) (Chen et al., 2018), implemented in this component, or redirect to a single agent, when there is only one (Truong et al., 2017).

3. **Natural Language Generation** (NLG) – applies postprocessing operations to the DM's output,

whether to make it more readable and natural (Truong et al., 2017), or to create a sentence from its output (Fang et al., 2018), possibly using a template of natural responses (Chen et al., 2018).

Spoken dialogue systems have two additional components: **Automatic Speech Recognition** (ASR) before the NLU, that converts speech to text, and **Text To Speech** (TTS) after the NLG, that converts text to speech.

Although we want to standardize our system so it has the aforementioned components, we do not perform any preprocessing to the user request nor postprocessing before returning the response to the user, as it is not in the objectives of this thesis, thus we will not have a NLU nor a NLG modules, only a DM.

3.2 Using context in dialogue systems

As previously seen, we consider the **context** to be **dialogue history between the user and the agent** (Sankar et al., 2019). Uses of context include:

- Techniques that change the context:
  - **Anaphora resolution**, which consists on determining the antecedent of an anaphor (Mitkov, 2007). To determine the correct antecedent, it is necessary to search for candidates in the context.
  - **Ellipsis resolution**, which consists on finding an antecedent in the context for an ellipsis, which is a phenomenon where words are omitted from the text (Zhang et al., 2019b).
  - **Question-in-context rewriting** introduced by Elgohary et al. (2019): given the context, rewrite a context-dependent question into a self-contained question with the same answer, solving language understanding elements, such as coreferences and ellipsis. A context-dependent question depends on the context of the conversation to be understood. It can have references to previous entities, or ellipsis that can only be resolved if the context is considered. A self-contained question does not need context to be understood; pronouns are replaced by the corresponding antecedent. In the following example, we show a context-dependent question and its rewritten self-contained question (Elgohary et al., 2019):

    Context-dependent: Did *they* marry?

    Rewritten: Did *Hannah Arendt and Heidegger* marry?

- Techniques that use the context:
  - **Dialogue Act Prediction** of the next response, such as “statement” and “question” (Kumar et al., 2018) (Kumar et al., 2019) (Tanaka et al., 2019). Kumar et al. (2018) showed that

dialogue acts information improves the task of response selection, for both retrieval and generative-based models. The dialogue acts were given as information on test time, but Kumar et al. (2019) developed a multi-task system that does not depend on being given the dialogue act information, being able to predict the dialogue acts and select a response. Although using dialogue acts to aid response selection is an interesting approach, it will not be done in this work since, to the best of our knowledge, there is no Portuguese corpus tagged with dialogue acts.

- **Question Generation** (Tuan et al., 2019) (Gao et al., 2019), which is usually performed considering just the answer sentence to which we want to generate a question. Tuan et al. (2019) showed that using context improves the quality of the generated questions. In this paper, context is modeled using a multi-stage attention mechanism, which captures the most relevant parts of the document that are related to the answer sentence, using this to generate the question.
- **Response Selection**, which consists on selecting a response from a set of candidates, considering the context of the conversation.

The last is a core task in retrieval-based chatbots. While early studies only consider the last utterance in the context to find an appropriate response (*single-turn response selection*) (Wang et al., 2013), it has recently been shown that considering more than one utterance of the context (*multi-turn response selection*) can improve the quality of the selected answer (Zhou et al., 2016), since human responses are based on the whole conversation at different granularities (words, phrases, sentences) rather than just on the last utterance.

Previous works consider context as a sequence of words, but Zhou et al. (2016) propose a multi-view approach for response selection, where one view considers context divided into words – *word sequence view* – and another view considers context divided into utterances – *utterance sequence view*. In the *word sequence view*, context and response are represented as low dimensional word embeddings, constructed by a Gated Recurrent Unit (GRU) (Chung et al., 2014), which is similar to an LSTM, as studied in Section 2.1, but without an output gate. In the *utterance sequence view*, utterance and response embeddings are built by a CNN. In each view, a confidence is calculated for each response, based on the similarity between the embeddings of the response and the context. The final score of each response is the sum of the two confidences, one per view, and the response with the highest score is selected.

Wu et al. (2017) state that representing the whole context as a vector, like in (Zhou et al., 2016), causes models to lose information. To avoid this, they propose a model – *Sequential Matching Network* – that matches a response with each utterance, encoding important information in a matching vector. Each utterance is represented by a word embedding, which is then fed to a GRU. All matching vectors

(one per utterance) are then accumulated to compute the final matching score between context and response.

Both the aforementioned systems use GRU, which are a type of RNN. Since RNNs are used for encoding text, which is too costly to use for capturing multi-grained semantic representations and have the disadvantages seen before, Zhou et al. (2018) propose a multi-turn response selection system using dependency information which is entirely based on the attention mechanism, inspired in the Transformer architecture. The attention mechanism is extended in two ways:

- self-attention – sentence attends to itself, allowing the capture of intra word-level dependencies.
- cross-attention – context and response attend to each other, to find which response has more in common with the context.

The proposed model is called DAM – *Deep Attention Matching network* – which aims to match a response with multi-turn context, by learning a matching model which measures the relevance between the context and a candidate response. The model selects the *k* best-matched responses from *n* available candidates for a given conversation context *c*. Although we initially meant to built this work from this model, we will see further that we do not use it and why.

Bapna et al. (2017) explore how to improve dialogue context modeling within a RNN-based spoken language understanding system. The performed experiments suggest that *encoding more context from the dialogue results in a reduction in overall frame error rate*, improving the quality of the given answers.

To use context, systems must represent it. This requires transforming the context's utterances into a vector representation. Table 3.1 resumes some current systems' ways of representing context.

Through its analysis, we conclude that context is used in many different tasks, and is represented using mostly BiLSTM, LSTM, RNN, GRU and word embeddings.

3.3 Next Sentence Prediction

In this thesis, one of the models used to develop our contextual agent and to assess the impact of our tailored distractors is fine-tuned from BERT. One of BERT's pre-training objectives is Next Sentence Prediction (NSP): given two sentences `A` and `B`, 50% of the time `B` is the actual sentence that follows `A`, and is labeled as so (`IsNext`), and 50% of the time it is a random sentence from the corpus (`NotNext`).

The Hugging Face[1] library has made available some interfaces built on top of BERT, that are available for fine-tuning for some NLP tasks, such as Question Answering (`BertForQuestionAnswering`[2]), Sen-

[1] https://huggingface.co/ (Last accessed on: 11/12/2020)
[2] https://huggingface.co/transformers/model_doc/bert.html#bertforquestionanswering (Last accessed on 04/12/2020)

Model	Task	Architecture used to create a context representation
Gao et al. (2019)	Question Generation	Bidirectional LSTM (BiLSTM)
Ma et al. (2019b)	Response Selection using a novel triple attention mechanism	Self-attention is used to create a vector representation of each utterance, which is fed into an LSTM layer
Tanaka et al. (2019)	Dialogue-Act Prediction	Two RNNs: one for encoding utterances and one which takes the output of each utterance encoder to generate a context vector
Zhou et al. (2018)	Response Selection using a novel model only based on attention	Each utterance is represented by word embeddings, which are then fed to an attentive module, that outputs the context representation
Elgohary et al. (2019)	Question-in-Context Rewriting	BiLSTM
Zhou et al. (2016)	Response Selection using a multi-view approach	GRU in word sequence view, CNN in utterance sequence view
Wu et al. (2017)	Response Selection using a sequential matching network	Word embeddings and GRU
Ohsugi et al. (2019)	Conversational Machine Comprehension	BERT encodes relation between context and current paragraph

Table 3.1: Ways of representing context

tence Classification (`BertForSequenceClassification`[3]) and NSP (`BertForNextSentencePrediction`[4]). The last corresponds to the NSP objective used in BERT's pre-training, and consists of a NSP head on top of BERT model.

`BertForNextSentencePrediction` always takes two sentences as input, `s1` and `s2`, and returns a vector [`p1`, `p2`], where `p1` is the probability of `s2` following `s1`, and `p2` is the probability of `s2` being a random sentence.

Although it is a recent model, `BertForNextSentencePrediction` model has already been used for various tasks:

- Wang et al. (2019) use it to help Reddit users find posts that are similar to theirs, by predicting

[3]https://huggingface.co/transformers/model_doc/bert.html#bertforsequenceclassification (Last accessed on 04/12/2020)
[4]https://huggingface.co/transformers/model_doc/bert.html#bertfornextsentenceprediction (Last accessed on 04/12/2020)

the probability of a post `B`'s text body following a post `A`'s title. Besides using the pre-trained `BertForNextSentencePrediction` model, they also fine-tune it: for each post, they assume that different posts from the same author and same subreddit are similar, thus they select one of those as a positive example (`IsNext`). As a negative example, they randomly select a post that is not from the same author (`NotNext`).

- Shi and Demberg (2019) study it for the task of Implicit Discourse Relation Classification, by comparing the performance of BERT pre-trained with the NSP objective and without it, and showing that the first has better results. This proves that the NSP task helps at modeling the semantic relations between two sentences.

- Kong et al. (2020) introduce the task of sentence level cloze completion, which consists on filling the blanks in a text with complete sentences. They use `BertForNextSentencePrediction` to predict the most appropriate candidate to each blank space, given its context.

As we will see in Section 4.2.3, `BertForNextSentencePrediction` will also be used in our work.

3.4 Distractor Selection

In the previous section, it was seen that the task of NSP includes fetching a random sentence from a corpus, as a negative example. In this section, we study the importance of selecting negative examples using some heuristic, instead of selecting them randomly, as one of the objectives of this thesis is to study the hypothesis that selecting tailored distractors improves the performance of a retrieval and generative model.

The task of **distractor selection** was created to aid in the creation of multiple choice questions (MCQ) from long texts. Mitkov and Ha (2003) introduced this task that uses NLP methods, such as term extraction, word sense disambiguation and WordNet (Miller, 1995), to generate questions and corresponding items. While one of the items is the correct answer, the others are **distractors**, which must be *semantically close* to the correct answer, so that finding the correct answer is less obvious for students. In this novel approach, the selection of distractors is done using WordNet. Through user evaluation, it was realized that, from all the tasks involved in MCQ generation, the task of distractor selection was the one that needed more improvement.

Mitkov et al. (2009) studied how to improve the quality of the selected distractors by testing different ways of calculating semantic similarity, but no method was found to outperform the others.

Mitkov and Ha (2003) created MCQs from long texts, but using only one sentence of the text for each question. Araki et al. (2016) was the novel system to create MCQs from multiple sentences, in a

way that requires the student to take inference steps, such as coreference resolution, to find the correct answer.

Another traditional task that has been automatized and uses distractors is Cloze (Taylor, 1953) (Jiang and Lee, 2017) (Gao et al., 2020), which is a test where parts of a text have been removed and the student must fill the gaps, choosing from a set of candidates that include the correct missing span and distractors.

The mentioned systems train their models using English exams designed by teachers.

The task of distractor selection for multiple choice questions usually consists on computing a metric that compares each distractor (*d*) to the correct answer (*c*). Namely, as mentioned, for the task of multiple choice questions, Mitkov and Ha (2003) compute the semantic similarity using the Wordnet, which retrieves hypernyms and hyponyms, to have *d* semantically close to *c*. Gao et al. (2020), to select distractors for the Cloze task, use the **length difference** between *c* and *d*, the **cosine similarity** between *c* and *d*, the **distractor frequency**, where *d* has highest score if it appears less, and the **frequency difference** between *c* and *d*. Jiang and Lee (2017), also for the Cloze task, compute a **semantic similarity** using word2vec (Mikolov et al., 2013), a **spelling similarity** and a **word co-ocurrence similarity**, assuming that sentences with common words or spelling are harder to distinguish by students.

On this work, we select distractors computing a semantic similarity with the correct response, among other methods, as we will see further.

Current dialogue systems, namely response selection systems, use a dialogue corpus, some of them the Ubuntu Dialogue Corpus (Lowe et al., 2015). To train their models, for each training example, they need the context of a conversation, and one positive and one (or more) negative examples. This negative example is, in most response selection systems, *randomly* sampled from the corpus (Lowe et al., 2015) (Gunasekara et al., 2019) (Wu et al., 2017) (Zhou et al., 2016) (Zhou et al., 2018) (Henderson et al., 2019) (Gu et al., 2019) (Ma et al., 2019a) (Yuan et al., 2019). (Zhang et al., 2017) propose a more sophisticated approach, where negative examples are randomly chosen from all other utterances *within the same document*, instead of randomly chosen from the whole corpus, so "distractors are likely from the same sub-conversation or even from the same sender but at different time steps". Devlin et al. (2019) also use random distractors in their NSP pre-training task, as previously mentioned in Sections 2.3 and 3.3.

Recent works have motivated the importance of selecting distractors instead of using random ones. Based on the assumption that, in real-world scenarios, models have to select a correct response from a set of strong distractors instead of random ones, this is, distractors that are harder to distinguish from the correct response than random ones, Lin et al. (2020) propose the creation of a grayscale dataset to train response selection systems: instead of considering the ground-truth response the `correct` response and all the distractors as `incorrect`, they use a multi-level ranking, where the ground-truth response

is *white*, randomly sampled utterances are *black*, and utterances obtained using retrieval or generative systems are *gray*.

In order to evaluate how a response selection system performs with strong distractors, Sato et al. (2020) propose a method to build test sets with *well-chosen false candidates*. The choice of these candidates consists on retrieving candidates related to the ground-truth response, based on the similarity between their content words, and, from these, remove utterances that are acceptable as a response through human evaluation. This is, to the best of our knowledge, the closest approach to ours. Our **tailored distractors** are inspired by the distractor selection process in multiple choice question systems, and correspond to Sato et al. (2020)'s well-chosen false candidates, but, while they only use them for testing, we use them to **train** our model. Furthermore, we select them by taking into account the **similarity** between the whole sentences, whereas they only take the content words into account.

3.5 Evaluation

To evaluate our different components, we start with a brief study of the guidelines for machine learning experiments (Section 3.5.1).

Common NLP tasks' performance is evaluated through metrics as accuracy, recall and precision. For instance, when doing response selection, there is labeled data, so if the system selects the gold reply, it is correct, if it selects another utterance, it is incorrect. Particularly, Zhou et al. (2018) use the evaluation metric *R(n)@k* for the task of response selection, which is the recall of true positive replies among the *k* selected.

As previously mentioned, DialoGP3T has two objectives: ranking and generation. Therefore, we study evaluation metrics for those two objectives, respectively, in Sections 3.5.2 and 3.5.3.

However, evaluating the usage of context cannot be done using such metrics. To do so, we study how to evaluate the usage of context in Section 3.5.4.

3.5.1 Guidelines for Machine Learning experiments

Alpaydin (2010) describe a set of guidelines to take into consideration when evaluating machine learning models, and how to plan and design experiments.

They state that the three basic principles of experimental design are:

- **Randomization** – the order by which the settings are ran must be random, so that the results are independent
- **Replication** – for the same configuration, the experiment must be ran a number of times (cross-validation). This allows to obtain the experimental error which leads to the conclusion of whether

the results have *statistical significance*.

- **Blocking** – remove variability from the results. For example, if we are comparing to machine learning algorithms, the same training and testing data must be used in both of them, so that the results are comparable.

The suggested guidelines for machine learning experiments concern:

1. Define the *aim of the study*, by clearly stating the objectives of the experiment.
2. Select a response variable, such as the error or loss.
3. Choose *factors* (different algorithms, different datasets) and *levels* (different parameters for algorithm).
4. Choose the *experimental design*, as the size of training and testing sets, how the different factors will be tested (one or multiple at a time).
5. *Perform the experiment*, making sure that it is reproducible, by, for example, stating the seed value.
6. Perform a *statistical analysis of the data* to discover if the results obtained are statistically significant.

