TÉCNICO LISBOA

Goal-Oriented Dialogue with Sparse Language Models

Rita Fernandes Leite dos Santos Costa

Thesis to obtain the Master of Science Degree in

Aerospace Engineering

Supervisors: Prof. André Filipe Torres Martins
Prof. Maria Luísa Torres Ribeiro Marques da Silva Coheur

Examination Committee

Chairperson: Prof. José Fernando Alves da Silva
Supervisor: Prof. André Filipe Torres Martins
Member of the Committee: Prof. Alberto Abad Gareta

January 2021

In memory of Jorge.

Declaration

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

Acknowledgments

First and foremost, I would like to express my deepest appreciation to my supervisors, Prof. André Martins and Prof. Luísa Coheur, for all the support, guidance and inspiration, which were crucial for the development of this project.

I wish to thank the Unbabel research team for the prompt availability and shared knowledge. I am particularly grateful for the assistance given by Ricardo Rei, whose involvement and advice were very important to complete this work.

A word of thanks to all the annotators for the essential contribution to this study.

I would also like to acknowledge those who have accompanied me along this path. A special thanks to my close university friends, for teaching, inspiring me and growing with me throughout these years. To BEST Lisbon, for the unique opportunities and developed bonds. To my India peers, with whom I shared the most overwhelming experience of my life. To those who lived with me, for being a family away from home. And to my friends from Braga, for always being by my side.

Finally, and most importantly, I would like to express my deepest gratitude to my family, and particularly to my parents, for all the lessons taught, doors opened and unconditional love and support.

Resumo

Sistemas de diálogo orientados a um objetivo têm o propósito de fornecer uma resposta automática em conversas com uma finalidade específica. Dados recentes avanços em arquiteturas de Aprendizagem Profunda, abordagens mais flexíveis têm aparecido, com a possibilidade de aplicar conhecimento pré-existente de modelos treinados de uma forma auto-supervisionada a sistemas de diálogo. Contudo, a necessidade de adaptar a resposta original a cada contexto torna a tarefa de a gerar particularmente desafiante. Diferentes estratégias para formar uma resposta têm sido propostas, com o intuito de tornar o texto gerado mais fluente, coerente, e relevante. O objetivo deste estudo consiste em experimentar a utilização de técnicas de geração esparsas neste contexto, recorrendo à amostragem da transformação α-entmax. Esta técnica será comparada com outras abordagens do estado da arte, como *busca gananciosa* e *amostragem de núcleo*, avaliando com detalhe os diferentes sistemas originados.

Por outro lado, à medida que as abordagens modulares são substituídas por arquiteturas *ponta-a-ponta*, torna-se mais difícil avaliar estes sistemas de diálogo. Vários trabalhos recorrem a métodos de avaliação caraterísticos de outras tarefas, nomeadamente tradução automática, levantando dúvidas sobre a sua relevância para avaliar diálogo. Para as esclarecermos, conduzimos uma recolha de anotações humanas sobre o desempenho de vários sistemas, com o objetivo de determinar a correlação entre estas métricas automáticas e a perceção humana de qualidade. O método de avaliação é uma parte importante da análise de desempenho, já que uma escolha inapropriada deste poderá levar a conclusões erradas.

Palavras-chave: sistemas de diálogo orientados a objetivo, amostragem α-entmax, métricas automáticas, avaliação humana

Abstract

The purpose of goal-oriented dialogue systems is to provide automatic responses in a conversation with a specific goal. Given recent advances in Deep Learning, this task is now more flexible, as pre-existing knowledge from models trained with self-supervised learning can be transferred to conversation systems. However, the need to adapt the original answer to the dialogue context makes the task of generating it particularly challenging. Different strategies to decode a sentence have been proposed, aiming at making the generated text more fluent, coherent, and relevant. The goal of this study is to experiment sparse generation techniques in this framework, which sample from the recently proposed α-*entmax* transformation. We compare this technique with other state-of-the-art approaches, such as *greedy search* and *nucleus sampling*, by thoroughly assessing the different systems.

Moreover, as the modularized approach is replaced by *end-to-end* architectures, goal-oriented systems become more difficult to be evaluated. Many works resort to evaluation methods imported from other tasks, namely machine translation, raising the question of whether they are suitable for evaluating dialogue. To address this problem, we conduct a study to determine the correlation between these automatic metrics and human perception of quality. The evaluation procedure is an important part of the performance analysis, since choosing an inappropriate method can lead to the wrong conclusions.

Keywords: goal-oriented dialogue systems, α-entmax sampling, automatic metrics, human evaluation

Contents

List of Tables

List of Figures

List of Acronyms

AI Artificial Intelligence.

BERT Bidirectional Encoder Representations from Transformers.

BLEU Bilingual Evaluation Understudy.

BPE Byte-Pair Encoding.

DL Deep Learning.

DST Dialogue State Tracker.

FFNN Feed-Forward Neural Network.

GPT Generative Pre-trained Transformer.

GPT-2 Generative Pre-trained Transformer 2.

IAA Inter Annotator Agreement.

LSTM Long Short-Term Memory.

METEOR Metric for Evaluation of Translation with Explicit Ordering.

ML Machine Learning.

MLP Multi-Layer Perceptron.

MultiWOZ Multi-Domain Wizard-of-Oz.

NLG Natural Language Generation.

NLP Natural Language Processing.

NLU Natural Language Understanding.

NN Neural Network.

RNN Recurrent Neural Network.

Seq2Seq Sequence-to-Sequence.

Chapter 1

Introduction

The evolution of technology has allowed the automation of several processes across diversified engineering industry fields. Namely, customer support services have drastically evolved with the relative recent advances in Machine Learning (ML). One of the biggest goals of Natural Language Processing (NLP) is to develop a conversational Artificial Intelligence (AI) system which can interact with humans in goal-oriented dialogue tasks.

The study of goal-oriented dialogue systems can be particularly relevant to Aerospace Engineering in two main frontiers. The most self-evident is the direct application of these systems in the aviation business, where conversational AI leads to an improvement of customer support platforms, consequently enhancing the client experience.[1] Besides, recent studies support the introduction of speech dialogue systems in manufacturing processes. An efficient human-robot communication can reform the aerospace industry, not only enhancing their alone performance through cooperation between the two parts, but also supporting decision making with information-rich systems (Gaizauskas et al., 2018).

1.1 Motivation

Goal-oriented dialogue systems are developed to serve a purpose in a conversation. Historically, these systems were a complex and highly hand-crafted pipeline, consisting of different and individually optimized components (Figure 1.1).

The evolution of Deep Learning (DL) and consequent advances in language modelling allow for the development of neural approaches in answer generation, which typically rely on Sequence-to-Sequence architectures (Sutskever et al., 2014; Wen et al., 2015). The traditional modular approach is replaced by the joint optimization of multiple components, originating less complex systems with a stiffer architecture, learnable in an end-to-end manner (Wen et al., 2017). Rather than a rule-based translation of system's instructions into natural language utterances, finding the most adequate answer in a conversation is now a more flexible process, which results in better answers, but also opens the door to less predictable system behaviours.

[1] As an example, we have the work done by the company Unbabel: `https://unbabel.com/customer-service/travel/`, last accessed on 14-12-2020.

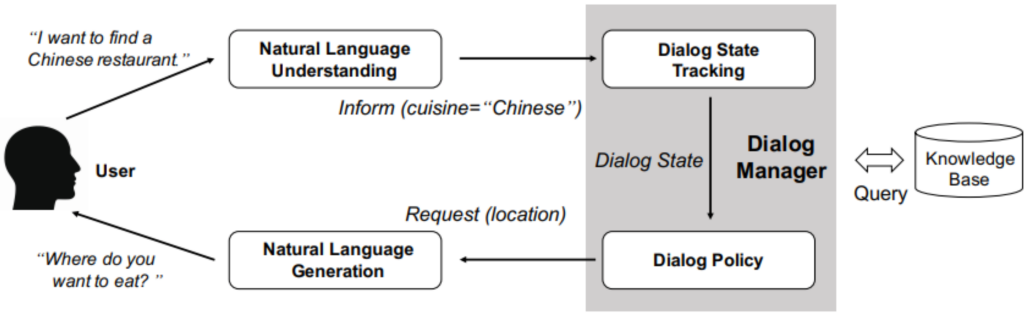


Figure 1.1: The composition of a pipeline system. Image source: Zhang et al. (2020b)

Research on answer generation can be categorized in two different approaches: retrieval and generative based methods. The latter are more flexible and better aligned with the goal of providing a response that is the closest to human behaviour, in the sense that they allow to adapt to a new context and generate an original possible answer, rather than selecting from a finite set of existing possibilities. These methods are naturally more complex, as systems are required to learn grammar, syntax, dialogue management and language generation at the same time (Celikyilmaz et al., 2020). Recent advances in large-scaled pre-trained Language Models, such as Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2019) and Generative Pre-trained Transformer 2 (GPT-2) (Radford et al., 2019), have paved the way for higher quality generation, as pre-existing language knowledge can be fine-tuned for a specific task. Namely, Wolf et al. (2019) and Golovanov et al. (2019) have shown the applicability of these models to conversational systems, with Budzianowski and Vulić (2019) introducing them to the goal-oriented framework.

Nonetheless, mimicking the human way of constructing a sentence is a challenging task and the chosen method can have great influence on the final result. The fact that the most suitable sentence is not always the combination of the highest probability words has led to investigating options besides greedy search, namely top-k sampling (Fan et al., 2018) and nucleus sampling (Holtzman et al., 2020). This thesis adds to the study of end-to-end generative conversation systems, by leveraging α-entmax transformations to decoding a sentence in a Language Model (Peters et al., 2019; Martins et al., 2020), in the goal-oriented setting. We compare this approach with state-of-the-art generation techniques and evaluate how different strategies influence the quality of the generated answer, using both automatic and human metrics.

Furthermore, as important as developing a system that resembles a human is the ability to correctly evaluate its performance. End-to-end goal-oriented dialogue generation demands a change in this paradigm, as the traditional rule-based methods measuring how well the system fulfills the required slots are less appropriate (Deriu et al., 2020). Given the scarcity of automatic evaluation methods for dialogue systems, many works have started reporting metrics created for different purposes, namely machine translation. By measuring the similarity between a generated sentence and its gold reference, BLEU (Papineni et al., 2002), METEOR (Banerjee and Lavie, 2005) and BERTScore (Zhang et al., 2020a) are applicable to different NLP tasks. Along with the increased usage of these metrics comes the inquiry of whether they are accurate indicators of quality in dialogue systems. To answer this ques-

tion, we will exploit the most reliable tool to judge if a system is faithfully imitating the behaviour of a human — human judgement itself.

1.2 Objectives

The objectives of this work lie in the study of several aspects of end-to-end task oriented dialogue generation, whose development has opened up new challenges.

First and foremost, given the reported good performance of α-entmax sampling in the open-domain setting, we propose to introduce this technique to the goal-oriented framework, aiming at decreasing the stochasticity of the generation process, while maintaining the quality. To do so, we recreate the architecture of a state-of-the-art dialogue generation end-to-end system and compare our approach with the originally documented results. Besides, we intend to analyze some points which were not explored in the pioneer work, such as the importance of the format of the context in the developed system.

Alongside with the comparison of these dialogue systems, comes the question of how representative are the adopted metrics of the human perception of quality. To answer it, we propose to collect human annotations for the several developed dialogue systems, with the goal of defining a correlation with the scores for those metrics. However, in goal-oriented dialogue generation, most existing annotation approaches resort on an interactive evaluation, rating the systems' performance at the end of the conversation or measuring its length, which is supposed to be a good indicator of the system's efficiency. This strategy can not be applied to our systems, as they are not able to perform in an interactive way — they appear from the idea of AI helping humans to successfully complete their tasks, in a more efficient manner, rather than replacing them. Therefore, we need to collect annotations at the turn-level. As these criteria is poorly defined in similar works, we develop our own annotation strategy, which will allow us to draw the intended conclusions.

1.3 Main Contributions

To achieve the proposed objectives, the main contributions of this work lie in the following:

1. Experimenting alternative sparse transformations in goal-oriented dialogue, using both greedy and sampling methods. We recreate a previously developed Neural Dialogue Language Model, adapting it to our goals.

2. A rigorous comparison between different state-of-the-art decoding methods for dialogue generation, using several automatic evaluation metrics. These are complemented with a considerable amount of examples, allowing to understand some positive and negative aspects of each technique. We observe that response generation in the goal-oriented setting benefits more from deterministic techniques, which can be justified by the nature of these systems, whose restriction to certain domains and topics results in a lower spectrum of possible answers to a given context.

3. An evaluation of other aspects of the proposed architecture, such as the context importance in end-to-end conversational Natural Language Generation. We compare systems with different types of context, with both automatic metrics and practical examples, confirming the usefulness of having key information in the context. However, the mismatch between some metrics results and observed examples also motivate the investigation of the used metrics significance.

4. An original suggested set of evaluation dimensions which are considered relevant to properly evaluate the dialogue generation problem, in the goal-oriented framework.

5. An empiric human evaluation with several annotators, to collect impressions regarding the developed systems, followed by an analysis of the possible correlation between automatic metrics and human perception of quality. We conclude on the adequateness of these metrics for the task, which can be used in system comparison, but are not able to grasp certain nuances of response generation.

1.4 Thesis Outline

In Chapter 2, some ML background is reviewed, serving as foundation for this work development. We then connect it to NLP applications, followed by an overview of answer generation in goal-oriented dialogue. Finally, an introduction to sparse transformations is provided, with particular focus on α-entmax.

In Chapter 3, we start by introducing state-of-the-art Language Models for dialogue, followed by some corpora details. The adopted Neural Dialogue Language Model is then presented, with a thorough description of the systems' implementation. We focus on describing the several decoding strategies to be analyzed, as well as the chosen automatic evaluation metrics.

Chapter 4 consists of a presentation of the experimental results, with a detailed comparison of the different systems' performance. Besides the metrics' scores, several practical examples are displayed, motivating further analysis of the chosen evaluation methods.

In Chapter 5, we conduct a human evaluation experience, concluding on the correlation between the automatic metrics and human judgements, regarding several aspects of dialogue construction.

Finally, Chapter 6 holds the main findings of this study and proposes some directions for future work.

Chapter 2

Background

In this chapter, the necessary theoretical concepts to understand this work are presented. In Section 2.1, an overview of the most relevant Deep Learning architectures is provided, from the Multi-Layer Perceptron (MLP) to the Transformer, which will be used in this work. Following this, α-entmax transformation is presented in Section 2.2, linked to some of the main innovations in this study. Section 2.3 covers how text is represented and explains how to apply the presented architectures to the NLP field. In Section 2.4, the concept of Transfer Learning is introduced, as it will be used for this work model's implementation (Chapter 3). Finally, a review of dialogue systems is made in Section 2.5, relating the previous sections to the task of Dialogue Generation, and presenting the problem setting.

2.1 From the Multi-Layer Perceptron to the Transformer

Machine Learning (ML) is the ability of AI Systems to acquire knowledge by extracting patterns from data — training set — which can be used to make predictions on unseen data — test set. The most widely-used ML methods are supervised learning methods, which approximate a function $f(x)$, in order to learn to map a variable $x \in \mathcal{X}$ into $y \in \mathcal{Y}$. Depending on how we process x, we can describe several different ML tasks, such as **Classification** problems, where the model assigns an input x to a category identified by the numeric code y; and **Regression** problems, where the model should predict a numerical value given an input, being $\mathcal{Y}$ composed of continuous variables of real values (Goodfellow et al., 2016).

To evaluate the candidate solution, we need to choose an objective function, which we seek to minimize or maximize, depending on the context. When we are minimizing it, we may call it cost function, or **loss function**. In most cases, our model defines a distribution $p(y|x;\theta)$ and we simply use the principle of maximum likelihood, minimizing the negative log-likelihood $-\log p(y|x;\theta)$, equivalently described as the cross-entropy between the training data and the model's predictions, and use this as the loss function. As an example, we have a simple type of prediction algorithms, the Linear Regression Models, which estimate y by defining $f(x)$ as a linear combination of the observed variables:

$$f(x) = w^T x + b, \tag{2.1}$$

where w is a set of weights determining how each input feature x_i affects the prediction, and b is the intercept term, also called bias. To goal is to estimate the unknown parameters $\theta = (w, b)$ in order to minimize the cross-entropy loss.

The evolution of ML paved the way for other linear models, including Naive Bayes Classifiers and Support Vector Machines (SVMs), which are able to solve linearly separable problems. One important innovation lied in extending linear models to represent non-linear functions of x, by applying a linear algorithm to the transformer input $\phi(x)$, where ϕ is a non-linear function of x, commonly referred to as **feature**.

As advances were made in ML fields, non-linear methods were developed, such as Decision Trees, Random Forests, and Neural Networks (NNs). The latter have been showing impressive performance in diverse applications, leading to the development of increasingly more complex architectures. In the next sections, an overview of NN architectures will be performed, starting with the simplest single neuron and culminating in the transformer architectures, which have been scoring remarkable results in many NLP fields, such as Machine Translation, Text Summarization and Question-Answering (Radford and Salimans, 2018; Radford et al., 2019).

2.1.1 Multi-Layer Perceptron

The Multi-Layer Perceptron (MLP), also called Feed-Forward Neural Network (FFNN), is inspired by the behaviour of a biological neuron, specially the human ability of making associations. The goal is to learn y through a new representation of x, that is, modelling:

$$y = f(x; \theta, w) = \phi(x; \theta)^T w. \tag{2.2}$$

This is an example of a NN, with $\phi(x)$ defining a hidden layer. FFNNs are called networks because they are typically composed of a chain of many different functions, or many layers of functions, to capture the non-linear patterns in the data. The learning algorithm must decide how to use these layers to produce the desired output. The length of the chain gives the depth of the model, hence the name **Deep Learning**. Each layer of a FFNN is a function of the one preceeding it, as shown bellow, where the first layer is given by Equation 2.3 and the second one is given by Equation 2.4.

$$h^{(1)} = g^{(1)}(w^{(1)T}x + b^{(1)}), \tag{2.3}$$

$$h^{(2)} = g^{(2)}(w^{(2)T}h^{(1)} + b^{(2)}). \tag{2.4}$$

Assuming $L \geq 1$ layers, FFNNs can, in theory, approximate any function we want, by learning the optimal parameters $\theta = \{(w^{(l)}, b^{(l)})\}_{l=1}^{L+1}$ from a set of training data (Hornik et al., 1989).

The total cost function used to train a NN often combines the loss function $L(f(x_i; \theta), y_i)$ with a regularizer term $\lambda\Omega(\theta)$, which refers to the way the weights are penalized to avoid over-fitting, where

$\Omega(\theta)$ is a regularizer, for example the l_2 norm $\Omega(\theta) = ||\theta||^2$, and λ is the regularization constant:

$$\mathcal{L}(\theta) = \lambda\Omega(\theta) + \frac{1}{N}\sum_{i=1}^{N} L(f(x_i;\theta), y_i). \tag{2.5}$$

To minimize the loss, we need to calculate its gradient:

$$\nabla_\theta \mathcal{L}(\theta) = \frac{1}{N}\sum_{i=1}^{N} \nabla_\theta \mathcal{L}_i(\theta), \tag{2.6}$$

which is is usually done using the **Gradient Backpropagation Algorithm**, where the derivatives are calculated according to the chain rule. The values of θ are randomly initialized and then updated using the **Stochastic Gradient Descent** algorithm, slowly adapting the weights of the links to minimize the loss function, by sampling a single training example:

$$\theta \leftarrow \theta - \eta\nabla_\theta \mathcal{L}_j(\theta). \tag{2.7}$$

To make the process more efficient, it is common to use the **Mini-Batch Gradient Descent** algorithm, which estimates the gradient in mini-batches $j_1, ..., j_B$, with $B << N$, only updating the weights when all the examples in the mini-batch have been seen:

$$\theta \leftarrow \theta - \eta\frac{1}{B}\sum_{i=1}^{B} \nabla_\theta \mathcal{L}_{j_i}(\theta). \tag{2.8}$$

One crucial part of projecting a FFNN is the choice for loss function, directly related to the output unit of the network, which transforms the hidden features in the final output. The type of output unit will depend on the nature of the problem: in a regression problem, the output layer is a node with a linear activation unit; in a classification problem, the output unit is usually the sigmoid activation, for binary classification, or the softmax activation, for multi-class classification (Goodfellow et al., 2016).

2.1.2 Sequence-to-Sequence Learning

FFNNs have accomplished outstanding performance in multiple learning tasks with large labelled datasets. However, despite their power and flexibility, FFNNs can only be applied to problems where both inputs and outputs have a fixed size. This is a critical limitation, since many problems require an unknown a priori length to represent the sequences, specially in NLP applications.

To represent an arbitrary long sequence of inputs, a more suitable architecture is the Recurrent Neural Network (RNN) (Elman, 1990), which is applicable to sequence prediction problems. In RNNs, the outputs are also influenced by hidden states representing context based on prior inputs, allowing to share knowledge between networks. One of the possible usage of RNNs is **Language Modelling**. Defining a sentence as a sequence of words $w = (w_1, ..., w_T)$, a Language Model (Bengio et al., 2003) is able to look at a part of a sentence and predict the next word, calculating the probability of a word

given the previous ones:

$$p_\theta(w) = \prod_{t=1}^{T} p_\theta(w_t|w_1, ..., w_{t-1}). \tag{2.9}$$

Given a set S of training sentences, the strategy to learn the language modelling parameters θ is to minimize the cross-entropy, or the negative log-likelihood:

$$\mathcal{L}(\theta) = \sum_{i=1}^{|S|} \sum_{t=1}^{T} \log p_\theta(w_t|w_{<t}), \tag{2.10}$$

which is equivalent to minimize the perplexity $2^{\mathcal{L}(\theta)}$. Training RNNs is done by backpropagation through time, which means that long-term information has to travel through all cells before getting to the present processing cell. This information can be easily corrupted if multiplied many times for small values — the vanishing gradient problem (Pascanu et al., 2013) — meaning, in practice, that long-term dependencies are difficult to learn. Some innovations appeared to solve this problem, such as the Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997), where memory cells are used to decide what to keep and what to erase from memory, and the Gated Recurrent Unit (Cho et al., 2014), which creates shortcut connections between nodes. These architectures allowed for improvements in different NLP fields, such as Speech Recognition (Graves et al., 2013) and Machine Translation (Sutskever et al., 2014).