One way to assess the significance of the results is through *hypothesis testing*. We may define an hypothesis that $\mathtt{X} > \mathtt{Y}$; if the results are consistent with that hypothesis, then we fail to reject it, otherwise, we reject it. The null hypothesis is the one we want to test ($\mathtt{X} > \mathtt{Y}$), against the alternative one ($\mathtt{X} \leq \mathtt{Y}$). To test the null hypothesis, a *ttest* can be performed, to find, from the means of two independent samples, $\mathtt{X}$ and $\mathtt{Y}$, if those samples are from populations with different mean values. A *level of confidence* α is chosen as a statistically significance threshold, usually 0.05, and the *ttest* returns a *p-value*:

- if $\mathtt{p\text{-}value} <= 0.05$: we fail to reject the null hypothesis, meaning that the difference between the two means is statistically significantly different from zero at level of significance α
- if $\mathtt{p\text{-}value} > 0.05$: we have evidence to reject the null hypothesis, meaning that the difference between the two means is not statistically significantly different from zero at level of significance α

We will use these principles to guide the evaluation of our results.

3.5.2 Retrieval evaluation

To evaluate how good a system is at ranking candidates, the metric used is *Hits@k*, k in [1,5,10], which represents the correct answer in the top k hits. When k = 1, the result is the accuracy.

3.5.3 Generation evaluation

To evaluate how good a system is at generating responses, the following metrics are used:

- **BLEU** (Bilingual Evaluation Understudy Score) (Papineni et al., 2002) – evaluates machine translation by measuring how many words overlap on a translation and a reference translation.
- **TER** (Translation Error Rate) Score (Snover et al., 2006) – evaluates machine translation by measuring how much a translator would have to edit a translation so that it would match a reference translation. As this score is an error, the lower it is, the better.
- **BertScore** (Zhang et al., 2019a) – evaluates text generation by measuring the similarity between a candidate and a reference sentence.

3.5.4 Context evaluation

An **ablation** is a term that originates from neuropsychology, where parts of animals brains where removed to study how it affected their behaviour. Applied to machine learning, it consists on removing a feature from a model, to observe the effect it has on its performance (Sheikholeslami, 2019).

Ablation studies are used in two works that have the goal of understanding how neural models use context (Khandelwal et al., 2018) (Sankar et al., 2019), by measuring changes in the model's performance in absence of contextual information. Khandelwal et al. (2018) use it to explain how models use long-range context (larger sequence of words considered, beyond sentence level), and focus on LSTM, due to its ability to model long-range dependencies, and to remember some aspects that are useful to model context, such as sentence length and word order. Sankar et al. (2019) use ablation to show that neural models do not make use of all the information available to it (including conversation history), and use two models: recurrent seq2seq and transformer-based seq2seq.

In order to understand their approach, Khandelwal et al. (2018) propose two important concepts:

- *Infinite context setting* – provide the model with all the tokens prior to the target word.
- *Effective context size* – number of tokens of context which need to be provided to the model in order to achieve similar loss to providing infinite context.

Some interesting answers are found to the question "How much context is used?", in particular, that LSTM language models have an *effective context size* of about 200 tokens, which means that considering more than 200 tokens has the same loss as the *infinite context setting*. Knowing the *effective context size*, they study the importance of contextual information, such as word order and word identity, looking at both nearby and faraway context. They found that:

- *Local word order*, which considers the average length of one sentence, only matters for the most recent 20 tokens. The range (20 tokens) was determined by changing the order of tokens and observing how it affected the loss.

- *Global word order*, that considers all sentences in the conversation history, only matters for the most recent 50 tokens. This means that introducing perturbations in the context more than 50 tokens before the current one has no effect on the performance. From this analysis, 50 tokens was set as the boundary between nearby and long-range context.

- *Content words* (nouns, verbs and adjectives) matter more than *function words* (determiners and prepositions) – when a perturbation function is applied to the context (dropping), losing content words has a higher loss than losing function words.

While Khandelwal et al. (2018) introduce perturbations in the context to take the aforementioned conclusions, in (Sankar et al., 2019) these are used to show that neural models do not make use of all the information available to it (including conversation history). Two models were used: recurrent seq2seq and transformer-based seq2seq. The central premise of Sankar et al. (2019) is that models use some of the information available to it if they are insensitive to perturbations that destroy them. So, to see if the models use the context of the conversation, they introduce the following perturbations:

- utterance-level
  - shuffle sequence of utterances in dialog history
  - reverse order of utterances in dialog history
- word-level (within every utterance)
  - shuffle words in utterance
  - reverse order of words in utterance
  - drop 30% of words
  - drop all nouns
  - drop all verbs

The following observations were made:

- Even under extreme perturbations, **the models show little change in perplexity**, which means that they are insensitive to the perturbations and, therefore, using the central premise of the paper, make minimal use of the dialog history.

- **Transformers are insensitive to word-reordering**, indicating that they could be learning bag of words like representations. This observation is particularly odd, since Transformers have a Positional Encoding mechanism to represent the order of words in a sentence.
- Transformers are **less sensitive to perturbations that mess with utterance structure** than recurrent models, which suggests that the latter are better to model conversational dynamics.

The major conclusion is that both recurrent and transformer-based seq2seq models are not significantly affected by drastic changes in the conversation history, which means that they do not really take it into account to generate the answers.

Considering the conclusions taken in the two aforementioned works (Khandelwal et al., 2018) (Sankar et al., 2019), current neural models are not using context as we would expect them to be. This is a motivation for our work, since we want to study if considering context when choosing an answer will improve the quality of the given answers.

4

Multi-Agent Platform

Contents

In this chapter, we focus on integrating two retrieval-based multi-agent dialogue systems: MULTI-SSS (Santos, 2019) and CHATTUGA (Fernandes, 2019). Both systems focus on the idea that *all agents can potentially answer all questions*. In Section 4.1, we describe how we joined each of their components, in Section 4.2 we describe the creation of new agents, and in Section 4.3 we describe the addition of these agents to our system.

4.1 Integrating two multi-agent dialogue systems

Although the systems have different strengths and functionalities, here we focus on what they have in common: a **Dialogue Manager** (from now on, DM), **Agents**, and **Decision making strategies**.

In both systems, when the user poses a query to the system, it is received by the *DM*. The differences between them, in what regards the interaction between the *DM*, the *agents* and the *decision making strategies*, are pointed in Table 4.1.

Step	MULTI-SSS	CHATTUGA
Interaction between *DM* and *Agents*	Upon receiving a User query, the DM sends it to the *Agent Manager*, who is responsible for initializing the *agents* once and, when it receives a query from the DM, it sends the query to all *agents* and receives their answers, returning the set of answers to the DM.	When the DM starts, it sets up all *agents*. Upon receiving a User query, it sends it to all *agents* and receives their answers.
Interaction between DM and *Decision Making Strategies*	The DM sends the *agents*' answers to the *Decision Maker*, who sends them to the only active *Decision Making Strategy* and receives its answer, which is returned to the DM, who returns that answer to the User.	The DM directly sends the *agents*' answers to all the *decision making strategies*, with each of them returning a single answer. The set of answers by strategy is returned to the DM, who chooses the best answer to return to the User based on its confidence on each strategy.

Table 4.1: Relevant differences between the systems

Integrating the two mentioned systems consisted on joining the DMs, the agents and the decision making strategies.

4.1.1 Joining the DMs

Combining the DMs consisted on understanding what they had in common and what were their differences regarding their interactions with the agents and the decision making strategies, so that we could create a new one, for our system, that encompasses their similarities and differences while maintaining the system's consistency.

Regarding the **interaction between the DM and the agents**, as seen in the previous section's figures and in Table 4.1, both DMs are responsible for sending the user query to the agents and receiving their answers. In MULTI-SSS, an *Agent Manager* serves as an interaction point between the DM and the agents, while in CHATTUGA their interaction is direct. In our system, we chose to follow MULTI-SSS and have an *Agent Manager* to handle the interactions between the DM and the agents.

Regarding the **interaction between the DM and the decision making strategies**, both DMs send the set of answers given by the agents to the decision making strategies. In MULTI-SSS, this is done using the *Decision Maker*, an interaction point between the DM and the decision making strategies, responsible for sending the agents' answers to the only active decision making strategy and returning its answer to the DM. So, in MULTI-SSS, the final answer is chosen by the decision making strategy. In CHATTUGA, the agents' answers are sent directly from the DM to each decision making strategy, where each returns an answer to the DM, who, based on the weights given to each strategy, chooses the final answer to return to the user. So, in CHATTUGA, the final answer is chosen by the DM. In our system, we chose to have an interaction point between the DM and the decision making strategies – the *Decision Manager* – as in MULTI-SSS, but, instead of one single strategy active at a time, we have a panoply of strategies with weights associated, as in CHATTUGA, with the DM being responsible for the choice of the final answer to return to the user.

An advantage of keeping the decision making strategies' weights, as done in CHATTUGA, is that, in future work, they can be used by our system to learn which strategy gives better answers, using, for example, MULTI-SSS's online learning, which so far is only used to evaluate agents.

Regarding other functionalities of the DMs, from MULTI-SSS we kept its five running modes, including its online learning module, although we only use the multi-agent mode, but we kept the other functionalities available for future use. From CHATTUGA's DM, we used the training of classifiers, which are used to classify the user's queries.

4.1.2 Joining the agents

Joining the agents consisted on analysing the agents from both systems and removing any redundancy.

As previously mentioned, MULTI-SSS's agents include similarity metrics, as Cosine, Jaccard and Levenshtein, and it also has an Edgar agent. CHATTUGA's agents are Cheat, Edgar, SSS, SSS-AMA

and Talkpedia.

Analysing those agents, we see that CHATTUGA has an agent SSS, that is a previous version of MULTI-SSS, thus it is redundant to have this agent. It also has an agent SSS-AMA, which is the same as SSS, but using a corpus from Balcão do Empreendedor instead of the subtitles corpus. This was removed at first, but later replaced, as we will see further.

Both systems have an Edgar agent. In our system, the decision on which one to choose was made based on their storage usage and simplicity, so we chose to keep MULTI-SSS's.

Summarizing, the removed agents were CHATTUGA's SSS, CHATTUGA's SSS-AMA and CHATTUGA's Edgar.

New agents were created and added, as we will see in Sections 4.2 and 4.3.

We also made it easier to activate and deactivate agents, by just changing the value associated to the agent in the system's configuration file, where 0 means that the agent is not active, and 1 means it is active.

4.1.3 Joining the decision making strategies

There were no conflicts or redundancies between both previous systems' decision making strategies, particularly since they have different heuristics: MULTI-SSS's strategies make their decisions only based on the set of answers given by the agents, while CHATTUGA also takes other things into account, such as the area of expertise of the agents, the query labels, output by its classifier, and the query itself.

4.1.4 Final architecture

The integrated multi-agent system also focuses on the idea that all agents can potentially answer all questions. Regarding the interaction between the DM and the agents, we chose to keep MULTI-SSS's component that manages the agents – the *Agent Manager*. We also chose to keep the component that manages the decision making strategies – the *Decision Manager* – which has a different name, that will be explained next. The main components of the integrated multi-agent system are the following, some of which were already described when the systems were introduced in Section 2.4:

- ***Dialogue Manager*** (*DM*) – interaction point between the user, the *Agent Manager* and the *Decision Manager*. Upon receiving a user query, it sends it to the *Agent Manager*, that returns a set of answers by *agent*. The *DM* forwards these answers to the *Decision Manager*, which returns a set of answers, one given by each *decision making strategy*. Based on the *DM*'s confidence on each *strategy*, it chooses one final answer that is returned to the User.
- ***Agent Manager*** – interaction point between the *DM* and the *agents*. On its first call, it is responsible for initializing the *agents*. Upon receiving a query from the *DM*, it retrieves a set of candidates

using a search engine. It forwards the query to all active *agents*, along with the candidates, for those who require it, and receives the responses given by each of them. The set of responses by *agent* is then returned to the *DM*.

- ***Agent*** – receives a query and, optionally, a set of candidates and returns one or more answers, based on its functionality.
- ***Decision Manager*** - interaction point between the *DM* and the *decision making strategies*. On its first call, it is responsible for initializing the *decision making strategies*. Upon receiving a set of answers from the *DM*, it forwards it to each *decision making strategy*, with each of them returning one answer. The set of answers by *strategy* is then returned to the *DM*. The name of this module was changed from Decision Maker to *Decision Manager* because it does not decide which answer to return to the user; this decision is made by the *DM*, as in CHATTUGA. Instead, it is responsible for managing the *decision making strategies*.
- ***Decision Making Strategy*** - receives a set of answers by *agent* and chooses one of them to return, based on its heuristic.

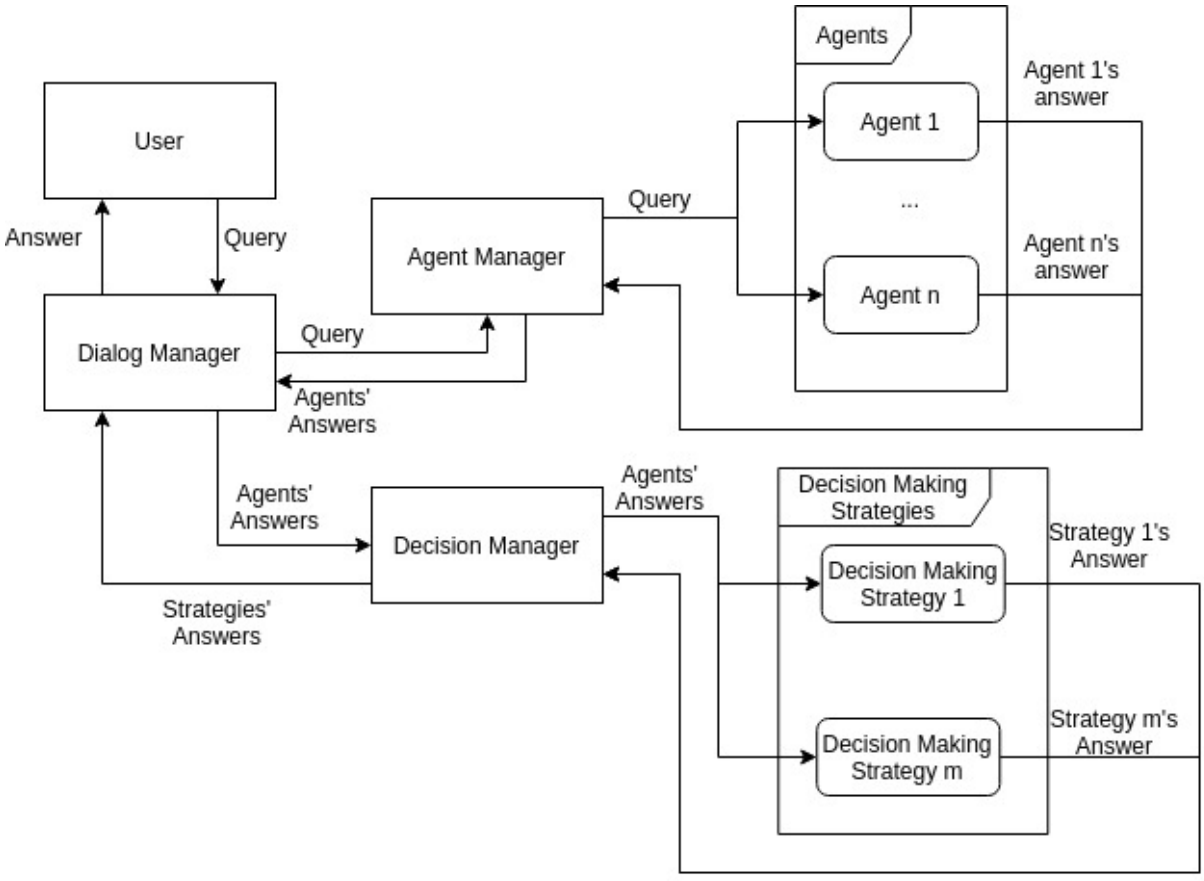

Figure 4.1: Integrated system's architecture

The architecture of the resulting integrated multi-agent system can be seen in Figure 4.1.