Statistical Machine Translation evolved to Neural Machine Translation, with the popularity of Sequence-to-Sequence (Seq2Seq) Models, which typically have an Encoder-Decoder architecture. The Encoder RNN processes the input information $x = (x_1, ..., x_T)$ into a context vector $c = q(h_1, ...h_T)$, where the hidden state at time t is $h_t = f(x_t, h_{t-1})$, with f and q being some nonlinear functions. The Decoder is trained to predict the word w_t, given not only the previously generated words $w_1, ..., w_{t-1}$ (which is the case of Equation 2.9), but also a context c:

$$p_\theta(w) = \prod_{t=1}^{T} p_\theta(w_t|\{w_1, ..., w_{t-1}\}, c). \tag{2.11}$$

This is the concept of **Conditional Language Modelling**, which can be applied to several tasks, depending on the nature of the input x: Machine Translation, if x is a sentence in a foreign language, Summarization, if it is a document, or Dialogue Systems, with x being a conversation history. With a RNN,

$$p_\theta(w_t|\{w_1, ..., w_{t-1}\}, c) = g(w_{t-1}, s_t, c), \tag{2.12}$$

where g is a non-linear function, giving a probability conditioned on the hidden state s_t of the RNN.

2.1.3 Attention

The fixed-size vector state is a bottleneck in the variant of Seq2Seq models described above, as it is challenging to deal with long sentences. One alternative is to encode the input as a matrix, with each column representing a different input part. An **attention** mechanism is then used, to focus on specific

parts of the input when decoding (Bahdanau et al., 2015). Attention allows to focus on any position in the source sentence, helping with both the bottleneck and vanishing gradients problems, and also providing some interpretability to the models. The probability is now conditioned on a different context vector c_i for each target word y_i, instead of the original single c. Each context vector c_i is computed as a weighted sum of the annotations $(h_1, ..., h_T)$ — the encoder hidden states:

$$c_i = \sum_{j=1}^{T} \alpha_{ij} h_j, \tag{2.13}$$

where α_{ij} gives the importance of the annotation h_j related to the previous hidden state s_{i-1} in deciding the next state s_i and generating the next word w_i. Each attention score α_{ij} is traditionally calculated using the **softmax function**:

$$\alpha_{ij} = \text{softmax}(e) = \frac{\exp(e_{ij})}{\sum_{k=1}^{T} \exp(e_{ik})}, \tag{2.14}$$

where the alignment model $e_{ij} = a(s_{i-1}, h_j)$ quantifies how well the inputs around position j and the output at position i match.

In some applications, one drawback of the softmax distribution is that the computed weights are scattered along the whole sequence. To be able to have a more selective attention focus, Martins and Astudillo (2016) introduced the **sparsemax attention**, which is able to return sparse distributions, assigning zero probability to some of the output variables.

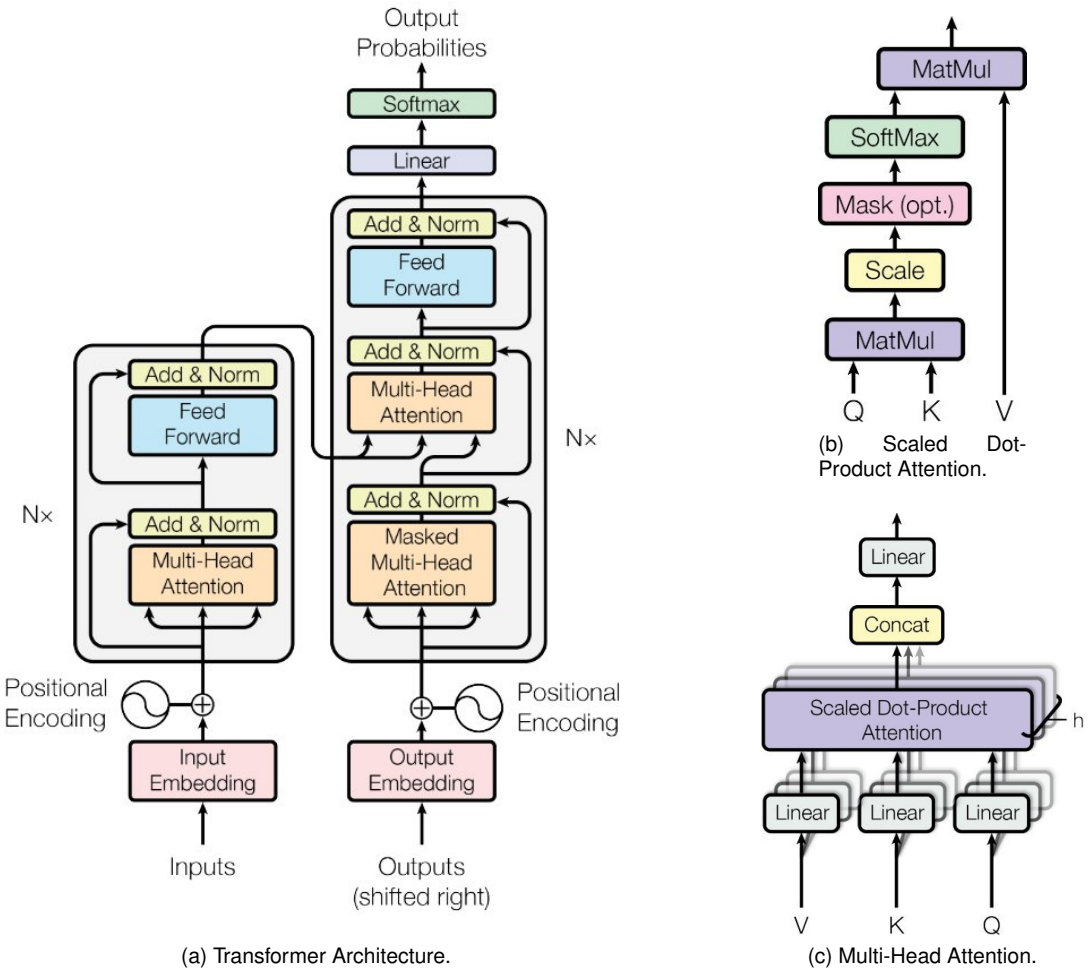
2.1.4 Transformer

Attention allows to focus on arbitrary input positions, shortcutting the computation graphs and leading to an encoder-decoder architecture proposition which does not need RNNs — the **Transformer** (Vaswani et al., 2017). It instead relies exclusively on **self-attention** mechanisms to build dependencies between inputs and outputs.

The transformer architecture uses Scaled Dot-Product Atention to calculate self-attention. Firstly, we create 3 vectors per input vector: **query** and **key**, both of dimension d_k and **value**, of dimension d_v. The output is computed as a weighted sum of the values, where the weight assigned to each value is the **self-attention score**, determining how much focus should be given to each part of the sentence as we encode a certain word. The score is then normalized, being divided by $\sqrt{d_k}$ and applyed a softmax function. This process is schematized in Figure 2.1b. In practice, attention is calculated on a set of queries simultaneously, packed in a matrix Q, with keys and values also packed in matrices K and V, respectively:

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

To increase the model's performance, the authors introduce multiple (h) attention heads, instead of performing a single attention function. **Multi-Head Attention** allows the model to attend to information from different representation subspaces at different positions, as the input embeddings are projected in h sets of query/key/value (Figure 2.1c).



(a) Transformer Architecture.

(c) Multi-Head Attention.

Figure 2.1: Transformer Schemes. Image source: Vaswani et al. (2017)

The transformer uses learned embeddings (Section 2.3) to convert the input tokens and output tokens into vectors of dimension d_{model}. To introduce knowledge about the position of the tokens in the input sequence, **positional encodings** are added to the input embeddings.

Multi-Head Attention is used both in the encoder and the decoder (left and right parts of Figure 2.1a, respectively). The encoder is composed by a stack of N=6 layers, each of them consisting of 2 sublayers: a multi-head attention mechanism followed by a FFNN. The decoder is also composed by a stack of N=6 layers, with 3 sublayers each: the first is a multi-head attention layer, where future positions are masked, to only attend at previous embeddings on the output sequence; the second sublayer is a multi-head attention mechanism over the output of the encoder, transformed in a set of attention vectors K and V; the last sublayer is a FFNN, similar to the encoder's.

Finally, the output vector of the decoder is fed into a linear layer, a NN which projects the output into a score vector over the words in the vocabulary. The output of the NN is a softmax layer, which turns these scores into probabilities, all positive and added up to 1. The next word to be chosen will then depend on the decoding strategy (Section 3.4).

2.2 α-entmax Transformation

The final step of the decoding consists of taking the scores produced in the last linear layer into a softmax function (Equation 2.15), to turn them into a probability distribution. Softmax is a dense distribution, which means that a mass probability is always assigned to all the words, even if it is very small. When sampling directly from it, the system can generate unnatural text, due to the unreliability of the tail of this distribution. Some techniques arose from this problem, such as nucleus and top-k sampling, which will be presented in Section 3.4.

However, these sampling techniques are only applied at decoding time, while at training time they are still optimized with the original softmax distribution, naturally creating a mismatch between the training and testing process. In this work, experiments will be performed sampling from the recently proposed α-entmax transformation (Peters et al., 2019), which automatically produces sparse probability distributions, avoiding the mismatch created between training and testing. The α-entmax transformation is defined as:

$$\alpha\text{-entmax}(z_t) = \underset{p \in \Delta^d}{\operatorname{argmax}}\ p^T z_t + H_\alpha(p), \tag{2.16}$$

where z_t are the scores produced by the model, $\Delta^d = \{p \in \mathbb{R}^d | \sum_{i=1}^d p_i = 1, p \geq 0\}$ is the probability simplex, and H_α is the Tsallis α-entropy:

$$H_\alpha = \begin{cases} \frac{1}{\alpha(\alpha-1)} \sum_j (p_j - p_j^\alpha) & \alpha \neq 1 \\ -\sum_j p_j \log p_j & \alpha = 1 \end{cases} \tag{2.17}$$

The negative log-likelihood loss in Equation 2.10 is replaced by:

$$\mathcal{L}(\theta) = \sum_{i=1}^{|S|} \sum_{t=1}^{T} l_\alpha(z_t(\theta, w_{<t}), w_t), \tag{2.18}$$

where $l_\alpha(z_t, w_t)$ is the proposed α-entmax loss:

$$l_\alpha(z_t, w_t) = (p_\theta - e_w)^T w_t + H_\alpha(p_\theta), \tag{2.19}$$

with $p_\theta = \alpha\text{-entmax}(z_t)$ and e_w a one-hot vector representing the ground truth word w.

In Figure 2.2, the representation of α-entmax in two dimensions is presented, for multiple values of α. As it is possible to observe, all mappings besides softmax saturate at $t = \pm 1/\alpha - 1$. By replacing the softmax function in the language modelling task, the probability distribution will be shaped in a different way, which can have beneficial impact in the way the text is generated, similar to Martins et al. (2020). [1]

[1] The implementation uses the entmax repository: `https://github.com/deep-spin/entmax`, last accessed on 29-06-2020.

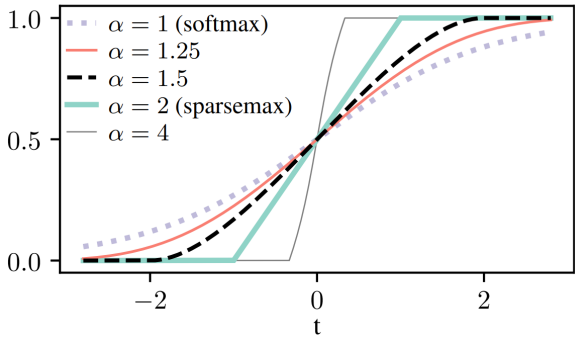


Figure 2.2: Illustration of α-entmax in 2 dimensions. Image source: Peters et al. (2019)

2.3 Text Representation

In Section 2.1, it was explained how DL algorithms grasp knowledge from data, which can be used to make predictions. In order to apply these mechanisms to NLP problems, it is necessary to represent textual information in a suitable way.

2.3.1 Sparse Representations

In sparse representations, a portion of text corresponds to a vector (or embedding) $v \in \mathbb{R}^{|\mathcal{V}|}$, with each entry representing a word belonging to a fixed-size vocabulary $\mathcal{V}$. The most basic approach to represent text through a **one-hot vector**, with value 1 assigned to a word that is present, and 0 corresponding to not present. One drawback of this method is the impossibility to represent similarity between words, ending up not being the ideal method for text representation.

2.3.2 Dense Representations

Contrasting to sparse representations, dense representations provide a better generalization and require much fewer parameters to represent words in the embedding space (Jurafsky and Martin, 2019).

Word embeddings are a continuous representation of words in a low dimensional vector space, where similar words take similar parts of the modelling space. These representations are therefore capable of capturing the semantic meanings of words. The goal is to learn a global word embedding matrix E $\in \mathbb{R}^{|\mathcal{V}| \times d}$, with vocabulary size $|\mathcal{V}|$ and number of dimensions d, where each word is represented by a dense vector composed of values for these dimensions. In Section 2.1.2, Language Models were presented as models with the task of predicting words given a certain context. They are also used to learn internal representations, that can be transferred across tasks. Well-known Language Models for learning word embeddings include Word2Vec (Mikolov et al., 2013), which considers a context window around each word in the sentence, and GloVe (Pennington et al., 2014), where the training is performed on aggregated global word–word co-occurance statistics of a corpus.

Although word embeddings constitute a useful development in text representation, this technique represents a word regardless of the context it is inserted in. Some words can have different meanings,

such as *"book"*, which can be either use as an object or a verb, depending on the context. When a word has a fixed representation, some of its meanings are lost, damaging our NLP system. **Contextual embeddings** move beyond word-level semantics, in that each word has a representation which is a function of the entire input sequence, being able to capture syntax and semantics. Peters et al. (2018) presented Embeddings from Language Models (ELMo) that concatenates representations from the forward and backward LSTMs without considering the interactions between the left and right contexts. Transformer models use attention to learn these embeddings, such as GPT (Radford and Salimans, 2018) and GPT-2 (Radford et al., 2019), which use a left-to-right decoder, where every token can only attend to its left context (Section 3.1.2). For tasks where it is important to incorporate context from both directions, Devlin et al. (2019) proposed Bidirectional Encoder Representations from Transformers (BERT), a masked language model where some tokens of the input sequence are randomly masked, applying a transformer encoder to attend to bi-directional contexts.

2.4 Transfer Learning

So far, multiple architectures supported by the performance of NNs have been presented, many of which performed great achievements on NLP applications for the past years. However, despite their success, NNs do not generalize for data they have not seen during training, only succeeding under the assumption that the training and testing data are taken from the same feature space and distribution. This constitutes a problem, since Supervised Learning demands the labelling of sufficient amount of examples to every new possible setting, which is unfeasible in real life applications (Ruder, 2019).

Besides, the concept of learning from a blank state is antithetical to the way humans actually learn. The idea of **Transfer Learning** comes from people's ability to apply previously learned knowledge to solve new problems, with either faster or better solutions: for example, in the task of learning how to play the piano, a person who knows how to play the electric organ will be easily more successful than a person who does not possess this knowledge. Research on Transfer Learning has been intensely conducted for the past years, with the goal of allowing to share knowledge between different tasks, domains and languages (Pan and Yang, 2010).

When introducing GPT, Radford and Salimans (2018) showed that NLP tasks such as Question Answering and Document Classification can be improved by unsupervised pre-training of a model on a corpus of unlabelled text, followed by supervised fine-tuning. By pre-training this standard Language Model, it acquires linguistic knowledge useful for several tasks, having been proved to succeed in some zero-shot applications — that is, being directed applied to tasks or corpora it was not trained on. However, the fine-tuning phase is particularly relevant if the model should acquire knowledge regarding a task or language specificities, to be shown in the following sections. Following GPT, many others transformer based models were developed, such as BERT (Devlin et al., 2019), GPT-2 (Radford et al., 2019) and TransformerXL (Dai et al., 2019).

2.5 Conversational AI Systems

After having an overview of the relevant practices in NLP, the problem setting can now be presented. A dialogue system is a computer system whose function is to interact with a human in natural language. Conversational AI systems can be classified according to their application in two different paradigms (Chen et al., 2017):

- **Open-Domain Systems**, most commonly referred to as chitchat bots, are mostly used for companion, not being specialized in any specific domain and with the major goal of maximizing the user engagement. Sometimes, these systems can also be used for therapeutic purposes, such as ELIZA (Weizenbaum, 1966), the first chatbot ever created, designed to simulate a psychologist.
- **Goal-Oriented Systems**, which are aimed to assist the user in a specific task completion, belonging to one or multiple domains, such as flight booking, restaurant information query or technical support. Examples of goal-oriented dialogue systems can be Apple Siri and Amazon Alexa, which work as personal assistants, or any chat bot from a support application.

Depending on the type of dialogue system, the model will need different types of architectures and settings: for instance, while an open-domain system can be trained on a large amount of unlabeled corpora, a goal-oriented system requires more precise language understanding, with more in-domain data.

2.5.1 Goal-Oriented Systems

The existing studies on goal-oriented systems classify them in two: pipeline architecture and end-to-end methods. Before getting into details about these architectures, it is crucial to define some concepts related to domain specific vocabulary, in a task oriented setting (Jurafsky and Martin, 2019; Schatzmann and Young, 2009; Gao et al., 2018):

- The **dialogue act** represents the interactive function of the turn or sentence, such as *greet*, *request* or *confirm*. Some dialogue acts may have slots or slot-value pairs as arguments, such as *request(num_tickets)* for the utterance *"How many tickets do you need?"*. Different dialogue systems may require the labeling of different kinds of acts, and dialogue acts are generally domain specific;
- The **ontology** is a structure representing the kinds of intentions the system can extract from user sentences. It is composed of the collection of attributes of a specific domain, called slots, and a set of possible values for each slot. For example, in the *flight booking* domain, the *destination* of the flight can be a slot with *Lisbon* being a possible value for this slot;
- The **dialogue state** contains all information about what the user is looking for at the current turn of the conversation. It includes the most recent dialogue act, but also the resume of the dialogue history;

- The **dialogue policy**, whose goal is to decide what action the system should take next, that is, what dialogue act to generate.

Pipeline Architecture

In a pipeline (or modular) architecture, the system is traditionally composed of 4 main modules:

- **Natural Language Understanding (NLU)** — converts the words in the utterance to a meaningful representation, by firstly classifying the domain, then determining the user's intent and finally extracting the particular values for each slot;
- **Dialogue State Tracker (DST)** — updates the current dialogue state, based on the possibilities defined by the domain ontology;
- **Dialog Policy Module** — given the dialogue state, decides the optimal system action, in order to lead the dialogue towards its goal. This module can be ruled based (decides based on a rule), supervised (learns to resemble the answers from a corpus) or based on reinforcement learning (learns in the dialogue environment). Before deciding what will be the system action in the Dialog Policy Module, the system may interact with a Knowledge Base, depending on the applications;
- **Natural Language Generation (NLG)** — converts the system action into an actual system utterance.

Systems with a pipeline architecture were initially developed based on *slot-filling*, narrowing the structure of a dialogue state to a predefined set of slots to be filled during the dialogue (Lemon et al., 2006; Young et al., 2013; Wang and Lemon, 2013). Although reliable, this rule-based technique is limited to particular domains and hard to be generalized to new scenarios. These limitations motivated the study of *corpus-based* approaches, with minimal dependence on rules (Angeli et al., 2010; Kondadadi et al., 2013). By learning directly from data, the systems are able to mimic human responses in a more natural way and become easily extendable to other domains.

End-to-End Methods

The modules of a pipeline system are optimized separately, which does not necessarily lead to an optimal performance (Gao et al., 2018). The development of Seq2Seq Learning (Section 2.1.2) proposed using NNs to map sequences into sequences, not only leading to advances in machine translation, but also motivating end-to-end approaches in the development of dialogue systems, initially in open-domain (Vinyals and Le, 2015; Serban et al., 2016), but soon exploiting to goal-oriented (Wen et al., 2017). These type of systems use learned neural models to translate the conversation history into the next system response, requiring much fewer hand-craft rules, and therefore replacing the more traditional pipeline architectures.

End-to-end approaches have been also applied to the joint optimization of only some of the four traditional dialogue components. Examples are joint word-level DST models, combining NLU and DST

(Wu et al., 2019), or even systems which include the NLU, the DST and the Dialogue Policy (Zhao and Eskenazi, 2016). In this work, the implemented system incorporates both the Dialog Policy and NLG models, with the goal of generating a response based on the conversation history and dialogue state, using world-level policy to directly generate a response.

2.5.2 Answer Generation

In general, two essential techniques have been developed to generate system utterances in a dialogue: **retrieval-based methods**, which learn to retrieve the most adequate answer from an existing set of options, and **generation methods**, such as Seq2Seq models (mentioned in Section 2.1.2), that generate a sequence of words which corresponds to the system's answer (Chen et al., 2017). On one hand, retrieval models are restricted to the retrieval set, leading sometimes to inaccurate replies, since the coverage is usually scarce for so many domains and tasks. On the other, although generative models have a good performance on simple domains, their application on many domains require large amounts of domain specific annotated data, which is usually not available for real world scenarios (Gao et al., 2018; Peng et al., 2020).