As a technical detail, in CHATTUGA all the decision making strategies were on the same file; in SSS, they were classes in different files, with similar code functions, but there was no inheritance. The same happens with the agents. To avoid code repetition and better modularize our code, we created two abstract classes: `Agent`, with the abstract method `requestAnswer(query)`, and `DecisionMethod`, with abstract method `getAnswer(answers)`. Each agent is now represented in a different class and inherits from `Agent`, overriding `requestAnswer`'s behaviour to match that agent's behaviour. The same applies to all decision making strategies, which are now in different classes which inherit from `DecisionMethod` and override `getAnswer` to select an answer based on that strategy's heuristic.

For a better understanding of our system, we present a running example:

1. The user poses the query *q* to the system: "Quantos anos tens?" ("How old are you?")
2. The DM classifies *q* with labels QUESTION, WH_QUESTION, PERSONAL, OTHER, NUM, COUNT and AGENT_LIFE, and forwards `q` to the *Agent Manager*.
3. The *Agent Manager* instantiates the active agents: Cosine, Edgar and Talkpedia, and forwards `q` to them.
4. The agents return their selected responses to the *Agent Manager*:
   - a_{Cosine}: "Ela é do ano do galo" ("She is from the year of the rooster.")
   - a_{Edgar}: "Tenho 65 anos." ("I'm 65 years old.")
   - $a_{Talkpedia}$: "Não sou um grande perito nesse tema, mas sobre isso sei que em agosto de 2019, os Wet Bed Gang actuaram no festival de música MEO Sudoeste." ("I am not a great expert on this subject, but I know that in August 2019, Wet Bed Gang performed at the MEO Sudoeste music festival.")
5. The *Agent Manager* sends the agents' answers to the DM.
6. The `DM` forwards the agents' answers, [a_{Cosine}, a_{Edgar}, $a_{Talkpedia}$], to the *Decision Manager*.
7. The *Decision Manager* sends the agents' answers to the active Decision Making Strategies (simple majority, with a weight of 0.4, and query agent, with a weight of 0.6).
8. Each of the decision making strategies selects one of the agents' answers and returns it to the *Decision Manager*:
   - query agent – a_{Edgar}, "Tenho 65 anos." ("I'm 65 years old.")
   - simple majority – a_{Cosine}, "Ela é do ano do galo" ("She is from the year of the rooster.")

9. The *Decision Manager* returns the strategies' answers to the DM.

10. Since the decision strategies returned different answers, and query agent has a higher weight, the DM returns Edgar's answer, a_{Edgar}, to the user: "Tenho 65 anos." ("I'm 65 years old.")

4.2 Creating new agents

After the work developed in the previous section, a first version of our system was ready to use. When a user introduces a query, it is classified by the classifiers developed on CHATTUGA, getting a set of labels. These labels are used, for example, in the *Query Agent* strategy, where, to select an answer, the labels of the query are compared to those of the agent who gave each answer. These labels can be, among others, `YN_QUESTION`, when it is a question that can be answered with "yes" or "no", `OR_QUESTION`, when it is a question where the answer is one thing or another, or `IMPERSONAL`, where the question is not personal. For the latter, we have a specialized agent – Talkpedia (Mota, 2015) – that answers impersonal questions using Wikipedia[1]. There were, however, no specialized agents for labels `YN_QUESTION` and `OR_QUESTION`, which are common types of question. To solve this, we created two new simple agents specialized in those types of question: *Yes No Agent* and *Or Agent*. Section 4.2.1 describes these agents.

In the previous section, it was mentioned that CHATTUGA's agent SSS-AMA was deleted, because it was an exact copy of the original SSS, but with a different corpus. A direct solution to solve this would be to create an AMA agent, that would take the AMA corpus and return an answer. However, this would imply that, every time we wanted to add a new agent with a different corpus to our system, we would have to manually create a new agent, whose behaviour would be identical to the other agents. To avoid this repetition, and to add other functionalities that the system also lacked, the **General Agent**, described in Section 4.2.2 was created.

Finally, since one of the objectives of this thesis is to develop an agent that takes context into account, Section 4.2.3 describes the creation of a contextual agent.

4.2.1 Simple agents

- **Yes No Agent** – given a query, randomly returns "yes" or "no".
- **Or Agent** – given a query, randomly returns the phrase that is before or after the "or" in the sentence, using syntactic parsing, as seen in the following example, where the answer is randomly chosen between "azeite" ("olive oil") and "vinagre" ("vinegar").:

[1] https://www.wikipedia.org/ (Last accessed on: 01/08/2020).

"Gostas mais de azeite ou vinagre?" ("Do you prefer olive oil or vinegar?")

4.2.2 General agent

As the name implies, the General Agent is a standard agent that can have multiple instances, where each instance has a different corpus. The agent's name and the path to its corpus are defined in the General Agent's configuration file, as shown in Listing 4.1, where there is one active agent and corresponding paths.

Listing 4.1: General Agent configuration file example

```
<?xml version="1.0" encoding="UTF-8"?>
<config>
    <mainClass>GeneralAgent</mainClass>
    <agentAmount>1</agentAmount>

    <excelPath name='ExampleAgent'>None</excelPath>
    <corpusPath name='ExampleAgent'>corpora/example.txt</corpusPath>
    <indexPath name='ExampleAgent'>resources/whooshIndexes/example</indexPath>
    <labelsPath name='ExampleAgent'>corpora/exampleLabels.txt</labelsPath>

    <threshold>0.35</threshold>
    <useWhoosh>true</useWhoosh>
    <stopwords>resources/stopwords/edgar_stop.txt</stopwords>
</config>
```

General Agent has the following new features:

- **dynamically adding new agents** - in order to create a new agent, the only requirement is a *csv* or *excel* file, with the following mandatory columns:
  - *topic* – topic of the question, needed for classification purposes.
  - *question*
  - *answer*

  Optionally, it can also have the following columns:
  - *source* – the source of the information, useful, for example, when using FAQs from different websites or public institutions.
  - *paraphrases* – list of paraphrases of the question.

As we wanted our system to, given a paraphrase of a query in the corpora, give the correct answer, an experiment was made to test the performance with paraphrases of the search engine used to retrieve candidates to feed the agents, Whoosh. We fed it with question-answer pairs and tested whether, retrieving only one result for a query which is a paraphrase of the question, the retrieved question was the correct one. The result was 64% accuracy, so we added each paraphrase with its corresponding question's answer to Whoosh. Although these are kept as different question-answer pairs on Whoosh, they are still signaled on our corpus as being paraphrases, having the same `DialogId`, as it may be handled in the future.

Thus, given a question, `Q`, an answer, `A`, and a list of paraphrases, for instance, [`P1`, `P2`], what enters our system is the following:

```
SubId - 001001
DialogId - 98
Diff - 0
I - Q
R - A
SubId - 001001
DialogId - 98
Diff - 0
I - P1
R - A
SubId - 001001
DialogId - 98
Diff - 0
I - P2
R - A
```

Notice how `DialogId` and `R` are the same, but the second and third `I` are paraphrases of the first, which is the one in the corpus.

From this file, all the needed files are automatically created, including a text file in the subtitle format that Whoosh will read, and those needed for question and answer classification. The new labels are also automatically extracted and added to the agents configuration file as the labels of expertise of that agent.

- **synonyms list** - a list of synonyms inside of the agent's domain can be provided as a text file, where each line has the following format:

  ```
  word,synonym1,...,synonymN
  ```

To improve efficiency, if a synonym is found in a user query, it is replaced by the main word before sending the query to Whoosh.

- **acronyms list** - a list of acronyms can also be provided in the same manner as the synonyms list described above, where each line has the following format:

  ```
  acronym,expansion
  ```

- **context as a list of labels** - a first implementation of context in our system consists on storing the topic, predicted by the classifier, of each user query, producing a history of topics. This is useful because, when the user poses a query that could match two different labelled questions in our corpus, we can choose the one whose label is the same as the user's previous query, as it is reasonable to assume that the topic is the same.

- **WordNet synonyms** - while comparing a user query to a list of candidate queries, General Agent calculates a set distance between the user query and the candidate query, using WordNet synonyms and stemming. If a synonym of a word in the user query is present on the candidate query, the word on the candidate query is replaced by the one in the user query.

4.2.3 Contextual agent

One of the objectives of this thesis is to extend the multi-agent system to use the context of the conversation. To accomplish that, we create an agent that uses the context of the conversation to select a response.

To calculate a matching between a context and a given response, we need to focus on their semantics. For this purpose, we represent them using word embeddings, which are representations of the meaning of words (Jurafsky and H. Martin, 2019). Recent matching models use attention to learn to which parts of each sentence they should pay more attention. Zhou et al. (2018) propose a deep attention matching network to match context and response that applies stacked attention layers over word embeddings to get multi-grained semantic representations of words. With this considered, our initial plan was to follow Zhou et al. (2018) and work with word embeddings and multiple attention layers, and eventually get to a number that would express the level of matching between a context and a given response.

However, the idea of joining word embeddings and attention layers was recently simplified through the use of BERT (Devlin et al., 2019), which was introduced in Section 2.3. It is a pre-trained language model that joins word embeddings and attention by stacking Transformer encoder layers. BERT is an example of transfer learning, where models are pre-trained with very large amounts of data and can be fine-tuned, using a small amount of data, to specific tasks. Particularly, it can be fine-tuned to numerous specific

NLP tasks. Hugging Face[2] provides a set of classes for fine-tuning `BertModel`, the class that represents BERT, for NLP tasks, such as `BertForSequenceClassification`, `BertForNextSentencePrediction`, and `BertForQuestionAnswering`.

`BertForNextSentencePrediction` consists on a `NextSentencePrediction` layer on top of `BertModel`. `BertModel`'s input can be composed of one or two sentences; in the former, the output is a sentence embedding, while in the latter, it is a single embedding representing both sentences. `BertForNextSentencePrediction` always takes two sentences as input, `s1` and `s2`, and returns a vector [`p1`, `p2`], where `p1` is the probability of `s2` following `s1`, and `p2` is the probability of `s2` being a random sentence.

Since our goal is to select the answer that best matches the context of the conversation, that is, the sentence that, given the context of the conversation, is more likely to be the next utterance returned by our system, we decided to use `BertForNextSentencePrediction` to solve our problem, using as input the context of the conversation (`s1`) and a candidate response (`s2`). Thus, the best response will be the one with highest `p1`.

Since our system is for the Portuguese language, the proper pre-trained model to use is *Bert Base Multilingual cased*[3], which is pre-trained for 104 languages. This model is supposed to support sentences with up to 512 tokens. Although the context of a conversation can be longer than 512 tokens, we chose to only consider the last 512 tokens of the context. However, we found that there were indexing problems with sentences with more than 200 tokens. Thus, we used a model fine-tuned from the original *Bert Base Multilingual cased* that accepts long sentences – *bert-base-multilingual-cased-sentence*[4].

Considering that we have a conversational system, we want our model to be able to predict if a sentence is the best next utterance in a *dialogue*. However, the available models were not pre-trained on a dialog dataset. Thus, we use transfer learning, by instantiating the Hugging Face model `BertForNextSentencePrediction` with the pre-trained model configuration *bert-base-multilingual-cased-sentence*. Then, we fine-tune this model to our specific task, which is to predict if a sentence is the best next utterance in a dialogue. For such task, we need a dialogue dataset.

Our system uses subtitle data (Subtle corpus) to retrieve responses using a search engine, which are sent to each agent, although, depending on their configuration, they can use another corpus. In the Subtle corpus, each entry has a `SubId` (subtitle id), `DialogId`, `Diff`, `I` (interaction) and `R` (response). Although the field `DialogId` exists, it does not identify a full dialogue, but a single interaction between two communicators, and is incremented upon each new utterance.

However, to train our model, we need to be able to properly identify the context of a conversation, and, therefore, need a dataset, in Portuguese, with full dialogues identified.

Existing datasets with the structure we need are the Ubuntu Dialogue Corpus (Lowe et al., 2015) (in

[2]https://huggingface.co/ (Last accessed on: 01/08/2020).
[3]https://huggingface.co/bert-base-multilingual-cased (Last accessed on: 01/08/2020).
[4]https://huggingface.co/DeepPavlov/bert-base-multilingual-cased-sentence (Last accessed on: 01/08/2020).

Measure	Number of utterances	Number of tokens	Number of tokens per utterance
Minimum	1	2	1
Maximum	28	584	217
Mean	3.6136	37.5411	10.3889
Median	3	26	7
Mode	2	12	4
Standard Deviation	2.5213	38.2454	11.3855
Variance	6.3570	1462.7072	129.6301

Table 4.2: Dataset dialog's statistical measures

English), the Douban Conversation Corpus (in Chinese) (Wu et al., 2017) and the Cornell Movie-Dialogs Corpus (Danescu-Niculescu-Mizil and Lee, 2011) (in English). The last is made of movie subtitles, like Subtle is, but with a file that contains the structure of its 83096 conversations, `movie_conversations.txt`, from which we can obtain the utterances that belong to each conversation, allowing us to, at each point of a conversation, know its context. However, this corpus contains data for the English language, and no similar corpus was found for the Portuguese language. Because we considered that having this data in Portuguese would be crucial for our work, we decided to manually translate 5000 dialogues to Portuguese.

Although 5000 out of 83096 dialogues is a small sample, one of the goals of transfer learning is to use a small amount of data to fine-tune the pre-trained model, thus we thought that we could get good results with that number of dialogues.

The translated dataset contains 18039 utterances across 5000 dialogues. It contains four columns:

- *line_id* – uniquely identifies each utterance.
- *dialog_id* – uniquely identifies each dialog, where each dialog has one or more utterances.
- *utterance* – text of the utterance, in English.
- *utterance_pt* – text of the utterance, in Portuguese.

In Table 4.2, the intra-dialogue statistics `number of utterances` and `number of tokens` are shown. It is also shown the statistics of the number of tokens per utterance, in the whole dataset.

The intuitive way to use the dataset to train our model would be, for each dialogue, and for each utterance in that dialogue, consider the context to be all the previous utterances before the current one. We would use the model `BertForNextSentencePrediction`, fine-tuning it by stating that the current utterance is the continuation of the context, using the label 0 (`isNext`). However, we would only be training our model with one label, not giving it any information regarding the label 1 (`notNext`), so it would always predict the label 0.

To solve the aforementioned problem, for the same context, we give our model two possible next sentences: the utterance that follows it – *gold reply* – and a **distractor**, with labels 0 and 1, respectively.

The criteria used for selecting a distractor for each gold reply is described in Chapter 5.

Having a distractor selected, we encode two sentence-pairs: the context with the gold reply and the context with the distractor, and add the labels 0 and 1 to a list of labels. From these encodings and list of labels, we can create a dataset structure from where batches will be extracted to train our model. The process of creating the dataset structure from the dialogues is summarized in Algorithm 4.1.

Algorithm 4.1: Dataset creation from dialogues

```
inputs = [];
labels = [];
for dialog in dialogs do
    for i in range(len(dialog)) do
        gold_reply = dialog[i];
        context = dialog[ : i];
        distractor = selectDistractor(gold_reply);
        sim = spacy.similarity(gold_reply, distractor);
        encoded_gold = tokenizer.encode_plus(context, text_pair=gold_reply);
        encoded_distractor = tokenizer.encode_plus(context, text_pair=distractor);
        inputs.append(encoded_gold);
        labels.append(0) ;                          // 0 - gold_reply follows the context
        inputs.append(encoded_distractor);
        labels.append(1) ;                  // 1 - distractor does not follow the context
dataset = createDataset(inputs, labels);
```

To train our model, instead of retraining the entire `BertForNextSentencePrediction` model, which consists on the `BertModel` with a `NextSentencePrediction` layer on top, we froze `BertModel`'s layers and only trained the last layer, which speeds up the training. To adjust the model's weights and the learning rate, we used the Adam optimizer and a scheduler. After each epoch of training, the performance of the model is tested on the validation set. The training and validation loops are shown, respectively, in

Algorithms 4.2 and 4.3.