A lot of work has been done in the past years with both retrieval and generation techniques, for either open-domain or task-oriented applications, showing the advantages and drawbacks of each approach. Typically, the focus in goal-oriented applications has been in retrieval models (Kannan et al., 2016; Henderson et al., 2019), since it allows to have full control over the systems' responses, and mainly due to the data scarcity problem: besides the ability of constructing coherent phrases, these systems require the knowledge of specific domains and their ontology. This multi-domain annotated corpora is usually scarce, because the collection and annotation of data for these applications is an expensive, complex and time-consuming procedure. In this setting, the possibility to transfer knowledge is particularly useful, as many NLP tasks share common knowledge about language.

Wolf et al. (2019) and Golovanov et al. (2019) have recently shown that fine-tuning generative language models for conversational applications can have state-of-the-art results, in the domain of personal conversations. Budzianowski and Vulić (2019) reproduced this implementation in a goal-oriented setting, documenting a great performance. Following this work, we propose to study how different decoding strategies can influence the final result, focusing our attention in the recently proposed **α-entmax sampling** (Martins et al., 2020), which uses the α-entmax transformation (Peters et al., 2019) to directly sample from a sparse model.

Chapter 3

Neural Dialogue Language Model

Throughout the last chapter, an introduction to the most important DL architectures was made, connecting them to NLP, specifically to the dialogue generation task. In this chapter, we apply these concepts to the implementation of the systems. We start by presenting some relevant architectures (Section 3.1), followed by an overview of the corpora and its preprocessing details (Section 3.2). Our systems' implementation is further explained in Section 3.3, with particular emphasis on the several decoding strategies (Section 3.4) and performance evaluation metrics (Section 3.5), to be subject of analysis in the next chapters.

3.1 Relevant Architectures

As mentioned in Section 2.5.2, the implementation of a dialogue system can take advanteged of pre-trained language models. In this Section, the pionner work using this type of implementation is introduced. Followed by a presentation of GPT-2, it culminates in its original application to the goal-oriented framework.

3.1.1 *TransferTransfo*

TransferTransfo (Wolf et al., 2019) is an open-domain dialogue system which uses a multilayer transformer encoder based on GPT. The model is pre-trained on the BooksCorpus Dataset (Zhu et al., 2015), consisting of more than 7000 books, from a variety of genres. It is followed by fine-tuning on the Persona-Chat Dataset (Zhang et al., 2018), which contains 10907 dialogues between a pair of speakers, all conditioned on a given profile, the *persona*. There is a total of 1155 available *personas*, each of them consisting of at least 5 profile sentences, aiming at increasing the engagingness in chit-chat conversations.

The model is trained on single text input, whose representation is adapted to include information regarding the speaker personality, so it can easily switch from a single speaker to a two speaker setting. This is achieved through adding a set of *dialogue_state embeddings*, in the fine-tuning phase, to the already learned *word embeddings* and *positional embeddings*. These new embeddings serve to identify

whether a certain token belongs to a personality trait, a utterance from *person1* or a utterance from *person2*.

The fine-tuning phase uses **Multi-Task Learning**, defined by jointly training a model in multiple related tasks. In this case, the authors make use of the Language Modelling Task, which obtains next token probabilities over the vocabulary, and introduce the Next-Sentence Classification Task, a classifier whose goal is to determine the most accurate response from a setting of 2–6 possible candidates. To do so, a **Double-Heads Model** (Figure 3.1) is used, where each model's head computes the loss for each task:

- *Language Modelling Loss* — consists of training a common Language Model, where the final hidden state is fed into a softmax layer, obtaining a probability distribution over the vocabulary. In this case, the cross-entropy loss is then applied to the portion of text corresponding to the gold reply, which will be referred to as $Loss_{LM}$ throghout this work.

- *Next-Sentence Prediction Loss* — consists of passing the hidden state of the last token through a linear layer to get a score. The cross-entropy loss is then applied, to classify the correct gold answer among the distractors. It is also referred to as Multiple Choice Loss ($Loss_{MC}$).

The total loss is a weighted sum between the both losses, as evident in Equation 3.1:

$$Loss = LM_{coef} Loss_{LM} + MC_{coef} Loss_{MC}. \tag{3.1}$$

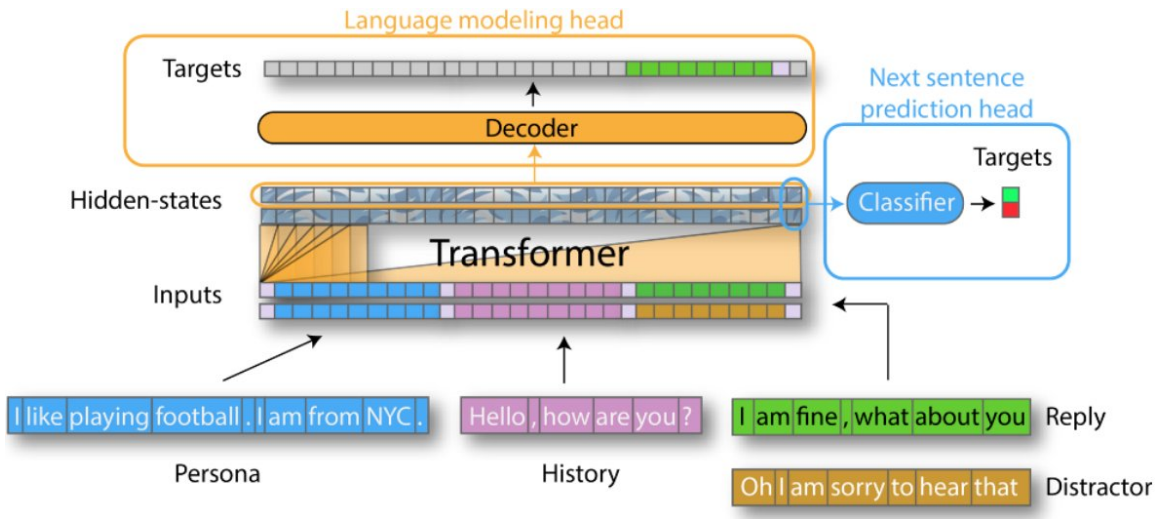


Figure 3.1: Double Heads Model. Image source: How to Build a State-of-the-Art Conversational AI with Transfer Learning, last accessed on 26-12-2020.

3.1.2 GPT-2

The Generative Pre-trained Transformer 2 (GPT-2) is a Language Model trained in a self-supervised manner on the WebText dataset, a massive dataset created with the text present in 45 million internet

links. GPT-2 was demonstrated to have a good performance in several NLP tasks without any task-specific supervision, being trained on a zero-shot setting (Radford et al., 2019). There are four GPT-2 models, whose main difference is on the architecture size.

The tokenization of the language is done using **Byte-Pair Encoding (BPE)** (Sennrich et al., 2016), as a middle ground between character and word level language modelling. This technique breaks up the words into tokens which are longer than characters, but shorter than complete words. The main goal is to break up complex words into simpler ones, making it easier to deal with out-of-vocabulary words. Besides this, to avoid allocating vocabulary slots to many versions of common words (such as "dog", "dog.", and "dog!"), the authors prevent BPE from merging across character categories for any byte sequence, adding an exception for spaces. The vocabulary consists of 50257 tokens.

Like traditional language models, GPT-2 outputs one token at a time, based on the assumption that the probability of a word sequence corresponds to the product of conditional next word distributions, given the context. Once a token is produced, it is added to the sequence of inputs, belonging to the input sequence in the next step, which is called **auto-regression**.

GPT-2 learns contextual embeddings (Section 2.3.2) using self-attention mechanisms (Section 2.1.4). Its architecture is based on the Transformer, being composed of stacked decoder only blocks. Each block has its own weights in both sublayers that constitute it: the Masked Self-Attention sublayer (as seen in the Transformer, each token can only attend to its left context), and the FFNN sublayer. Each head has a different pattern to attend to specific words, explaining why this model captures well many linguistic properties. Similarly to the Transformer's, the output of GPT-2 is a vector with a distribution of probabilities over the vocabulary, to which the next word is chosen depending on the decoding strategy.

3.1.3 *Hello It's GPT-2 — How Are You?*

Builduing on top of Wolf et al. (2019) and Radford et al. (2019), Budzianowski and Vulić (2019) demonstrated the applicability of fine-tuning a pre-trained language model to a goal-oriented dialogue system, addressing the data scarcity problem mentioned in Section 2.5. The combination of Multi-Task Learning with simple text input is validated in the MultiWOZ Dataset (Section 3.2), effectively projecting the Dialog Policy and NLG modules in an end-to-end manner.

Similarly to TransferTransfo (Section 3.1.1), the GPT-2 Language Modelling Head receives three levels of inputs, as represented in Figure 3.2: the *word level* input, the *token level* input and the *position level* input. The *word level* input has information regarding the context, in the form of simple text, simplifying the paradigm of building goal-oriented models, as new information can simply be added to the input. This *word level* input is composed of *belief_state*, *database_state*, *context* and the *system response*, whose composition will be further explained in Section 3.3.1. Similarly to what was explained in Section 3.1.2, the *position level* input simply consists of the information regarding the order of the tokens in the word level input. The *token level* input uses *dialogue_state embeddings*, which identify the tokens in the word level input by informing whether they belong to the user or the system.

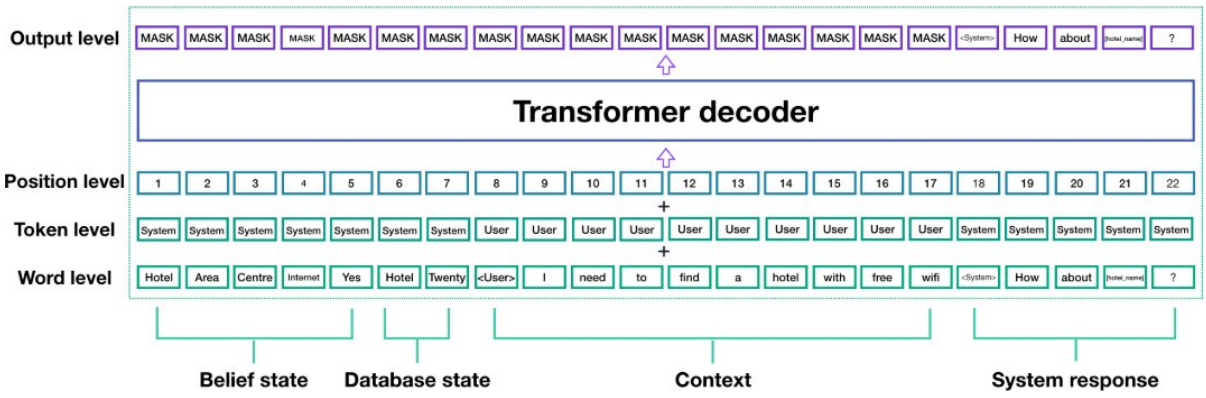


Figure 3.2: GPT-2 input for the fine-tuning phase. Image source: Budzianowski and Vulić (2019)

3.2 MultiWOZ Dataset

The choice of corpora can have a decisive impact in the performance of the dialogue system. For the past years, a lot of work has been performed in order to achieve the perfect dataset, resulting in a variety of available options. In the goal-oriented framework, the Multi-Domain Wizard-of-Oz (MultiWOZ) Dataset (Budzianowski et al., 2018) stands out due to its structured annotations and size, with the impressive amount of 10438 dialogues, being at least one order of magnitude larger than all previous annotated goal-oriented datasets. In terms of size, it was recently surpassed by the Taskmaster-1 (Byrne et al., 2019), which reports 13215 dialogues across 6 domains.

To allow direct comparison with previously developed systems, this study uses MultiWOZ 2.0. However, we are aware that this version contains annotation errors in many dialogues, which can influence the system performance. Improved versions have been recently released, namely MultiWOZ 2.1 (Eric et al., 2019) and MultiWOZ 2.2 (Zang et al., 2020), for which we also present results. Throughout the experiments, we use the original division into *Training*, *Validation* and *Testing Sets*.

3.2.1 Corpora Specifications

The MultiWOZ Dataset is a labelled human-human collection of goal-oriented dialogues, simulating natural conversations between a tourist and an assistant from an information center in a touristic city. The corpus has conversations spanning over 7 domains — *Attraction, Hospital, Police, Hotel, Restaurant, Taxi, Train* — with diverse complexity of tasks, going from a simple information query about an attraction, to booking a night at a hotel, a restaurant reservation and a taxi to connect both places.

The dataset is composed of 10438 dialogues, which can be either single domain or multi-domain. The average number of turns per dialogue is 8.93 and 15.39, for single and multi-domain, respectively. In this section, we present the domains' ontology, followed by the annotations and database structure.

Ontology

As it is common in a goal-oriented system, the domains of MultiWOZ are defined by an ontology, introduced in Section 2.5.1. The slots can be categorized as *informable* and *requestable*. *Informable* slots

correspond to domain information which can narrow down the search for options — for example, price range or area. *Requestable* slots consist of additional details, relevant when a booking is pursued — for example, booking reference or telephone number. The ontology also includes the dialog act type, which summarizes the intention of the utterance. In Figure 3.3, it is possible to see the ontology for all the 7 domains in this dataset, where the upper scripts represent which domains it belogs to: * - universal; 1 - restaurant; 2 - hotel; 3 - attraction; 4 - taxi; 5 - train; 6 - hospital; 7 - police.

act type	inform* / request* / select^{123} / recommend/^{123} / not found^{123} request booking info^{123} / offer booking^{1235} / inform booked^{1235} / decline booking^{1235} welcome* /greet* / bye* / reqmore*
slots	address* / postcode* / phone* / name^{1234} / no of choices^{1235} / area^{123} / pricerange^{123} / type^{123} / internet^{2} / parking^{2} / stars^{2} / open hours^{3} / departure^{45} destination^{45} / leave after^{45} / arrive by^{45} / no of people^{1235} / reference no.^{1235} / trainID^{5} / ticket price^{5} / travel time^{5} / department^{7} / day^{1235} / no of days^{123}

Figure 3.3: Ontology for all domains in MultiWOZ. Image source: Budzianowski et al. (2018)

Annotations

One of the main features of this dataset lies in the richness in annotations. For each dialogue, there is a *goal* entry, which specifies the conversation details. The *goal* is composed of an entry per domain. If the domain was tackled in the conversation, the *informable* slots and values will appear in the *info* field and the *requestable* ones in the *book* field. If there are any restrictions or booking details which are not able to be fulfilled, it will appear in the *fail_info* and *fail_book* fields, respectively.

Besides, the dialogue has a *log* entry, in which the conversation history is presented. Each entry is composed of *text* and *metadata* fields, corresponding to a user/system utterance and its information. Only the metadata fields from system utterances are filled, consisting of a summary of the topic tackled at the moment. Similarly to the *goal*, *metadata* is composed of an entry per domain, with a *book* field, in which the booking restrictions are stored, a *booked* field, which saves the information regarding the booking once it is completed, and a *semi* field, that saves the remaining information.

The annotations follow the ontology presented in Section 3.2.1. When a certain slot is not relevant for the user, the corresponding value is stored as *"don't care"*. If a certain slot was not tackled in the conversation, its value is annotated as *"not mentioned"*. An example of an annotated user utterance can be found bellow.

```
User: I'm looking for a cheap restaurant that serves modern european food.
Annotation: "restaurant": {
    "book": {
        "booked": [],
        "people": "",
        "day": "",
        "time": ""
    },
```

```
    "semi": {
        "food": "modern european",
        "pricerange": "cheap",
        "name": "don't care",
        "area": "not mentioned"
    }
}
```


The dataset is also composed of a *dialogue_acts* file. However, as there is no direct correspondence between these annotations and the dialogue utterances, they become less useful.

Database

In addition to the dialogues and their annotations, the dataset is composed of 7 database files, one for each possible domain of conversation. Each file consists of a set of domain entities, with the relevant attributes for the conversations. These attributes match the ontology presented in the previous section. An example of a database entry can be found bellow.


```
    {
        "address": "106 Regent Street City Centre",
        "area": "centre",
        "food": "indian",
        "id": "19214",
        "introduction": "curry garden serves traditional indian and bangladeshi cuisine
        cooked with fresh produce delivered every day",
        "location": [
            52.200187,
            0.126407
        ],
        "name": "curry garden",
        "phone": "01223302330",
        "postcode": "cb21dp",
        "pricerange": "expensive",
        "type": "restaurant"
    }
```


3.2.2 Preprocessing MultiWOZ

Before being put into use, the dataset should go through a preprocessing phase. The main ideas are explained in this section, whose implementation roughly follows the one provided in the MultiWOZ Repository. [1]

Delexicalization

According to Wen et al., 2017, a common practice in response generation is to break it into generate delexicalized sentences and then post-processing the system utterances. This delexicalization is crucial for the system to

[1] `https://github.com/budzianowski/multiwoz`, last accessed on 17-06-2020.

learn value independent parameters (Budzianowski and Vulić, 2019). To do so, for each domain, all possible values for each slot are collected from the database and then replaced by generic slots, such as *[value_address]* or *[value_pricerange]*. For domain specific vocabulary entities, the slots are replaced by *[domain_name]*. The numeric values are also replaced by *[value_count]*. This process is applied to all the dialogues sentences and not the annotations. This way, the model is less likely to hallucinate with domain specific vocabulary, while the information is accessible to be used. Besides this, the model will be more capable of generalizing to different domains and restrictions. Since the goal is to analyse the model's ability to generate sentences, they will not be post-processed to include specific entities, as the interaction with a database gets out of the scope of this work.

Apart from that, following the implementation used in GPT-2, all punctuation marks are added a space before, to make sure they are word independent in the tokenization phase. A similar approach is done with regular plural words, splitting the words into the singular word and *-s*.

```
User: Please locate me an italian restaurant in the centre area.
Delexicalized sentence: please locate me an [value_food] restaurant in the [value_area] area .
```

Adding to the presented advantages of delexicalizing the dataset, this process is necessary to allow the use of some evaluation metrics provided with the dataset, which will be presented in Section 3.5.

Database Pointer

The authors suggest creating a *database pointer*, which saves the information regarding the number of domain entities matching the user specifications, for the domains restaurant, hotel, attraction and train, if there are any. This information will be useful for the dialogue evaluation (Sections 3.5.3 and 3.5.4).

The database pointer has one-hot representation according to the number of available entities, which can be *zero*, *one*, *two*, *three*, *four* or *more than five*. Besides this, for domains requiring reservation (restaurant, hotel and train), the database pointer contains a booking pointer that informs about the availability of the entity at those specific restrictions. This booking pointer is `1 0` by default, and changed to `0 1` if booking is allowed at that stage of the dialogue. An example of the database pointer can be found bellow.

```
User: Please locate me an italian restaurant in the centre area.
Database_pointer:  0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 1 0 1 0 1 0
```

This case corresponds to the default composition of the database pointer, with all 4 domains having more than 5 entities (which corresponds to the first four sequences of 6 digits). The last 6 digits mean that it is not possible to book a restaurant, hotel or train at this point.

3.3 Implementation

After preprocessing the dataset, the implementation is done following Wolf et al. (2019)[2], using the smallest version of GPT-2. Similarly to Section 3.1.1, along with the Language Modelling task, the model is train in a Next-Sentence Prediction task, in which it learns to classify the correct response, given a set of possible answers. In each step, a distractor is randomly chosen from the dataset. Having the distractor chosen, the Language Model input will be fed twice into the model, one time with the correct gold answer, and the other with the distractor. As the labels are also

[2]The code is based on the Hugging Face Repository: `https://github.com/huggingface/transfer-learning-conv-ai`, last accessed on 13-04-2020.

provided to the model, it will learn to identify a correct answer from a wrong one, given a certain context. In the next sections, the Language Modelling task is explained in more detail.

3.3.1 Language Model Input

According to Radford and Salimans (2018) and Devlin et al. (2019), separation tokens are commonly added to separate the Transformer's inputs, which can be applied to the utterances of a dialogue. Therefore, the different parts of the *word_level* input are separated by the correspondent *dialogue_state embedding*, following Ham et al. (2020): <*belief*>, before the *belief_state*; <*db*>, before the *database_state*; <*usr*>, identifying the user utterance; and <*sys*>, identifying the system utterance. Besides, the input is initiated with a <*bos*> token (*beginning of sentence*) and finished with a <*eos*> token (*end of sentence*). Finally, <*pad*> tokens are added to the inputs, to make them have a constant length. These special tokens are added to the vocabulary, extending its size to $|\mathcal{V}| = 50264$.

In dialogue generation, the context is generally the whole conversation history. In this work, only the user's utterance is used, to which the optimal response will be learned, as the belief state solves the need of having the whole dialogue serving as context. This is an advantage for low computational power systems, as it tackles the issue of not being able to process a huge amount of tokens.

A scheme for the Language Model input can be seen in Figure 3.4, where the words are represented as tokens for easier visualization, despite not being totally correct, as GPT-2 uses BPE to tokenize the language, as seen in Section 3.1.2.

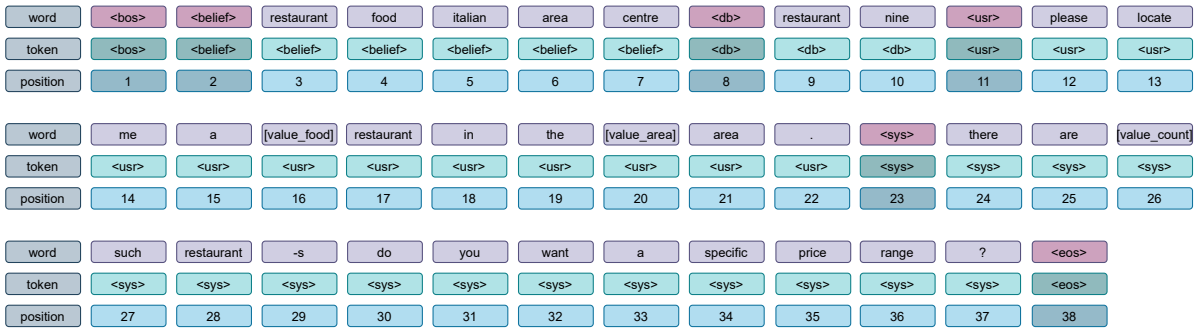


Figure 3.4: Language Model input.