Algorithm 4.2: Training loop

```
model = BertForNextSentencePrediction.from_pretrained("DeepPavlov/bert-base-multilingual-
  cased-sentence");

model.train();
nr_epochs = 4;
learning_rate = 2e-5;
for i in range(nr_epochs) do
    total_train_loss = 0;
    for batch in train_dataloader do
        inputs = batch[0];
        labels = batch[1];
        loss, logits = model(inputs, next_sentence_label=labels);
        total_train_loss += loss.item();
        loss.backward();
        optimizer.step();
        scheduler.step();
    end
    avg_epoch_train_loss = total_train_loss / len(train_dataloader);
end
```

Algorithm 4.3: Validation loop

```
model.eval();
total_accuracy = 0;
for batch in validation_dataloader do
    inputs = batch[0];
    labels = batch[1];
    logits = model(inputs, next_sentence_label=labels);
    total_accuracy += getAccuracy(labels,logits);
end
avg_val_accuracy = total_accuracy / len(validation_dataloader)
```

Regarding the hyperparameters used to train the model, according to Devlin et al. (2019)'s fine-tuning procedure, the batch size must be 16 or 32; the learning rate must be 2e-5, 3e-5 or 5e-5; and the number of epochs must be 2, 3 or 4.

For our model, we trained it using one GPU with 11178MiB of memory. We experimented both batch sizes of 16 and 32, and found that the available memory was not enough to train with the latter. Thus, we use a batch size of 16 to train our model. Since a scheduler is used to adjust the learning rate at each iteration of the training, we assumed that its initial value would not significantly impact the results and thus selected a learning rate of 2e-5.

To select the number of epochs necessary to train our model, in Section 6.2.1 we compare a baseline approach, with random distractors, to one with tailored distractors, and through the metric accuracy choose the best distractor selection approach and number of epochs.

4.3 Adding new agents

After creating the agents mentioned in the previous Section, in this Section we show how to add them to our system.

The procedure to add a new agent is:

1. Create a new folder inside folder ExternalAgents with the name of the new agent, with the class that contains the implementation of the agent and its configuration file:

   - `agents/ExternalAgents/NewAgent/config.xml`
   - `agents/ExternalAgents/NewAgent/NewAgent.py`

2. The class that implements the agent must be a subclass of `Agent` and, thus, implement the `requestAnswer` method, that receives a request and returns a response.

3. The agent's name must be added to the system's configuration file, with a new line under tag `agents`:

```
<agent name="NewAgent" active="0"/>
```

4. The agent's name and areas of expertise must be added to the agents' configuration file:

```
<externalAgent name="NewAgent">
    <labels>
      <label score="1.0">QUESTION</label>
      <label score="1.0">NON_QUESTION</label>
    </labels>
</externalAgent>
```

   If the agent does not have an area of expertise, the labels `QUESTION` and `NON_QUESTION` are added, so that all type of user input is sent to that agent, since every input is either a question or a non question.

4.3.1 Simple Agents

For the simple agents, Yes No Agent and Or Agent, no additional steps are required.

4.3.2 General Agent

After adding the General Agent to the system, using the steps described above, instances of it are created. To do so, the following steps are necessary:

- If the instance agent requires extra synonyms and/or acronyms, it is necessary to create a folder, inside the General Agent folder, with the instance agent's name, and the files `synonyms.txt` and/or `acronyms.txt`.
- Add the instance agent's information to the General Agent's configuration file, as seen in Section 4.2.

4.3.3 Contextual Agent

Besides the steps described above, on the Contextual Agent's initialization, the tokenizer and the fine-tuned model were loaded. Then, upon receiving a user query, the DM forwards it to the AgentHandler, along with the context, which is represented as a list with each query and system response. The AgentHandler sends the context to every agent, although only the ContextAgent uses it.

When the AgentHandler sends an user query to the ContextAgent, for each candidate retrieved by Whoosh, it passes the context concatenated with the user, `c`, input and the candidate's response, `r`, to the fine-tuned model. The model returns a list with two elements, where the first is the probability of `r` following `c`, and the second is the probability of `r` being a random sentence. The *n* response with higher first probability are returned by the agent, being *n* the number of responses given by each agent.

5

Distractor Selection

Contents

In the Next Sentence Prediction task of BERT pre-training, given two sentences `A` and `B`, 50% of the time `B` is the actual sentence that follows `A`, and is labeled as so (`IsNext`), and 50% of the time it is a **random** sentence from the corpus (`NotNext`).

However, in retrieval-based systems, a search engine is used to retrieve a number of candidates, from which the model selects a response. Thus, the candidates already have some degree of similarity between them, as proven by an experiment where we used a corpus of 360 chitchat questions for the Portuguese language, and a corpus with movie subtitles – subTle (Ameixa et al., 2013) – to retrieve the candidate responses. First, we computed the similarity, using spaCy[1], between candidate responses retrieved by Whoosh; then, we did the same but with responses randomly chosen from that corpus. In both settings, for each question in the chitchat corpus, *n* candidates were retrieved, where *n* is one of 2, 5, 10 and 20. The results are shown in Table 5.1, where each value is averaged over five runs of that setting.

# Retrieved candidates	Whoosh	Random
2	**0.3499** $\pm$ 0.2445	0.1703 $\pm$ 0.20736
5	**0.3427** $\pm$ 0.2410	0.1693 $\pm$ 0.21034
10	**0.3295** $\pm$ 0.2250	0.16676 $\pm$ 0.20818
20	**0.3252** $\pm$ 0.2196	0.16936 $\pm$ 0.20844

Table 5.1: Spacy similarity of responses retrieved by Whoosh and random responses

We conclude that, on average, when using Whoosh, the similarity amongst candidates decreases as the number of candidates increases; no correlation is found when using random candidates. Furthermore, we see that candidates retrieved by the search engine are, on average, **two times more similar** than the ones randomly retrieved, which motivates the introduction of other ways of selecting distractors, other than randomly.

To the best of our knowledge, the closest approach to ours of distractor selection is done in Sato et al. (2020). Assuming that systems that generate appropriate responses can also select them, they use response selection to evaluate response generation systems. This tackles the problem of having many appropriate responses for one input context (one-to-many problem), making it difficult to evaluate if a generated response is appropriate. To infer if a system is good at selecting responses, they feed it with the ground-truth response and false candidates. Instead of randomly choosing the false candidates, they select *well-chosen* false candidates, to test whether the system can select the correct response amongst similar responses. Their selection of false candidates is done by comparing the content words of each candidate with those of the ground-truth response.

Our distractors correspond to their well-chosen false candidates, but we use them when training our model, while they only use them for testing. Furthermore, we select distractors by taking into account the similarity between the whole sentences, whereas they only take the content words into account.

[1] https://spacy.io/ (Last accessed on: 06/12/2020)

Listing 5.1: Noisy distractors example

```
'history'={"hello what are doing today ?"}
'candidates'={"NGDPRUJUPFKHRYKBYMCS",
              "DRDSAKCLUQSJRXGNKVYT",
              "RIXAJZFIGELJFINUBEVE",
              "DTQFLDQQRBFICKTLNLOO",
              "i am good , i just got off work and tired , i have two jobs ."}
```

Next we present four ways of selecting distractors. First, we propose the creation of noisy distractors to observe how they affect the performance metrics (Section 5.1). Then, we propose an approach that uses a search engine to retrieve candidates close to the correct response (Section 5.2). Then, we introduce a technique that computes a semantic similarity between the gold reply and a set of random utterances, in order to select a set of distractors (Section 5.3). Finally, we use a ranking system to assign probabilities to each candidate in the training corpus, and then retrain the system using as distractors the candidates, different from the gold reply, that were given higher probabilities of being the correct answer (Section 5.4).

5.1 Noisy

The first approach consists on creating noisy distractors. This approach's goal is not to improve the performance metrics, but to study if retrieval models effectively use the distractors, or if their performance does not change when these are replaced with noisy data.

The noisy distractors are generated as random strings, so that they do not make sense syntactically nor semantically.

Given a corpus, the approach consists on replacing all the distractors by noisy distractors, while keeping the gold reply intact. For example, using 4 distractors with the PersonaChat corpus, the result of this approach is seen in Listing 5.1.

5.2 Selecting distractors using a search engine

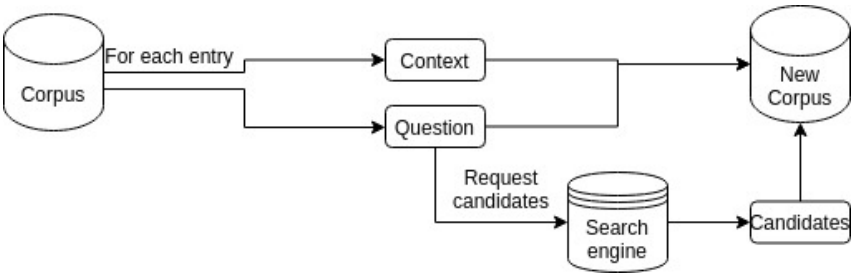

Figure 5.1: Search engine approach approach

When there is a large number of candidates, response selection systems use search engines to retrieve candidates, who are ranked according to that system's heuristic. To retrieve candidates from a search engine, it is necessary to have question-answer pairs, which are used to create indexes. Then, given a query and a number of hits, *n*, the search engine finds the *n* candidates that better matches that query, and returns their answers, ordered by their level of matching.

Given a corpus, *c*, the first step is to create indexes: first, preprocess *c* to only consider question-answer pairs, namely, for each entry, the last utterance in the history and the gold response. Having created the indexes, a new version of *c* is created: for each history, we give Whoosh the last utterance and request *k + 1* candidates, depending on the number of distractors, *k*, wanted. From those, the first retrieved candidate will be the gold reply and the remaining *k* candidates are shuffled and used as distractors. The resulting dataset requires an additional processing step, which is to delete entries with less than *k + 1* candidates, which can happen because, occasionally, the search engine does not match the question sent with the requested number of hits. This process is shown in Algorithm 5.1.

Algorithm 5.1: Search engine

```
questions, answers = createQAs(c);
engine.createIndexes(questions, answers);
new_corpus = [];
for question in c do
    candidates = engine.requestCandidates(question, n_hits = k+1);
    gold_reply = candidates[0];
    distractors = candidates[1:k+1];
    if len(distractors) < k then
        continue;
    new_corpus.append(gold_reply, distractors);
```

The process of creating a dataset with tailored distractors retrieved from a search engine is illustrated in Figure 5.1. An example of an entry of a corpus created using this technique and the PersonaChat dataset is shown in Listing 5.2.

Listing 5.2: Search engine distractors example

```
'history'={"hello what are doing today ?"}
'candidates'={"hello , i am looking to see which soup kitchen needs volunteers this week .",
              "i am thinking about my upcoming retirement . how about you ?",
              "packing for school this morning and then heading to work later . you ?",
              "hi ! i need some advice .",
              "i am good , i just got off work and tired , i have two jobs ."}
```

5.3 Semantic similarity

Another approach to select tailored distractors is based on its semantic similarity with the gold reply. As previously seen, in the multiple choice question generation task, the selected distractors have a high degree of similarity with the gold reply, enough to make it difficult for students to select the correct answer without minimal domain knowledge, but are not paraphrases of the gold reply.

Our approach consists on selecting distractors that are semantically similar to the gold reply, without being paraphrases. To do this, using a natural language inference corpus, the average semantic similarity for the `paraphrase` relation is computed. Then, a set of random utterances is sampled from the corpus, and the similarity between each of them and the gold reply is calculated. The ones selected as distractors are those with higher similarity below the paraphrase threshold.

Thus, to build a corpus with this method, it is necessary to have a **corpus**, *c*, with dialogues and gold replies, a **Natural Language Inference corpus**, *c_nli*, with annotated paraphrases, in the same language as *c*, and a **method to compute semantic similarity**, *m*.

This method is described in Algorithm 5.2.

Algorithm 5.2: Semantic similarity

```
sim_by_label = {};
new_corpus = [];
for sentence_1, sentence_2, label in c_nli do
    sim_by_label[label] += m(sentence_1, sentence_2);
sim_paraphrase = avg(sim_by_label[paraphrase]);
threshold = select_below(sim_paraphrase);
for gold_reply in c do
    utterances = random.sample(c, n) ;          // select n random utterances from c
    for u in utterances do
        u.sim = m(u, gold_reply);
    utterances.sort(sim, descending);
    distractors = [];
    for u in utterances do
        if u.sim < threshold then
            distractors.append(u);
        if len(distractors) == k then
            break;
new_corpus.append(gold_reply, distractors);
```

5.4 Top-ranking

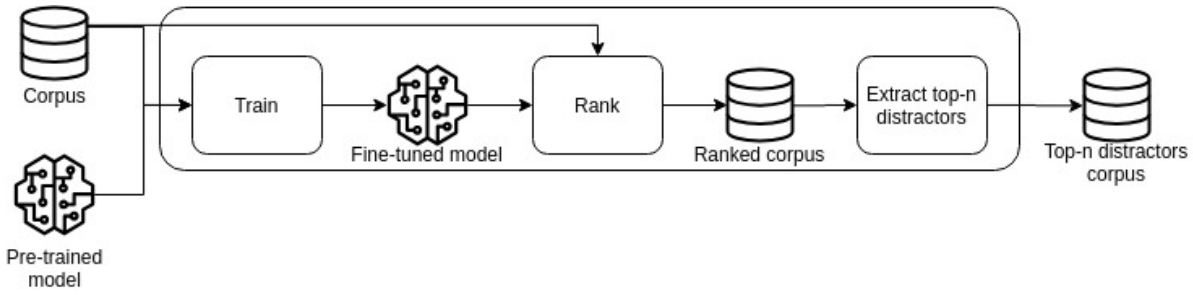

Figure 5.2: Top-ranking approach

This approach requires a ranking pre-trained model, *m*, and a corpus, *c*, with a set of entries consisting on a conversation history, a gold reply, and a set of distractors. We assume that *c* contains *n* distractors by entry, but, due to memory limitations, only *d*, randomly chosen, are used during training, where $d <= c$.

The approach consists on using a ranking model to rank candidates according to their probability of being the gold reply. Then, the *d* candidates with a higher probability, excluding the one that is the gold reply, are used as distractors in a new corpus, where each entry has the same conversation history and gold reply as the original corpus, the distractors being the only difference. Thus, the distractors in the

Listing 5.3: Top ranking distractors example

```
'history'={"hello what are doing today ?"}
'candidates'={"hey there , are you a mother ?",
              "yeah , well what about you ?",
              "i just got a pet fish for my 18th birthday yesterday from my parents .",
              "i am a recovering heavy drinker . full time . how about you ?",,
              "i am good , i just got off work and tired , i have two jobs ."}
```

tailored corpus are the ones that the model had higher difficulties to tell apart from the gold reply, and can then be used to train the model from scratch. This is more similar to a real world setting where the model will have to select a gold reply from a set of strong distractors.

This method is described in Algorithm 5.3 and shown in Figure 5.2. An example of an entry of a corpus created using this technique and the PersonaChat dataset is shown in Listing 5.3.

Algorithm 5.3: Top ranking

```
m.train(c);
rankings = m.rank(c);
new_corpus = [];
i = 0;
for history, gold_reply in c do
    top_k_ranked = rankings[i][:k+1];
    if gold_reply in top_k_ranked then
        top_k_ranked.remove(gold_reply)
    top_k_ranked = top_k_ranked[:-1]  distractors = top_k_ranked;
    new_corpus.append(history, gold_reply, distractors);
```

6

Evaluation

Contents

Listing 6.1: AMA example

```
1  [Marca_na_hora_e_marca_na_hora_online] 'Qual o custo de adquirir uma Marca na Hora e
       quais as modalidades de pagamento?'
2  ([Marca_na_hora_e_marca_na_hora_online] 'What is the cost of purchasing a Marca na Hora
       and what are payment methods?')
3
4  [Inscrições_online] 'Quais as modalidades de pagamento disponíveis?'
5  ([Inscrições_online] 'What are the available payment methods?')
```

To evaluate our work, we evaluate each of the developed components: how easy it is to add a new agent to our multi-agent system (Section 6.1), the impact of selecting tailored distractors (Section 6.2), a case study regarding customer support and tailored distractors (Section 6.3), and how a current model uses context (Section 6.4)

6.1 Creating and integrating agents into the multi-agent system: case studies

In this Section we show how to create and add new agents to our system. Since the creation and addition of other agents (not instances of General Agent) to our system were already tested with success, in collaboration with Universidade de Coimbra, here we only show how to create agents that are instances of the General Agent and how to add them to our system.