Belief State

The belief state summarizes all the relevant information from the conversation history. In practice, it is composed of the informable and requestable slots with corresponding values in natural language, in the format:

```
Domain_1 Slot_1 Value_1 ... Slot_n Value_n ... Domain_k Slot_1 Value_1 ... Slot_m Value_m
```

All the metadata content is considered to the belief state, with the exception of the *"booked"* details, since they are considered too specific. Before each slot-value pair of booking restrictions, the word *"book"* should be added, to inform that these details belong to the booking domain as well. An example for the belief state can be found bellow.

```
User: Please locate me an italian restaurant in the centre area.
Belief_state: restaurant food italian area centre
```

In this work, the necessary information for the belief state construction is stored in the annotations. To move this to a practical context, with real interaction between a user and an assistant, a Dialogue State Tracker system should be developed, to identify the slots and values for each user utterance.

Database State

While the belief state contains a summary of the conversation context, the database state represents the database information. At each point of the conversation, a database pointer (Section 3.2.2) is computed, to inform about how many entities in accordance with to the given restrictions exist. The database state is a simple text representation of the database pointer, following the form:

`Domain_1 n_1... Domain_k n_k,`

with n_k being the number of possible entities of the k^{th} domain, converted to textual representation. An example for the database state can be found bellow.

```
User: Please locate me an italian restaurant in the centre area.
Database_state: restaurant nine
```

In a real application, once the belief state is extracted, obtaining the database state is trivial through a simple interaction with the domain database.

3.3.2 Softmax vs α-entmax

Once the input is ready, the model can start learning to generate text. In the training phase, it receives the input in the format of Figure 3.4, along with the language modelling labels — in other words, the gold system reply (Figure 3.5). The labels correspond to the original input, with everything but the system answer tokens being replaced by $<$*pad*$>$.

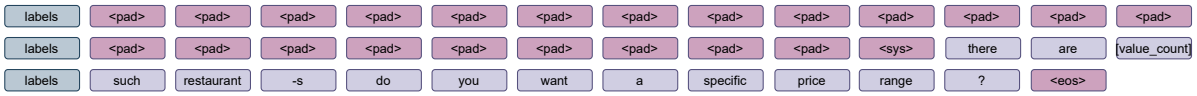


Figure 3.5: Language Model labels.

The tokenized inputs are fed into the decoder blocks, each of them composed of a Masked Self-Attention and a FFNN layer. Each layer produces independent scores over the masked input, which are then concatenated to be used by the FFNN layer. This layer uses the softmax or α-entmax distributions to project the scores into a vector of size $|\mathcal{V}| = 50264$. The loss between the produced scores and the provided labels is then calculated. Depending on the output distribution, the loss is either the cross-entropy loss, if the system uses the softmax distribution, or the α-entmax loss, in the case of the α-entmax distribution. The model's weights are then updated, in order to minimize the loss, as seen in Chapter 2. The model is trained by repeating this process while the loss values decrease.

In the generation phase, the input is similar to the training phase's, but with no system response, as schematized in Figure 3.6. The returned scores are then to be transformed into one of the two possible distributions, softmax or α-entmax, producing a vector of probabilities of size $|\mathcal{V}| = 50264$ over the vocabulary. The decoding strategy (Section 3.4) will define the way to select the value for the probability, determining the chosen token. Being a recursive model, the generated token is then added to the original input, taking part of the input for the next token generation. The generation process is conducted until the $<$*eos*$>$ token is generated, or until a maximum of 50 tokens per generated sentence.

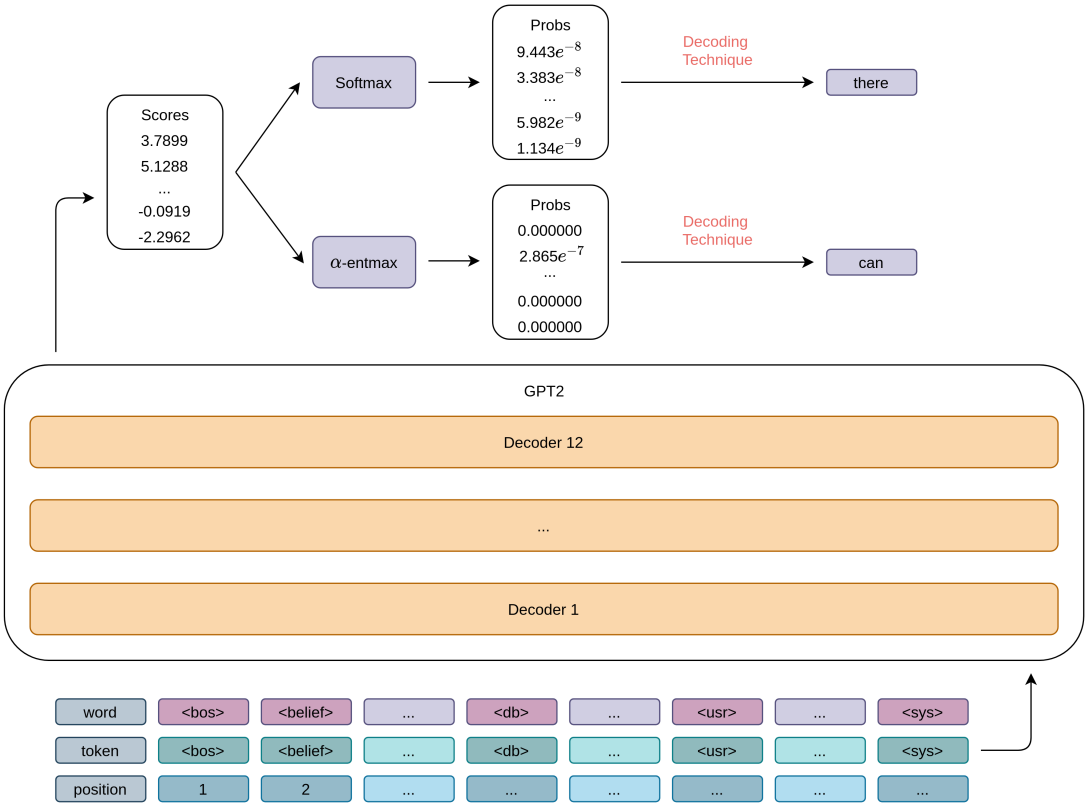


Figure 3.6: Simplified model — how to decode the next token.

3.4 Decoding Strategies

Given a probability distribution over the vocabulary, we select the next token based on the probability values. Therefore, the method chosen to select this probability — the decoding strategy — will have great influence on the generated sentence. In this section, an overview of some possible decoding strategies is performed.

3.4.1 Greedy Search

Once using **greedy search**, the chosen token w_t is the one with the highest probability value at each timestep t:

$$w_t = \underset{w}{\operatorname{argmax}}\, p_\theta(w|w_{<t}). \tag{3.2}$$

This is probably the most intuitive way to decode a sentence, as we could assume that the best results lie in the combination of the highest probability tokens. Nevertheless, some problems have been constantly reported throughout text generation studies, proving that this is not necessarily true. The most common problem of this strategy is the model being likely to start repeating itself, entering in a loop of text generation. Another fault lies in the fact that sometimes high probability phrases are “hidden” behind less likely words, which end up being ignored (Vijayakumar et al., 2016; Shao et al., 2017).

3.4.2 Sampling

Aiming at increasing the diversity of the output, **sampling** strategies avoid repetition by introducing stochastic decisions in the generation process, randomly picking the next word w_t given its conditional probability distribution:

$$w_t \sim p_\theta(w|w_{<t}). \tag{3.3}$$

To make the distribution $p_\theta(w|w_{<t})$ sharper, it is common to lower down the **temperature** of the softmax, which increases the higher probabilities and decreases the lower ones. This way, high likely words have more chance to be selected, while low likely ones get almost no chance. However, even with the temperature variable, one drawback of sampling is the ease to generate a very unlikely word, damaging the generation process and leading the model into strange paths (Holtzman et al., 2020).

3.4.3 Top-k Sampling

To colmat the problems described before, Fan et al. (2018) introduced **top-k sampling**. The idea is to restrict the sampling process to the *k* most probable words, reducing the probability of choosing out-of-the-box words, but also making the process more deterministic. The k most probable words are filtered and the probability mass is redistributed among them.

This sampling technique has shown to perform well, being considered to generate the most *human-sounding* text among the 3 introduced approaches. However, it has the drawback of not dynamically adapting the number of words which are filtered to the distribution. This means that sometimes the words can be sampled from a very sharp distribution, which can lead to the model producing strange text. On the other side of the spectrum, when it samples from a flatter distribution, its creativity can be limited by the number k.

3.4.4 Nucleus Sampling

Instead of relying on a fixed top-k highest probability words, Holtzman et al. (2020) proposes **nucleus sampling**, whose intuition comes from the fact that the vast majority of probability mass is concentrated in the nucleus. The idea is to sample from the smallest subset of words whose cumulative probability exceeds p, eliminating the possibility of choosing the less likely words.

The major advantage of this technique is allowing to contract and expand the number of candidates dynamically, depending on the probability distribution. Nucleus sampling has been used in many works, reported by many to be the most similar to the way humans speak.

3.4.5 α-entmax Sampling

The sampling techniques presented so far have been continuously tested and evaluated in text generation. However, there is a drawback in these sampling techniques: in generation time, the models sample from a new version of the softmax distribution, whose sparsity was not learned during training time.

On the other hand, applying the α-entmax transformation to the model scores also prevents implausible words from receiving any probability mass, which is the idea behind top-k and nucleus sampling. With **α-entmax sampling**, the chosen token w_t at time step t is:

$$w_t \sim p_\theta(w|w_{<t}) = \alpha\text{-entmax}(z_t(\theta, w_{<t})), \tag{3.4}$$

where z_t are the scores given by the model.

This decoding method has the advantage of eliminating the gap between training and testing conditions, in what comes to sparsity, as the model is already trained in a sparse distribution. Similarly to nucleus sampling, α-entmax sampling considers a varying number of tokens depending on the context, which is also an advantage.

3.5 Performance Evaluation

Given the different decoding strategies, it is necessary to find a way to fairly compare the quality of the generated text. According to Celikyilmaz et al. (2020), there are three types of methods to evaluate the performance of an answer generation system: automatic metrics requiring no training, machine learned metrics and human evaluation. In this section, relevant metrics which do not require human intervention will be presented, either automatic or machine learned ones. Along with the metrics proposed by Budzianowski and Vulić (2019) to evaluate this task (Inform Rate, Success Rate and BLEU), some others will be used, allowing a more detailed comparison between different techniques.