6.1.1 AMA

- **Creating the agent** As previously said, this thesis was developed as part of the project AIA – *Agente Inteligente para o Atendimento no Balcão do Empreendedor*, in collaboration with AMA – *Agência para a Modernização da Administração* (Administration Modernization Agency). A dataset containing FAQs about the services available by this agency was already on CHATTUGA, serving as SSS-AMA's corpus. This consists on a dataset with a label associated to each question-answer pair, where the label is the AMA service regarded in the question or answer.

  This new agent is useful to understand the impact of considering context as a list of labels, as seen in the previous section. In Listing 6.1, where the label is before the question, are two questions that belong to our corpus.

  If the user asks "Quais as modalidades de pagamento?" ("What are the payment methods?"), both the above questions could be a match, but they have different answers. If the previous interaction between the user and the system was predicted as being about `[Inscrições_online]`, it is intuitive

that the best match in this case is the one with this label. Since General Agent keeps the context as a list of labels, it is possible to access the previous label and give higher weight to answers with the same label, thus solving this dilemma.

- **Adding the agent to the system** For this agent, there is no list of synonyms nor acronyms, so we only need to add its information to General Agent's configuration file, as seen in Listing 6.2.

Listing 6.2: General Agent configuration file with AMA agent example

```
<?xml version="1.0" encoding="UTF-8"?>
<config>
    <mainClass>GeneralAgent</mainClass>
    <agentAmount>1</agentAmount>

    <excelPath name='AMAAgent'>None</excelPath>
    <corpusPath name='AMAAgent'>corpora/ama.txt</corpusPath>
    <indexPath name='AMAAgent'>resources/whooshIndexes/ama</indexPath>
    <labelsPath name='AMAAgent'>corpora/AMAlabels.txt</labelsPath>

    <threshold>0.35</threshold>
    <useWhoosh>true</useWhoosh>
    <stopwords>resources/stopwords/edgar_stop.txt</stopwords>
</config>
```

6.1.2 COVID

- **Creating the agent**

  Given the pandemic happening in the world when this thesis was being developed, an agent to answer questions about COVID-19 was created. The corpus was manually created by collecting 75 question-answer pairs from FAQs from Portuguese official health sources[1]. To create the corpus, a taxonomy was created to manually classify the *topic* of each question-answer pair, and a list of paraphrases for each question were manually filled.

  The following are some of the columns of the corpus, along with the corresponding values of an instance of the dataset:

[1] https://covid19.min-saude.pt/perguntas-frequentes/ (Last accessed on: 05/12/2020)

- *question* (example: “O vírus transmite-se pelos animais?” (“Is the virus transmitted by animals?”))
- *answer* (example: “Não havendo uma certeza absoluta, há indícios de que os cães e os gatos poderão ser fontes de contaminação, por exemplo através do pelo. Mas ainda não sabemos se o são a um nível que justifique preocupações sérias para os seres humanos.” (“With no absolute certainty, there are indications that dogs and cats may be sources of contamination, for example through hair. But we still don’t know if they are at a level that justifies serious concerns for human beings.”))
- *topic* – the topic of the question. For example, if the topic is PROTECÇÃO (PROTECTION), it means that the question is about how to protect from the virus. (example: CONTÁGIO (CONTAGION))
- *subtopic* (optional) – one or more subtopics of the question, related to the topic. For example, if the topic is PROTECÇÃO (PROTECTION), a subtopic can be MÁSCARAS (MASKS) – the question concerns protection using face masks, or LUVAS (GLOVES) – the question concerns protection using gloves. (example: ANIMAIS (ANIMALS))
- *source* – public institution from which the information was taken (example: JORNAL EXPRESSO)
- *paraphrase id* – this column is filled with sequential numbers, unless it is a paraphrase to an existing entry, which is possible since there are data from different sources.
- *paraphrases* (optional) – list of paraphrases of the question. (example: “Os animais transmitem o coronavírus aos humanos?” (“Do animals transmit the coronavirus to humans?”))

In this case study, it is clear why it is important to use a custom list of synonyms. When a user talks about COVID, he may also write it as “COVID-19” or “COVID 19”, and when talking about “coronavirus”, he may write it as “corona” or just “virus”. In the COVID domain, those examples are synonyms, but not in the WordNet, which is why it is crucial to have them as a list of synonyms to let our system treat them as such.

- **Adding the agent to the system**

Listing 6.3: General Agent configuration file with Covid agent example

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <config>
3     <mainClass>GeneralAgent</mainClass>
```

```
    <agentAmount>1</agentAmount>

    <excelPath name='AntiCovidAgent'>corpora/bot-FAQ.xlsx</excelPath>
    <corpusPath name='AntiCovidAgent'>corpora/covid.txt</corpusPath>
    <indexPath name='AntiCovidAgent'>resources/whooshIndexes/covid</indexPath>
    <labelsPath name='AntiCovidAgent'>corpora/covidLabels.txt</labelsPath>

    <threshold>0.35</threshold>
    <useWhoosh>true</useWhoosh>
    <stopwords>resources/stopwords/edgar_stop.txt</stopwords>
</config>
```

To add this agent to our system, we just need one file: the excel or csv with the columns mentioned above. Then, we add the agent's files to General Agent's configurtation file; particularly, the path to the excel file (`excelPath`), and the desired paths for the files that will be created: the corpus (`corpusPath`), the indexes (`indexPath`) and the labels (`labelsPath`), as seen in Listing 6.3. Also, to let the system know the custom synonyms and acronyms, the folder AntiCovidAgent containing an acronyms file – `acronyms.txt` – and a synonyms file – `synonyms.txt` – must be created inside the General Agent folder. An example for each of these files can be seen in Figures 6.1 and 6.2, respectively.

```
DGS,Direção Geral de Saúde
OMS,Organização Mundial de Saúde
SNS,Serviço Nacional de Saúde
```

Figure 6.1: generalAgent/AntiCovidAgent/acronyms.txt

```
covid 19,covid,covid19,covid-19
coronavirus,coronavírus,vírus,virus,sars-cov-2
```

Figure 6.2: generalAgent/AntiCovidAgent/synonyms.txt

6.2 Results concerning tailored distractors

In this Section, we present the results of our different methods of selecting distractors. We evaluate them for the Portuguese (Section 6.2.1) and English (Section 6.2.2) languages.

6.2.1 Portuguese

For the Portuguese language, our goal is to find the setting that shows better results for fine-tuning the `BertForNextSentencePrediction` model, to be used in our system's contextual agent. Here, we describe the results with the Semantic Similarity and Whoosh approaches. Finally, we compare them and choose the one to use in our agent.

6.2.1.A Noisy

As we will see further, in Section 6.2.2, using noisy distractors significantly decreased the retrieval metrics, therefore we decided not to test them for the Portuguese language.

6.2.1.B Search engine

To select distractors using a search engine, we use Whoosh and the translated mini Cornell corpus. Since the model used for the Portuguese language is `BertForNextSentencePrediction`, to fine-tune it we need, for each utterance, one positive example and one negative. Thus, in this setting, only one distractor is retrieved from Whoosh. For this experiment, we used the Whoosh indexes that were already created for the Subtle corpus. For each utterance of our corpus, 3 candidates were retrieved by Whoosh, and the 3rd one was used as a distractor.

To select whether we will fine-tune our `BertForNextSentencePrediction` model with 2, 3 or 4 epochs, we make experiments with 4 epochs of train and then select the one with higher average accuracy. Since the candidates retrieved from Whoosh are deterministically chosen, instead of creating five different datasets, as in the previous experiment, we create one dataset and randomly split it five times into training and testing set, and fine-tune the model with the training one, computing the accuracy at the end of each epoch. We also repeat this four times for each, in order to obtain enough results to measure if they have statistical significance.

We repeat the aforementioned process for datasets created with *random* distractors, and testing the models with a Whoosh validation set, to see if selecting Whoosh distractors in training improves the results in testing, compared to selecting them randomly.

Table 6.1 shows the results, averaged over all the results obtained as described above, and their statistical significance. We observe that training the model with the Whoosh distractors improves, on average, 5% compared to training with the random ones. From Figure 6.3, built with the data from Table 6.1, we observe that the results are, on average, better in the 4th epoch, when trained with Whoosh distractors, and better in the 3rd epoch, when trained with random ones. The original results can be consulted in A.2.

Epoch	Random	Whoosh	*p-value*	Significant
1	0.49454 ± 0.00953	**0.55024** ± 0.00262	9.68417e-5	Yes
2	0.49176 ± 0.00356	**0.5541** ± 0.00470	2.77729e-8	Yes
3	0.49468 ± 0.00696	**0.55556** ± 0.00322	3.62604e-6	Yes
4	0.49452 ± 0.00400	**0.55678** ± 0.00402	8.10505e-9	Yes

Table 6.1: Context agent - train with corpora built with Random and Whoosh distractors, test with a corpus built with Whoosh distractors

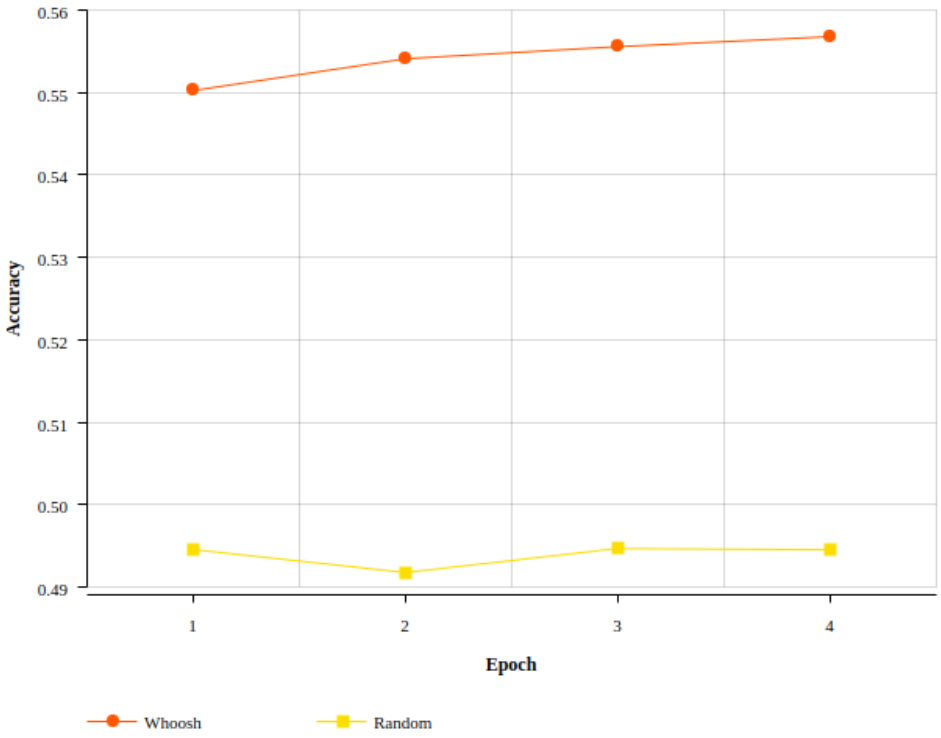

Figure 6.3: Accuracy by epoch, random vs Whoosh

6.2.1.C Semantic Similarity

As stated in Section 5.3, to use this method to select distractors, we need a **corpus**, a **NLI corpus** and a **method to compute semantic similarity**. We use our translated mini Cornell corpus, used to train the model for our contextual agent. The selection of threshold and similarity method are described next, as well as the results of using this distractor selection approach. 4.2.3.

- **Selecting a threshold and similarity method**

  Two natural language inference corpus for the Portuguese language were used: SICK_BR (Real et al., 2018) and ASSIN-1[2].

[2]http://propor2016.di.fc.ul.pt/?page$_id = 381(Lastaccessedon : 12/12/2020)$

– The SICK_BR corpus has 10000 pairs of sentences, A and B, annotated with the relation between them: `ENTAILMENT`, `NEUTRAL` or `CONTRADICTION`. It also annotates the relation from an unilateral point of view, with the features `A_entails_B`, `B_entails_A`, `A_neutral_B` and `B_neutral_A`. For example, given two sentences A and B, if `A_entails_B` and `B_neutral_A`, the relation is `ENTAILMENT`. If `A_entails_B` and `B_entails_A`, the annotated relation is also `ENTAILMENT`, but they are paraphrases. Since we want to compute the semantic similarity, we are interested in isolating paraphrases from entailments. Thus, we preprocessed the corpus so that these cases are labelled as `PARAPHRASE`. After having these labelled, we dropped the columns `A_entails_B`, `B_entails_A`, `A_neutral_B` and `B_neutral_A`, since their information was no longer relevant for our purpose.

– The ASSIN-1 corpus annotates 5000 sentence pairs with `None`, `Entailment` and `Paraphrase`. We preprocessed it from *XML* to *CSV* and converted `None` to `NEUTRAL`, `Entailment` to `ENTAILMENT` and `Paraphrase` to `PARAPHRASE`, so we could join it with the preprocessed SICK_BR corpus.

The resulting dataset contained 60.8% of the sentence pairs `NEUTRAL`, 9.6% `CONTRADICTION`, 18.4% `ENTAILMENT` and 11.2% `PARAPHRASE`.

To compute the semantic similarity between a gold reply and a given utterance, we tested two approaches: Spacy and BERT.

Spacy has a `similarity`[3] method that computes a cosine similarity over word vectors. There are three models available for the Portuguese language: small (`pt_core_news_sm`), medium (`pt_core_news_md`) and large (`pt_core_news_lg`). Both medium and large models include word vectors, while the small one does not include this feature. The difference between them is the number of unique vectors: the medium model has 20000 and the large one has 500000 unique vectors. Thus, two settings were tested using spacy: with the medium (`spacy md`) and the large (`spacy lg`) models.

As previously seen, BERT's output consists of the word embedding of each token, plus an embedding for an extra token, `[CLS]`, representing the whole sentence and used for sentence classification purposes. To test BERT to compute the semantic similarity between two sentences, two approaches were used: compute the cosine similarity over the `[CLS]` tokens of the two sentences (`BERT cls`), and compute the same similarity over the average of the embeddings of the words from each sentence (`BERT avg`).

Each setting was ran on the described inference dataset, and then averaged over each label. The results are shown in Table 6.2.

[3] https://spacy.io/usage/vectors-similarity (Last accessed on: 30/11/2020)

Setting	ENTAILMENT	NEUTRAL	PARAPHRASE	CONTRADICTION
Spacy md	0.7907	0.6904	0.8582	0.7802
Spacy lg	0.7891	0.6874	0.8571	0.7773
BERT cls	0.9698	0.9599	0.9817	0.9776
BERT avg	0.8528	0.7819	0.9009	0.8709

Table 6.2: Similarity by label

Given the settings results, we chose the one that better differentiated the labels. The BERT cls setting has very high and close results, so we excluded it. From the remaining, our intuition is that sim(NEUTRAL) $<$ sim(CONTRADICTION) $<$ sim(ENTAILMENT) $<$ sim(PARAPHRASE). The BERT avg setting has sim(CONTRADICTION) $>$ sim(ENTAILMENT), so we excluded it. Between the remaining, Spacy md and Spacy lg, as no significant difference was seen between them, we chose Spacy md as it is a smaller model.

Using the selected method to compute semantic similarity, spacy.similarity() using the medium model, our approach consists on selecting 100 random utterances from the dialog corpus and compute the similarity between each of them and the gold reply. Then, we order them by descending order of similarity. To select the one that is most similar to the gold reply, but without being a paraphrase of it, we use the results shown in Table 6.2, where the average similarity between paraphrases is 0.8582. We choose a superior threshold slightly lower than that, namely 0.83. Thus, from the set of ordered distractors, we select the first one whose similarity with the gold reply is less or equal than 0.83.

- **Find the best** BertForNextSentencePrediction **fine-tuning setting**

  As in the previous approach, we make experiments with 4 epochs of train and then select the one with higher average accuracy. The experiments consist on creating five different datasets, which will always be different because the distractor chosen for each gold reply is the one most similar to the gold reply but not too similar from a set of 100 randomly sampled utterances (as explained in Section 5.3). Then, for each dataset, we split it into training and validation sets, and fine-tune the model with the training set, computing the accuracy at the end of each epoch. We repeat this four times for each, in order to obtain enough results to measure if they have statistical significance.

  We repeat the aforementioned process for datasets created with *random* distractors, and testing the models with tailored distractors, to see if selecting tailored distractors in training improves the results in testing, compared to selecting them randomly.

  Table 6.3 shows the results, averaged over all the results obtained as described above, and their statistical significance. We observe that training the model with the tailored distractors improves, on average, 3% to 4% compared to training with the random ones. From Figure 6.4, built with the

data from Table 6.3, we observe that the results are, on average, better in the 4th epoch. The original results can be consulted in A.1.

Epoch	Random	Tailored	*p-value*	Significant
1	0.50314 ± 0.00634	**0.53662** ± 0.00593	0.00003	Yes
2	0.50296 ± 0.00438	**0.54258** ± 0.00784	0.00005	Yes
3	0.50452 ± 0.00497	**0.54462** ± 0.00883	0.00009	Yes
4	0.50626 ± 0.00544	**0.54578** ± 0.00950	0.00014	Yes

Table 6.3: Context agent random and tailored train, tailored test

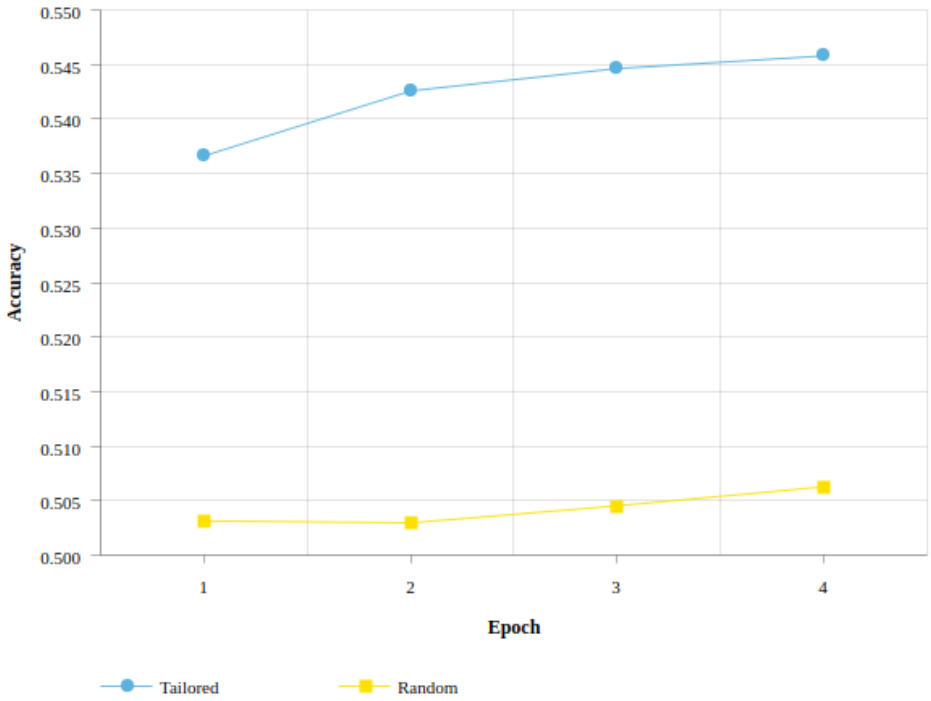

Figure 6.4: Accuracy by epoch, random vs tailored

6.2.1.D Top Ranking

To select distractors using the top ranking approach, we use the DialoGP3T model, that does both ranking and generation, allowing us to observe the impact of our distractors in both. As seen in Section 2.4, this model requires a pre-trained GPT-2 model, namely DialoGPT for the English language. However, there is no multilingual DialoGPT model, nor one for the Portuguese language; the only one available is `pierreguillou/gpt2-small-portuguese`[4]. Since this model was not fine-tuned for dialogue and has 37.99% accuracy after 5 epochs of training, we decided not to use it and only test the top ranking approach with English data.

[4]https://huggingface.co/pierreguillou/gpt2-small-portuguese (Last accessed on: 12/12/2020)

6.2.1.E Distractors used in contextual agent

In the previous Sections, we verified that fine-tuning `BertForNextSentencePrediction` using either semantic similarity chosen or Whoosh chosen distractors had better results than with random ones, with statistical significance. In both settings, the best accuracy was obtained after training 4 epochs, with the semantic similarity setting having an average of 54.578% accuracy and the Whoosh one having 55.678%. Since the Whoosh setting is around 1% better, and the contextual agent will use Whoosh to retrieve candidates, we choose to use the model trained with Whoosh distractors.

Summing up, our `BertForNextSentencePrediction` model used in the contextual agent is fine-tuned with:

- Whoosh distractors
- batch size = 16
- learning rate = 2e-5
- number of epochs = 4

6.2.2 English

Regarding the **semantic similarity** approach, for Portuguese it was done using SICK_BR and Assin data and the spacy similarity method. However, for the English language, using the MultiNLI corpus (Williams et al., 2018), labeled with `entailment`, `neutral` and `contradiction` relations, and the MSR paraphrase corpus [5], containing sentence-pairs labeled as `paraphrases`, the tested method for computing semantic similarity, spacy, with the English medium model, did not differentiate the labels as expected, with a `neutral` average of 0.78725 $\pm$ 0.087, `entailment` of 0.7746 $\pm$ 0.0925, `contradiction` of 0.7811 $\pm$ 0.0910, and `paraphrase` of 0.95675 $\pm$ 0.03, when we expected a more significant difference between the first three. Furthermore, when computing the average similarity of PersonaChat's responses with random distractors from the corpus, the result was 0.93, which, from the attained similarity averages, would suggest that all the utterances from the corpus were paraphrases, which does not make sense. Therefore, we decided not to use the semantic similarity approach for the English language.

To evaluate the impact of selecting tailored distractors, using the three remaining approaches, in ranking and generation, we use four versions of the PersonaChat dataset:

- `R` (baseline) – original PersonaChat with **random** distractors
- `N` – **noisy** PersonaChat

[5] https://www.microsoft.com/en-us/download/details.aspx?id=52398 (Last accessed on: 12/12/2020)

- `W` – Search engine (**Whoosh**) PersonaChat, built by creating a Whoosh index of question-answer pairs from the original corpus and retrieving four distractors for each gold reply
- `T` – **top-rank** PersonaChat, built using the top-ranking method with our DialoGP3Tmodel

Each of these datasets is used to train DialoGP3T, resulting in four different models: `R`, `N`, `W` and `T`.

To have results with statistical significance, we trained the aforementioned models with five different seeds. We tested them with testing data made of random (Table 6.4) and tailored distractors (both `T` and `W`, Table 6.5). The original tables with values across all seeds are shown in A.4 and A.5. Note that these testing sets have the same gold replies and history; only the distractors are different. Namely, the random testing set has 19 distractors, and both the tailored ones only have 4. Thus, since distractor selection does not affect generative results, only the ranking results change and, since in tailored test sets there are only 5 candidates, the Hits@5 and Hits@10 metrics are always 1, which explains why only the Hits@1 results are shown in Table 6.5.

Regarding the principles of design studied in Section 3.5, we ensure *Replication* by running the same setting five times with different seeds; we ensure *Blocking* by always testing with the same testing sets, whether random or tailored ones; finally, we ensure *Randomization* by running the settings in a random order across seeds and testing sets.

Metric	R	N	W	T
Hits@1	0.81882 ± 0.00513	0.05223 ± 0.01683	0.75448 ± 0.04487	**0.83476** ± 0.00455
Hits@5	**0.97736** ± 0.00126	0.27192 ± 0.06292	0.96324 ± 0.00881	0.97702 ± 0.00125
Hits@10	**0.99644** ± 0.00036	0.52522 ± 0.07984	0.99248 ± 0.00160	0.99514 ± 0.00055
BLEU	2.62674 ± 0.15890	**2.90554** ± 0.11438	2.70748 ± 0.15361	2.67988 ± 0.10051
TER	1.035 ± 0.01390	1.0419 ± 0.02092	1.041 ± 0.02826	**1.02646** ± 0.00895
BertScore	0.84874 ± 0.01223	**0.85576** ± 0.00117	0.8555 ± 0.00103	0.85468 ± 0.00119

Table 6.4: Mean and stdev by metric and training set for *random* testing

Test set	R	N	W	T
W	0.74712 ± 0.00214	0.20520 ± 0.03418	**0.80614** ± 0.02922	0.77470 ± 0.03960
T	0.82404 ± 0.00413	0.20408 ± 0.03762	0.75388 ± 0.00444	**0.84582** ± 0.00343

Table 6.5: Seed variation Hits@1 results (`T` and `W` test)

We observe that, regarding the Hits@1 metric, the model trained with the top-rank distractors has the best results. Regarding the Hits@5 and Hits@10 metrics, the model trained with random distractors shows the best results. Regarding the generation metrics, the model trained with noisy distractors surprisingly shows the best results, the BLEU and BertScore metrics.

Looking at the tailored testing results (Table 6.5), we observe that the setting with best results is the one whose training set contains distractors selected the same way as in the testing set, both for top-rank and Whoosh.

To assess the significance of these results, for each metric, we gather the five different results, one by seed, of each model. Then, to compare two models, we calculate the *p-value* using their corresponding results. If *p-value* < 0.05, we consider the result to be *significant*. We do this to assess if the following hypothesis are true: the T model has best Hits@1 result for R testing; the N model has best BLEU and BertScore results for R testing; the T model has best Hits@1 result for T testing; and the W model has best Hits@1 result for W testing. The results are shown in Table 6.6, where X > Y for Z stands for the hypothesis that model X performs better than model Y on test set Z, using the metric specified in column Metric.

Hypos	Metric	Original values	Test values	p-value	Significant
T > R for R	Hits@1	0.81882 ± 0.00513	**0.83476** ± 0.00455	0.00086	Yes
N > R for R	BLEU	2.62674 ± 0.15890	**2.90554** ± 0.11438	0.01465	Yes
N > R for R	BertScore	0.84874 ± 0.01223	**0.85576** ± 0.00117	0.26942	No
T > R for T	Hits@1	0.82404 ± 0.00413	**0.84582** ± 0.00343	0.00002	Yes
W > R for W	Hits@1	0.74712 ± 0.00214	**0.80614** ± 0.02922	0.01052	Yes
W > T for W	Hits@1	0.75388 ± 0.00444	**0.80614** ± 0.02922	0.01536	Yes

Table 6.6: Seed variation results (T and W test)

The only hypothesis that is not statistical significant is *the N model has better BertScore results for R testing*, which means that the improvement observed may be by chance. All the other hypothesis are statistically significant, namely:

1. the T model has better Hits@1 results than the R model for R testing;
2. the N model has better BLEU results than the R model for R testing;
3. the T model has better Hits@1 results than the R model for T testing;
4. the W model has better Hits@1 results than the R and T models for W testing.

From 3. and 4., we conclude that **for scenarios with strong distractors, training a model using strong distractors *generated using the same heuristic* is a better option than using random ones**.

6.3 Case study: customer support

Since this work was developed under the scope of project MAIA: Multilingual AI Agent Assistants[6], whose goal is to develop a platform where AI agents perform customer support, we also use a customer support dataset to test our tailored distractors.

[6]https://resources.unbabel.com/maia-unbabel-research (Last accessed on: 23/11/2020)

We used Twitter's Customer Support dataset[7]. It contains data from multiple companies, such as Apple, American Airlines and Xbox. For the purpose of our experiments, we only focused on the Xbox dataset.

We preprocessed the data to match the *json* structure required by the DialoGP3T model. One problem we found was the use of Twitter mentions, where all utterances started with one, or more, in the case where one user replies to multiple users, and these would not add useful information to our model. Thus, we removed all the mentions from the beginning of the utterances, and covered the remaining with `MENTION`. We also covered all URLs with `URL`. Examples taken from the corpora, before and after preprocessing, are shown in Table 6.7.

Before	After
"@XboxSupport Unless you do not provide any evidence then I would like my ban to be removed"	"Unless you do not provide any evidence then I would like my ban to be removed"
"@XboxSupport @784280 Happened to a friend of mine too..doesn't look good.. https://t.co/R1ll5kwN0A"	"Happened to a friend of mine too..doesn't look good.. URL"

Table 6.7: Before and after preprocessing

Tables 6.8 and 6.9 show the average results across five different seeds obtained for a test set with, respectively, randomly and top-rank chosen distractors. Since the number of candidates was 10 for the random test set and 5 for the tailored test set, we omit the Hits@10 metric for the first and also the Hits@5 metric for the latter, which are always 1. Also, as mentioned in previous experiments, the generative scores are independent of the distractors, thus are not shown in the second table. The original results before averaging are in Tables A.6 and A.7.

Metric	R	T	*p-value*	Significant
Hits@1	0.73362 $\pm$ 0.00913	**0.78172** $\pm$ 0.00927	0.00003	Yes
Hits@5	0.99134 $\pm$ 0.00205	**0.99324** $\pm$ 0.00084	0.11002	No
BLEU	**11.12594** $\pm$ 0.22760	11.00848 $\pm$ 0.23538	0.44566	No
TER	**1.0243** $\pm$ 0.01417	1.02768 $\pm$ 0.01648	0.73721	No
BertScore	0.8514 $\pm$ 0.00547	**0.85324** $\pm$ 0.00158	0.50423	No

Table 6.8: Xbox average results (random test)

Metric	R	T	*p-value*	Significant
Hits@1	0.73504 $\pm$ 0.00920	**0.78364** $\pm$ 0.00908	0.00003	Yes

Table 6.9: Xbox average results (tailored test)

We observe that, both for random and tailored testing, using the model trained with the top-rank

[7]https://www.kaggle.com/thoughtvector/customer-support-on-twitter (Last accessed on: 09/12/2020)

distractors improves the Hits@1 metric by almost 5%, with statistical significance. This reinforces the importance of selecting tailored distractors.

6.4 Do current language models use the context of the conversation?

In Section 3.5.4, it was seen that Khandelwal et al. (2018) and Sankar et al. (2019) showed that neural models were not using the context of the conversation as expected, by performing an ablation study that showed that the models' performance was not affected by drastic changes in the context.

However, with the fast development of NLP, many new models have emerged since those studies were made. Particularly, we want to show that our DialoGP3T model is using the context of the conversation, and, therefore, is sensitive to changes in the context. With that goal, we perform an ablation study where we introduce perturbations in the context to assess whether the performance metrics change. Here, we assume that the context does not include the most recent utterance, since it is the one to which the model will generate a response. In Listing 6.4 a randomly selected context from the PersonaChat corpus is shown.

Listing 6.4: Original history

```
'history':[
    "__ SILENCE __",
    "hi friend . let us talk .",
    "how are you doing tonight ?",
    "great . just working on my art and feeding my pets . what about you ?",
    "well i'm albert thanks for asking and art is fun",
    "i love canada , i am a great painter , my feline friends help me out too .",
    "just a nice guy really just i been told",
    "good for you . i hope you do not ea seafood . it isn't good .",
    "talk faster and sometimes i cant sleep phone rings all the time",
    "my cats always call me , i've too many .",
    "well people always sue everyone i'm the one to call"
]
```

Since the goal is to study the perturbations' impact independently, for each of them a new version of the corpus is produced, where the only difference is the context. The perturbations are the following:

- **No context** – delete all the utterances in the context.

Listing 6.5: No context history

```
'history':[
    ”well people always sue everyone i'm the one to call”
]
```

- **Half context** – randomly delete half of the utterances in the context.

Listing 6.6: Half context history

```
'history':[
    ”__ SILENCE __”,
    ”hi friend . let us talk .”,
    ”great . just working on my art and feeding my pets . what about you ?”,
    ”well i'm albert thanks for asking and art is fun”,
    ”i love canada , i am a great painter , my feline friends help me out too .”,
    ”well people always sue everyone i'm the one to call”
]
```

- **Shuffle context** – shuffle all the utterances in the context.

Listing 6.7: Shuffle context history

```
'history':[
    ”just a nice guy really just i been told”,
    ”hi friend . let us talk .”,
    ”talk faster and sometimes i cant sleep phone rings all the time”,
    ”great . just working on my art and feeding my pets . what about you ?”,
    ”i love canada , i am a great painter , my feline friends help me out too .”,
    ”good for you . i hope you do not ea seafood . it isn't good .”,
    ”well i'm albert thanks for asking and art is fun”,
    ”__ SILENCE __”,
    ”how are you doing tonight ?”,
    ”my cats always call me , i've too many .”,
    ”well people always sue everyone i'm the one to call”
]
```

Using the new three corpora, we train three versions of DialoGP3T : `no context model`, `half context model` and `shuffle context model`. These models are trained through 1 epoch, and with five different seeds, to assess their results' statistical significance. They are then tested using the PersonaChat validation corpus, with the original context.

Table 6.10 shows the mean and standard deviation of each setting across the different seeds (the results by seed are in Appendix A.3), and the *p-value* obtained by performing a *ttest* on each setting with the original setting, in order to assess if the obtained values have statistically significance. We use $alpha = 0.05$, thus a result is significant if it has $p - value < 0.05$. `O` represents the `original model` (intact context), `N` the `no context model`, `H` the `half context model` and `S` the `shuffle context model`. We observe that the only setting with significant results across all metrics is `No Context model` (`N`), whose results are significantly worse than those of the original model, namely, with a Hits@1 of 0.6142, compared to the original's 0.8188. The two other settings also show significantly worse results for the Hits@1 metric, but only approximately less 0.01 than the original; their BLEU results are significantly better than the original ones.

Metric	Dataset	Mean ± StDev	p-value	Significant
Hits@1	O	0.8188 ± 0.00513	-	-
	N	0.6142 ± 0.00894	3.4876x10^{-5}	Yes
	H	0.80632 ± 0.00581	0.00712	Yes
	S	0.8075 ± 0.00158	0.0060	Yes
Hits@5	O	0.97736 ± 0.00126	-	-
	N	0.88014 ± 0.00867	1.0944x10^{-5}	Yes
	H	0.97406 ± 0.00197	0.0167	Yes
	S	0.97502 ± 0.00227	0.0888	No
Hits@10	O	0.99644 ± 0.00036	-	-
	N	0.96332 ± 0.00382	3.885x10^{-5}	Yes
	H	0.99574 ± 0.00068	0.0892	No
	S	0.99566 ± 0.00082	0.1033	No
BLEU	O	2.62674 ± 0.15890	-	-
	N	1.47026 ± 0.21993	2.2732x10^{-5}	Yes
	H	2.92308 ± 0.10873	0.0106	Yes
	S	2.88062 ± 0.10773	0.0211	Yes
TER	O	1.035 ± 0.01390	-	-
	N	0.96830 ± 0.00735	7.2536x10^{-5}	Yes
	H	1.04446 ± 0.00679	0.2221	No
	S	1.03870 ± 0.01023	0.6456	No
BertScore	O	0.84874 ± 0.01223	-	-
	N	0.82348 ± 0.00921	0.00699	Yes
	H	0.85588 ± 0.00100	0.2624	No
	S	0.85554 ± 0.00121	0.2826	No

Table 6.10: Mean, stdev and p-value across seeds (Random vs each setting)

This experiment has shown that DialoGP3T **takes the context of a conversation into account when selecting a response**, since its ranking accuracy decreases around 20% when the context of the

conversation is removed from the dataset, thus demonstrating the model's robustness.

7

Conclusion and Future Work

Contents

7.1 Main contributions

In this thesis, we **joined two multi-agent systems** into one, and tackled two limitations. The first was that none of this agents allowed the use of paraphrases, synonyms and domain specific synonyms and acronyms, so a new agent architecture was proposed – the General Agent – which allows to specify paraphrases to questions in the corpus, define domain-specific synonyms and acronyms, and also consider synonyms when computing the distance between two sentences. It also allows to have different instances of that agent with different corpora. The second limitation was that none of the agents took the context of the conversation into account. To solve this and make it take **context** into consideration when selecting a response, a `BERT For Next Sentence Prediction` model was fine-tuned for dialogues, and used in the Contextual agent, to predict the probability of each candidate response being the most appropriate, given the context. To fine-tune the model, we proposed three techniques of **distractor selection**, instead of selecting them randomly: using a search engine, based on semantic similarity and based on a ranking model. These techniques were tested both on the `BERT For Next Sentence Prediction` model and on a DialoGPT based framework. Interesting conclusions were drawn, namely, that **training a model with tailored distractors improves the performance on testing with both random and tailored distractors** when compared to training with random distractors. We used the distractor selection technique that had the best results to select tailored distractors on a customer support dataset, where positive results were also observed. Finally, we performed an ablation study that showed that a current neural model is sensitive to changes in the context, by removing the whole context of each entry in the training corpus and observing a significant decrease in the model's performance when testing in a corpus with the full context, thus proving its robustness.

7.2 Future work

As future work, regarding the multi-agent dialogue system architecture, since a considerable number of modern systems are generative, it would be interesting to have generative agents, and Natural Language Understanding and Natural Language Generation modules in the system's architecture. It would also be interesting to explore MULTI-SSS's online learning, and possibly adapt it to also work with decision making strategies, as it currently only learns from agents, and train CHATTUGA's query classifiers to handle new labels, such as sentiments and intents, and handle them.

Regarding the usage of context, as previously seen, BERT only allows to represent 512 tokens. We decided to only consider the last 512 tokens of the context when matching it with a response. An alternative to consider a larger number of tokens would be to split the context into 512 token blocks and match each block with each response, sum all scores and select the response with a higher score. Context could also be used to solve linguistic phenomena, such as anaphora and ellipsis resolution.

It would also be interesting to, for the response selection task, fine-tune different models made available by HuggingFace, such as BertForQuestionAnswering[1] with a multilingual pre-trained model, to create an agent specialized in question answering in Portuguese, taking the context into account.

Concerning our tailored distractors, since we observed that they have a positive impact in different models' performance, it would be interesting to explore other ways of selecting them, other than the ones proposed in this thesis, such as generative approaches.

[1] https://huggingface.co/transformers/model$_doc/bert.html bertforquestionanswering(Lastaccessedon: 28/12/2020)$

Bibliography

E. Alpaydin. *Design and Analysis of Machine Learning Experiments*, pages 475–515. 2010.

D. Ameixa. Say Something Smart - ensinando um chatbot a responder com base em legendas de filmes. Master's thesis, Instituto Superior Técnico, Av. Rovisco Pais, 1049-001 Lisboa, 2015.

D. Ameixa, L. Coheur, and R. A. Redol. From subtitles to human interactions: introducing the subtle corpus. Technical report, Tech. rep., INESC-ID (November 2014), 2013.

J. A. Anderson and E. Rosenfeld, editors. *Neurocomputing: Foundations of Research*. MIT Press, Cambridge, MA, USA, 1988. ISBN 0262010976.

J. Araki, D. Rajagopal, S. Sankaranarayanan, S. Holm, Y. Yamakawa, and T. Mitamura. Generating questions and multiple-choice answers using semantic analysis of texts. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1125–1136, Osaka, Japan, Dec. 2016. The COLING 2016 Organizing Committee. URL https://www.aclweb.org/anthology/C16-1107.

A. Bapna, G. Tür, D. Hakkani-Tür, and L. Heck. Sequential dialogue context modeling for spoken language understanding. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 103–114, Saarbrücken, Germany, Aug. 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-5514. URL https://www.aclweb.org/anthology/W17-5514.

C.-Y. Chen, D. Yu, W. Wen, Y. M. Yang, J. Zhang, M. Zhou, K. Jesse, A. Chau, A. Bhowmick, S. Iyer, G. Sreenivasulu, R. Cheng, A. Bhandare, and Z. Yu. Gunrock: Building a human-like social bot by leveraging large scale real user data, 2018.

J. Chung, Ç. Gülçehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555, 2014. URL http://arxiv.org/abs/1412.3555.

C. Danescu-Niculescu-Mizil and L. Lee. Chameleons in imagined conversations: A new approach to understanding coordination of linguistic style in dialogs. In *Proceedings of the Workshop on Cognitive Modeling and Computational Linguistics, ACL 2011*, 2011.

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. 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 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://www.aclweb.org/anthology/N19-1423.

A. Elgohary, D. Peskov, and J. Boyd-Graber. Can you unpack that? learning to rewrite questions-in-context. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5917–5923, Hong Kong, China, Nov. 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1605. URL https://www.aclweb.org/anthology/D19-1605.

H. Fang, H. Cheng, M. Sap, E. Clark, A. Holtzman, Y. Choi, N. A. Smith, and M. Ostendorf. Sounding board: A user-centric and content-driven social chatbot. *CoRR*, abs/1804.10202, 2018. URL http://arxiv.org/abs/1804.10202.

M. G. Fernandes. Chattuga: A meta-chatbot for the Portuguese language. Master's thesis, Instituto Superior Técnico, Av. Rovisco Pais, 1049-001 Lisboa, 2019.

P. Fialho, L. Coheur, S. Curto, P. Cláudio, Â. Costa, A. Abad, H. Meinedo, and I. Trancoso. Meet EDGAR, a tutoring agent at MONSERRATE. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 61–66, Sofia, Bulgaria, Aug. 2013. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/P13-4011.

D. Fum, G. Guida, and C. Tasso. A distributed multi-agent architecture for natural language processing. In *Coling Budapest 1988 Volume 2: International Conference on Computational Linguistics*, 1988. URL https://www.aclweb.org/anthology/C88-2165.

L. Gao, K. Gimpel, and A. Jensson. Distractor analysis and selection for multiple-choice cloze questions for second-language learners. In *Proceedings of the Fifteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 102–114, Seattle, WA, USA â†' Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.bea-1.10. URL https://www.aclweb.org/anthology/2020.bea-1.10.

Y. Gao, P. Li, I. King, and M. R. Lyu. Interconnected question generation with coreference alignment and conversation flow modeling. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4853–4862, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1480. URL https://www.aclweb.org/anthology/P19-1480.

J.-C. Gu, Z.-H. Ling, and Q. Liu. Interactive matching network for multi-turn response selection in retrieval-based chatbots. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, CIKM '19, page 2321–2324, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450369763. doi: 10.1145/3357384.3358140. URL https://doi.org/10.1145/3357384.3358140.

C. Gunasekara, J. K. Kummerfeld, L. Polymenakos, and W. Lasecki. DSTC7 task 1: Noetic end-to-end response selection. In *Proceedings of the First Workshop on NLP for Conversational AI*, pages 60–67, Florence, Italy, Aug. 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-4107. URL https://www.aclweb.org/anthology/W19-4107.

M. Henderson, I. Vulić, D. Gerz, I. Casanueva, P. Budzianowski, S. Coope, G. Spithourakis, T.-H. Wen, N. Mrkšić, and P.-H. Su. Training neural response selection for task-oriented dialogue systems. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5392–5404, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1536. URL https://www.aclweb.org/anthology/P19-1536.

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, Nov. 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL http://dx.doi.org/10.1162/neco.1997.9.8.1735.

S. Jiang and J. Lee. Distractor generation for Chinese fill-in-the-blank items. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 143–148, Copenhagen, Denmark, Sept. 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-5015. URL https://www.aclweb.org/anthology/W17-5015.

D. Jurafsky and J. H. Martin. Speech and language processing, 2019. URL https://web.stanford.edu/~jurafsky/slp3/26.pdf.

U. Khandelwal, H. He, P. Qi, and D. Jurafsky. Sharp nearby, fuzzy far away: How neural language models use context. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 284–294, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1027. URL https://www.aclweb.org/anthology/P18-1027.

X. Kong, V. Gangal, and E. Hovy. SCDE: Sentence cloze dataset with high quality distractors from examinations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5668–5683, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.502. URL https://www.aclweb.org/anthology/2020.acl-main.502.

H. Kumar, A. Agarwal, and S. Joshi. Dialogue-act-driven conversation model : An experimental study. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1246–1256, Santa Fe, New Mexico, USA, Aug. 2018. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/C18-1106.

H. Kumar, A. Agarwal, and S. Joshi. A practical dialogue-act-driven conversation model for multi-turn response selection. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1980–1989, Hong Kong, China, Nov. 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1205. URL https://www.aclweb.org/anthology/D19-1205.

Z. Lin, D. Cai, Y. Wang, X. Liu, H. Zheng, and S. Shi. The world is not binary: Learning to rank with grayscale data for dialogue response selection. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9220–9229, Online, Nov. 2020. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/2020.emnlp-main.741.

R. Lowe, N. Pow, I. Serban, and J. Pineau. The Ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. In *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 285–294, Prague, Czech Republic, Sept. 2015. Association for Computational Linguistics. doi: 10.18653/v1/W15-4640. URL https://www.aclweb.org/anthology/W15-4640.

W. Ma, Y. Cui, N. Shao, S. He, W.-N. Zhang, T. Liu, S. Wang, and G. Hu. TripleNet: Triple attention network for multi-turn response selection in retrieval-based chatbots. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 737–746, Hong Kong, China, Nov. 2019a. Association for Computational Linguistics. doi: 10.18653/v1/K19-1069. URL https://www.aclweb.org/anthology/K19-1069.

W. Ma, Y. Cui, N. Shao, S. He, W.-N. Zhang, T. Liu, S. Wang, and G. Hu. TripleNet: Triple attention network for multi-turn response selection in retrieval-based chatbots. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 737–746, Hong Kong, China, Nov. 2019b. Association for Computational Linguistics. doi: 10.18653/v1/K19-1069. URL https://www.aclweb.org/anthology/K19-1069.

V. Mendonça, F. S. Melo, L. Coheur, and A. Sardinha. A conversational agent powered by online learning. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '17, page 1637–1639, Richland, SC, 2017. International Foundation for Autonomous Agents and Multiagent Systems.

T. Mikolov, K. Chen, G. S. Corrado, and J. Dean. Efficient estimation of word representations in vector space. In *ICLR*, 2013.

G. A. Miller. Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41, Nov. 1995. ISSN 0001-0782. doi: 10.1145/219717.219748. URL https://doi.org/10.1145/219717.219748.

R. Mitkov. Anaphora resolution: The state of the art. In *ACL 2007*, 2007.

R. Mitkov and L. A. Ha. Computer-aided generation of multiple-choice tests. In *Proceedings of the HLT-NAACL 03 Workshop on Building Educational Applications Using Natural Language Processing*, pages 17–22, 2003. URL https://www.aclweb.org/anthology/W03-0203.

R. Mitkov, L. A. Ha, A. Varga, and L. Rello. Semantic similarity of distractors in multiple-choice tests: Extrinsic evaluation. In *Proceedings of the Workshop on Geometrical Models of Natural Language Semantics*, pages 49–56, Athens, Greece, Mar. 2009. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W09-0207.

P. Mota. LUP: A Language Understanding Platform. Master's thesis, Instituto Superior Técnico, Av. Rovisco Pais, 1049-001 Lisboa, 2015.

Y. Ohsugi, I. Saito, K. Nishida, H. Asano, and J. Tomita. A simple but effective method to incorporate multi-turn context with BERT for conversational machine comprehension. In *Proceedings of the First Workshop on NLP for Conversational AI*, pages 11–17, Florence, Italy, Aug. 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-4102. URL https://www.aclweb.org/anthology/W19-4102.

A. Papangelis, Y.-C. Wang, P. Molino, and G. Tur. Collaborative multi-agent dialogue model training via reinforcement learning. In *Proceedings of the 20th Annual SIGdial Meeting on Discourse and Dialogue*, pages 92–102, Stockholm, Sweden, Sept. 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-5912. URL https://www.aclweb.org/anthology/W19-5912.

K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, page 311–318, USA, 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073135. URL https://doi.org/10.3115/1073083.1073135.

A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners. 2019.

L. Real, A. Rodrigues, and A. Vieira. Sick-br: A portuguese corpus for inference: 13th international conference, propor 2018, canela, brazil, september 24–26, 2018, proceedings, 01 2018.

C. Sankar, S. Subramanian, C. Pal, S. Chandar, and Y. Bengio. Do neural dialog systems use the conversation history effectively? an empirical study. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 32–37, Florence, Italy, July 2019. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/P19-1004.

J. Santos. Say Something Smart 3.0:A Multi-Agent Chatbot in Open Domain. Master's thesis, Instituto Superior Técnico, Av. Rovisco Pais, 1049-001 Lisboa, 2019.

S. Sato, R. Akama, H. Ouchi, J. Suzuki, and K. Inui. Evaluating dialogue generation systems via response selection, 2020.

S. Sheikholeslami. Ablation programming for machine learning. Master's thesis, KTH Royal Institute of Technology, Brinellvägen 8, 114 28 Stockholm, Sweden, 2019.

W. Shi and V. Demberg. Next sentence prediction helps implicit discourse relation classification within and across domains. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5790–5796, Hong Kong, China, Nov. 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1586. URL https://www.aclweb.org/anthology/D19-1586.

M. Snover, B. Dorr, R. Schwartz, L. Micciulla, and J. Makhoul. A study of translation edit rate with targeted human annotation. USA, 2006. Association for Machine Translation in the Americas. URL https://www.cs.umd.edu/~snover/pub/amta06/ter_amta.pdf.

C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu. A survey on deep transfer learning, 2018.

K. Tanaka, J. Takayama, and Y. Arase. Dialogue-act prediction of future responses based on conversation history. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 197–202, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-2027. URL https://www.aclweb.org/anthology/P19-2027.

W. L. Taylor. cloze procedure: A new tool for measuring readability, 1953.

H. P. Truong, P. Parthasarathi, and J. Pineau. MACA: A modular architecture for conversational agents. In *Proc 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 93–102, Saarbrücken, Germany, Aug. 2017. ACL.

L. A. Tuan, D. J. Shah, and R. Barzilay. Capturing greater context for question generation, 2019. URL https://arxiv.org/abs/1910.10274.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc., 2017. URL http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf.

H. Wang, Z. Lu, H. Li, and E. Chen. A dataset for research on short-text conversations. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 935–945, Seattle, Washington, USA, Oct. 2013. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/D13-1096.

Z. Wang, E. Rastorgueva, W. Lin, and X. Wu. No, you're not alone: A better way to find people with similar experiences on Reddit. In *Proceedings of the 5th Workshop on Noisy User-generated Text (W-NUT 2019)*, pages 307–315, Hong Kong, China, Nov. 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-5540. URL https://www.aclweb.org/anthology/D19-5540.

A. Williams, N. Nangia, and S. Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1101. URL https://www.aclweb.org/anthology/N18-1101.

T. Wolf, V. Sanh, J. Chaumond, and C. Delangue. Transfertransfo: A transfer learning approach for neural network based conversational agents. *CoRR*, abs/1901.08149, 2019. URL http://arxiv.org/abs/1901.08149.

Y. Wu, W. Wu, C. Xing, M. Zhou, and Z. Li. Sequential matching network: A new architecture for multi-turn response selection in retrieval-based chatbots. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 496–505, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1046. URL https://www.aclweb.org/anthology/P17-1046.

C. Yuan, W. Zhou, M. Li, S. Lv, F. Zhu, J. Han, and S. Hu. Multi-hop selector network for multi-turn response selection in retrieval-based chatbots. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 111–120, Hong Kong, China, Nov. 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1011. URL https://www.aclweb.org/anthology/D19-1011.

R. Zhang, H. Lee, L. Polymenakos, and D. R. Radev. Addressee and response selection in multi-party conversations with speaker interaction rnns. *CoRR*, abs/1709.04005, 2017. URL http://arxiv.org/abs/1709.04005.

T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi. Bertscore: Evaluating text generation with BERT. *CoRR*, abs/1904.09675, 2019a. URL http://arxiv.org/abs/1904.09675.

X. Zhang, C. Li, D. Yu, S. Davidson, and Z. Yu. Filling conversation ellipsis for better social dialog understanding. *CoRR*, abs/1911.10776, 2019b. URL http://arxiv.org/abs/1911.10776.

Y. Zhang, S. Sun, M. Galley, Y. Chen, C. Brockett, X. Gao, J. Gao, J. Liu, and B. Dolan. Dialogpt: Large-scale generative pre-training for conversational response generation. *CoRR*, abs/1911.00536, 2019c. URL http://arxiv.org/abs/1911.00536.

X. Zhou, D. Dong, H. Wu, S. Zhao, D. Yu, H. Tian, X. Liu, and R. Yan. Multi-view response selection for human-computer conversation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 372–381, Austin, Texas, Nov. 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1036. URL https://www.aclweb.org/anthology/D16-1036.

X. Zhou, L. Li, D. Dong, Y. Liu, Y. Chen, W. X. Zhao, D. Yu, and H. Wu. Multi-turn response selection for chatbots with deep attention matching network. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1118–1127, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1103. URL https://www.aclweb.org/anthology/P18-1103.

F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He. A comprehensive survey on transfer learning. *CoRR*, abs/1911.02685, 2019. URL http://arxiv.org/abs/1911.02685.

Context tables

Training set	Epoch	Run 1	Run 2	Run 3	Run 4	Avg
Tailored	1	0.535	0.524	0.534	0.538	0.5328
	2	0.533	0.535	0.534	0.541	0.5358
	3	0.535	0.535	0.535	0.535	0.535
	4	0.535	0.534	0.534	0.535	0.5345
	1	0.529	0.527	0.540	0.531	0.5318
	2	0.534	0.531	0.539	0.538	0.5355
	3	0.535	0.535	0.538	0.537	0.5363
	4	0.537	0.534	0.542	0.536	0.5373
	1	0.538	0.539	0.537	0.536	0.5375
	2	0.548	0.552	0.546	0.549	0.548
	3	0.554	0.547	0.547	0.548	0.549
	4	0.554	0.553	0.551	0.552	0.5525
	1	0.533	0.541	0.526	0.538	0.5345
	2	0.535	0.541	0.537	0.548	0.5403
	3	0.545	0.550	0.543	0.550	0.547
	4	0.544	0.554	0.543	0.552	0.5483
	1	0.543	0.554	0.543	0.546	0.5465
	2	0.554	0.558	0.552	0.549	0.5533
	3	0.557	0.554	0.558	0.554	0.5558
	4	0.555	0.557	0.558	0.555	0.5563
Random	1	0.531	0.502	0.506	0.511	0.5125
	2	0.517	0.491	0.511	0.507	0.5065
	3	0.524	0.503	0.516	0.507	0.5125
	4	0.533	0.503	0.513	0.508	0.5143
	1	0.491	0.501	0.513	0.516	0.5053
	2	0.495	0.498	0.526	0.515	0.5085
	3	0.490	0.502	0.515	0.504	0.5028
	4	0.500	0.500	0.529	0.509	0.5095
	1	0.503	0.488	0.496	0.494	0.4953
	2	0.502	0.493	0.505	0.502	0.5005
	3	0.508	0.500	0.492	0.497	0.4993
	4	0.505	0.503	0.501	0.500	0.5023
	1	0.509	0.496	0.520	0.478	0.5008
	2	0.500	0.494	0.496	0.502	0.498
	3	0.501	0.506	0.513	0.490	0.5025
	4	0.496	0.503	0.512	0.499	0.5025
	1	0.496	0.491	0.506	0.514	0.5018
	2	0.498	0.488	0.498	0.521	0.5013
	3	0.504	0.492	0.511	0.515	0.5055
	4	0.502	0.494	0.507	0.504	0.5018

Table A.1: Context agent random and tailored train, tailored test

Training set	Epoch	Run 1	Run 2	Run 3	Run 4	Avg
Whoosh	1	0.551	0.543	0.549	0.545	0.5470
	2	0.556	0.552	0.552	0.551	0.5528
	3	0.559	0.549	0.557	0.553	0.5545
	4	0.561	0.551	0.559	0.553	0.556
	1	0.557	0.548	0.554	0.556	0.5538
	2	0.561	0.555	0.561	0.560	0.5593
	3	0.559	0.555	0.558	0.560	0.558
	4	0.560	0.557	0.561	0.558	0.559
	1	0.555	0.545	0.547	0.550	0.5493
	2	0.548	0.541	0.547	0.551	0.5468
	3	0.554	0.546	0.552	0.551	0.5508
	4	0.554	0.547	0.552	0.548	0.5503
	1	0.555	0.556	0.549	0.547	0.5518
	2	0.550	0.563	0.553	0.559	0.5563
	3	0.552	0.566	0.555	0.563	0.5590
	4	0.553	0.566	0.560	0.564	0.5608
	1	0.563	0.554	0.524	0.556	0.5493
	2	0.562	0.558	0.556	0.545	0.5553
	3	0.559	0.556	0.555	0.552	0.5555
	4	0.563	0.557	0.556	0.555	0.5578
Random	1	0.491	0.483	0.484	0.507	0.4913
	2	0.473	0.485	0.500	0.505	0.4908
	3	0.471	0.486	0.494	0.507	0.4895
	4	0.485	0.484	0.502	0.507	0.4945
	1	0.503	0.505	0.500	0.492	0.5
	2	0.491	0.506	0.503	0.482	0.4955
	3	0.491	0.507	0.515	0.484	0.4933
	4	0.490	0.498	0.500	0.484	0.4930
	1	0.505	0.489	0.511	0.506	0.5028
	2	0.514	0.499	0.474	0.495	0.4955
	3	0.519	0.519	0.488	0.492	0.5045
	4	0.512	0.515	0.486	0.488	0.5003
	1	0.506	0.491	0.494	0.506	0.4993
	2	0.503	0.492	0.465	0.496	0.4890
	3	0.509	0.485	0.471	0.502	0.4918
	4	0.504	0.499	0.475	0.504	0.4955
	1	0.473	0.479	0.465	0.500	0.4793
	2	0.491	0.495	0.474	0.492	0.4880
	3	0.481	0.494	0.474	0.504	0.4883
	4	0.484	0.497	0.480	0.496	0.4893

Table A.2: Context agent random and Whoosh train, Whoosh test

Seed	TS	Hits@1	Hits@5	Hits@10	BLEU	TER	BertScore
3	O	0.8241	0.9783	0.9969	2.7408	1.0322	0.8533
	N	0.6006	0.8741	0.9612	1.3717	0.9633	0.8252
	H	0.8054	0.975	0.9959	2.7659	1.035	0.8569
	S	0.8103	0.9751	0.9955	2.6927	1.0231	0.8565
5	O	0.8134	0.9754	0.9967	2.6500	1.0493	0.8536
	N	0.6149	0.8764	0.9609	1.1377	0.976	0.8218
	H	0.8092	0.9737	0.9953	2.9346	1.0502	0.8545
	S	0.8063	0.9723	0.9962	2.7745	1.0308	0.8537
8	O	0.8135	0.9786	0.9962	2.5859	1.0226	0.8269
	N	0.6257	0.8923	0.9690	1.5276	0.9599	0.8093
	H	0.8116	0.9758	0.9967	3.052	1.0401	0.8558
	S	0.8078	0.9787	0.9967	2.9552	1.0381	0.855
10	O	0.8200	0.9774	0.9964	2.3771	1.0212	0.8551
	N	0.6140	0.8719	0.9601	1.6533	0.9759	0.8346
	H	0.7967	0.9708	0.9949	2.8781	1.0508	0.8554
	S	0.8064	0.9727	0.9945	2.7626	1.024	0.8545
12	O	0.8231	0.9771	0.9960	2.7799	1.0497	0.8548
	N	0.6157	0.886	0.9655	1.661	0.9664	0.8265
	H	0.8087	0.975	0.9959	2.9848	1.0462	0.8568
	S	0.8049	0.9753	0.9962	2.7713	1.0238	0.8551

Table A.3: Seed variation, context ablation results

Seed	TS	Hits@1	Hits@5	Hits@10	BLEU	TER	BertScore
3	R	0.8241	**0.9783**	**0.9969**	2.7408	**1.0322**	0.8533
	T	**0.8337**	0.9774	0.9959	2.6839	1.0329	0.8533
	W	0.7221	0.9556	0.9909	2.6861	1.0556	**0.8557**
	N	0.0415	0.2082	0.4478	**2.7529**	1.0629	0.8550
5	R	0.8134	**0.9754**	**0.9967**	2.6500	1.0493	0.8536
	T	**0.8284**	0.9750	0.9946	2.6314	**1.0329**	0.8538
	W	0.7439	0.9628	0.9927	**2.9579**	1.0804	**0.8562**
	N	0.0347	0.1992	0.4296	2.8977	1.0333	0.8541
8	R	0.8135	**0.9786**	**0.9962**	2.5859	1.0226	0.8269
	T	**0.8381**	0.9781	0.9953	2.678	1.0241	0.8547
	W	0.7286	0.9565	0.9918	2.6629	1.0398	0.8537
	N	0.0701	0.3238	0.5943	**2.9171**	**1.0116**	**0.8564**
10	R	0.8200	**0.9774**	**0.9964**	2.3771	1.0212	0.8551
	T	**0.8335**	0.9767	0.9953	2.5671	1.0118	0.8553
	W	0.7450	0.9637	0.9919	2.5371	**1.0105**	0.8561
	N	0.0690	0.3247	0.5868	**2.8858**	1.0420	**0.8564**
12	R	0.8231	0.9771	**0.9960**	2.7799	1.0497	0.8548
	T	**0.8401**	0.9769	0.9946	2.8390	1.0306	0.8563
	W	0.8328	**0.9776**	0.9951	2.6934	**1.0187**	0.8558
	N	0.0626	0.3037	0.5676	**3.0742**	1.0597	**0.8569**

Table A.4: Seed variation results (random test)

Seed	TS	Hits@1 T	Hits@1 W
3	R	0.8282	0.75
	T	**0.8441**	0.7602
	W	0.7509	**0.8208**
	N	0.1775	0.1676
5	R	0.8217	0.745
	T	**0.8410**	0.753
	W	0.7656	**0.8191**
	N	0.1591	0.1690
8	R	0.8180	0.7450
	T	**0.8490**	0.7499
	W	0.7503	**0.8187**
	N	0.2275	0.2260
10	R	0.8264	0.7480
	T	**0.8459**	0.7564
	W	0.7622	**0.8182**
	N	0.2529	0.2392
12	R	0.8259	0.7476
	T	**0.8491**	0.7499
	W	0.8445	**0.7539**
	N	0.2034	0.2242

Table A.5: Seed variation results (T and W test)

Seed	TS	Hits@1	Hits@5	BLEU	TER	BertScore
3	R	0.7484	0.9934	11.1763	1.0130	0.8545
	T	0.7921	0.9947	11.0988	1.0166	0.8542
5	R	0.7281	0.9881	11.3060	1.0258	0.8545
	T	0.7903	0.9926	10.8267	1.0344	0.8538
8	R	0.7347	0.9918	10.9970	1.0183	0.8417
	T	0.7805	0.9929	10.7776	1.0046	0.8508
10	R	0.7246	0.9926	11.3495	1.0482	0.8533
	T	0.7715	0.9931	11.3635	1.0392	0.8526
12	R	0.7323	0.9908	10.8009	1.0162	0.8530
	T	0.7742	0.9929	10.9758	1.0436	0.8548

Table A.6: Xbox seed variation results (random test)

Seed	TS	Hits@1 TopRank
3	R	0.7499
	T	0.7937
5	R	0.7296
	T	0.7924
8	R	0.7370
	T	0.7821
10	R	0.7262
	T	0.7750
12	R	0.7325
	T	0.775

Table A.7: Xbox seed variation results (tailored test)