3.5.1 ϵ-perplexity

Perplexity (ppl) translates into the model's ability to predict the next word, given the context, being the inverse probability of the test set, normalized by the number of words (Jurafsky and Martin, 2019). On other words, the perplexity metric informs on how perplex is the model with the target sentence, meaning that a lower perplexity indicates a better model. In a Language Model, perplexity is the exponential average log-likelihood of a sequence. Given a sentence $w = (w_1, ..., w_T)$, it can be calculated as:

$$\text{ppl}(w) = \exp\left\{-\frac{1}{T}\sum_{t=1}^{T}\log p_\theta(w_t|w_{<t})\right\}, \tag{3.5}$$

with $\log p_\theta(w_t|w_{<t})$ being the log-likelihood of the token w_t, given the preceding tokens $w_{<t}$.

However, the computation of the logarithm of the probability distribution over the dataset reference can be problematic in sparse distributions, since $\lim_{p\to 0}\log p = -\infty$. With this in mind, Martins et al. (2020) propose ϵ-perplexity, a smoothed version of perplexity, obtained by adding ϵ to all the terms, followed by renormalization over the vocabulary size $|\mathcal{V}|$:

$$\epsilon\text{-ppl}(W) = \exp\left\{-\frac{1}{T}\sum_{t=1}^{T}\log\frac{p_\theta(w_t|w_{<t})+\epsilon}{1+\epsilon|\mathcal{V}|}\right\} \tag{3.6}$$

Like ppl, ϵ-ppl needs to be computed in test time, as it requires to see the dataset references one by one.

3.5.2 Sparsemax Score

Also proposed by Martins et al. (2020), the sparsemax score (sp) is based on the sparsemax loss from Martins and Astudillo (2016), which is the presented Equation 2.19 with $\alpha = 2$. This metric is defined as:

$$\begin{aligned}\text{sp} =& 1 - \min\{l_2(z,w)|\,\text{sparsemax}(z) = p_\theta\} && (3.7)\\ =& 1 - (p_\theta - e_w)^T p_\theta - H_2(p_\theta) && (3.8)\\ =& p_\theta(w) + H_2(p_\theta), && (3.9)\end{aligned}$$

where $H_2(p) := \frac{1}{2}\sum_j p_j(1-p_j)$ is the Gini Entropy. Similarly to perplexity, sparsemax score needs to be computed along the generation of the answers. However, sparxemax score is always bounded between 0 and 1.

3.5.3 Inform Rate

Proposed along with the MultiWOZ dataset, the inform rate aims at assessing whether the offered entity matches all the constraints specified in the user goal. In this context, the use of a delexicalized dataset prevents the inform metric from totally pursuing its goal, as no specific entity is given at any time. Nevertheless, it is possible to check if, given the context, there was indeed an available option in the dataset. To do so, the *database pointer* (Section 3.2.2) is analysed, which informs how many venues allow booking given the restrictions. A 0 inform rate means that the system hallucinated in the response, and no venue corresponded to the user's specifications.

3.5.4 Success Rate

The sucess rate is also suggested with the dataset, with the goal of evaluating whether all the requestable slots were provided to the user. Similarly to the inform rate, the success metric is calculated on the dialogue level. The dialogue is only considered successful if all the slots were filled during the dialogue. To calculate the success metric, the *goal* is investigated and matched with the system responses. The *database pointer* is also checked, only considering the task of providing a phone or reservation numbers successful in cases where the booking is allowed.

3.5.5 BLEU

Bilingual Evaluation Understudy (BLEU) (Papineni et al., 2002) measures the fluency of the answer by analysing the overlap of n-grams (sequences of n words) between the proposed response and a set of one or more reference sentences, regardless of the word order. Despite having been proposed for machine translation, BLEU is commonly used to evaluate other NLP tasks, being considered the dominant metric in language generation. The sacrebleu version will be used, which aims at uniformizing the use of the BLEU metric, whose parameters can vary, making it difficult to compare different implementations (Post, 2018).

3.5.6 METEOR

Metric for Evaluation of Translation with Explicit Ordering (METEOR) (Banerjee and Lavie, 2005) is also proposed for machine translation, based on alignments between the generated sentences and one or more references. These alignments are sets of matchings between unigrams in two different strings, such that every word in each string maps to at most one word in the other string. The matches occur if there is an exact match (the words are identical), stem match (the words have similar stems), or synonymy match (based on an external database).

3.5.7 BERTScore

Similarly to BLEU, BERTScore (Zhang et al., 2020a) calculates a similarity between each token of the generated sentence and each token of the original sentence. However, instead of n-gram matching, this similarity is computed as a sum of cosine similarities between contextual embeddings of the tokens (Section 2.3.2), as it uses prior knowledge from BERT to originate the tokenized text. Hence, BERTScore has some context awareness, which

is very relevant to evaluate NLP tasks. Focusing on token level semantic similarity, BERTScore has been proved to be particularly effective in paraphrase detection (Devlin et al., 2019).

Chapter 4

Experiments

In this chapter, the different approaches presented in the previous chapters are experimented and evaluated. Firstly, some hyperparameters are tuned in Section 4.1, to optimize each model's performance. In Section 4.2, a comparison between the different techniques is performed, allowing to draw some conclusions regarding the systems. Finally, the results are thoroughly discussed in Section 4.3, with reference to relevant specific situations and examples.

4.1 Hyperparameter Tuning

To ensure the models are evaluated at their finest performance, some training hyperparameters should be firstly tuned:

- Language Modelling Coefficient (LM_{coef}) — is the balance between $Loss_{LM}$ and $Loss_{MC}$ for the loss calculation (see Equation 2.10). Since the task we are focusing on is the Language Modelling task, we should naturally give more importance to $Loss_{LM}$ than $Loss_{MC}$. Therefore, MC_{coef} is kept at 1, and a grid search is made for $LM_{coef} \in \{1, 2, 4\}$.
- Learning Rate (η) — also referred to as step size, is the value η in Equations 2.7 and 2.8. It represents the amount that the weights are updated, controlling how quickly the model is adapted to the problem. A grid search over this parameter is performed for $\eta \in \{1e^{-5}, 5e^{-5}, 1e^{-4}, 5e^{-4}, 1e^{-3}\}$.
- Batch Size (B) — the size of the batch to update the weights using Mini-Batch Gradient descent, as referred to in Section 2.1.1.
- Gradient Accumulation Steps (S) — it is used to accumulate the losses and gradients over several batches, only updating the weights after a given number of steps. In practice, it increases the effective batch size in S times.

Since there is not much flexibility in terms of memory usage, the values for B and S are be kept at $B = 4; S = 8$, following the original implementation from Wolf et al. (2019). It is relevant to note that, in a first phase, other combinations such as $B = 4; S = 4$ and $B = 2; S = 12$ were experimented, leading to worse results given fixed values for LM_{coef} and η.

In all cases, the training phase was performed for as long as the loss value decreased, with a maximum of 10 epochs per training. At this stage, as the goal is to tune the parameters, a single metric is used to have an idea

of the model behaviour. The chosen metric, usually used in MultiWOZ works, consists of a balance between the Inform, Success and BLEU metrics, to be referred to as tune metric: $0.5 \times (\text{Inform} + \text{Success}) + \text{BLEU}$.

4.1.1 Softmax

Despite previous works having constantly reported results for $LM_{coef} = 2$, we tune this parameter in order to understand the MC head importance in the problem, alongside with η. The results are reported for the tune metric on the validation set. To have an idea of which parameters the model benefits the most, a grid search is made for the two techniques on the opposite sides of the sampling spectrum: sampling and greedy search.

Sampling

Table 4.1: Grid search over η and LM_{coef} for sampling.

	$LM_{coef} = 1$	$LM_{coef} = 2$	$LM_{coef} = 4$
$\eta = 1e^{-5}$	57.10	60.97	55.10
$\eta = 2.5e^{-5}$	65.48	67.21	66.03
$\eta = 5e^{-5}$	68.13	70.59	70.02
$\eta = 7.5e^{-5}$	69.72	**72.33**	71.89
$\eta = 1e^{-4}$	70.13	70.15	70.72
$\eta = 5e^{-4}$	67.10	66.50	65.70
$\eta = 1e^{-3}$	63.04	65.66	66.57

From Table 4.1, the value for LM_{coef} seems to have little influence on the final result. This can be due to the fact that, as the evaluated task is Language Modelling, the model does almost not benefit from the MC head. Nevertheless, $LM_{coef} = 2$ generally gives slightly better results, confirming the experiences from previous approaches. In what comes to the η, the optimal performance appears to be between $\eta = 5e^{-5}$ and $\eta = 1e^{-4}$. To better tune this parameter, the results for $\eta = 2.5e^{-5}$ and $\eta = 7.5e^{-5}$ are also reported, having the best value for η is $7.5e^{-5}$.

Greedy Search

Table 4.2: Grid search over η and LM_{coef} for greedy search.

	$LM_{coef} = 1$	$LM_{coef} = 2$	$LM_{coef} = 4$
$\eta = 1e^{-5}$	75.13	77.86	74.52
$\eta = 2.5e^{-5}$	84.34	82.85	84.22
$\eta = 5e^{-5}$	88.03	87.96	**90.73**
$\eta = 7.5e^{-5}$	90.05	**86.95**	88.25
$\eta = 1e^{-4}$	86.10	88.27	88.73
$\eta = 5e^{-4}$	87.95	82.54	86.83
$\eta = 1e^{-3}$	84.34	85.01	88.31

Table 4.2 provides the confirmation that the value for LM_{coef} has little influence on the final result, not being possible to draw conclusions regarding which value conducts to the optimal performance. However, contrary to what was previously reported, the best performance is no longer with $LM_{coef} = 2$. Highlighted in the table, we have the values for the chosen parameters from the previous section and the optimal performance for greedy search.

At this point, it is clear that greedy search leads to an overall better performance than simple sampling. One intuition can be that, in a goal-oriented setting, the type of conversations are well structured, being the sentence construction more deterministic. Nevertheless, a further analysis will be performed in the following sections.

Top-k Sampling

For top-k sampling, we look at $k \in \{5, 10, 20, 50, 100\}$. The results with greedy-search are also presented for comparison, as this decoding technique corresponds to top-k sampling with $k = 1$. Results for the both best combinations reported previously are included: $LM_{coef} = 2$, $\eta = 7.5e^{-5}$; $LM_{coef} = 4$, $\eta = 5e^{-5}$.

Table 4.3: Grid search over k for top-k sampling.

	$k = 1$	$k = 5$	$k = 10$	$k = 20$	$k = 50$	$k = 100$
$LM_{coef} = 2$, $\eta = 7.5e^{-5}$	86.95	**76.72**	71.72	72.04	72.03	70.06
$LM_{coef} = 4$, $\eta = 5e^{-5}$	90.73	76.67	74.42	71.73	70.03	68.35

Analyzing the registered values, one can detect a decrease in the model's performance once the value of k is augmented. There is a strong contrast between the scores obtained with greedy search (k=1) and with other values of k.

Nucleus Sampling

In this section, the parameter p is tuned for $p \in \{0.5, 0.7, 0.8, 0.9\}$. Given the previous results, the chosen combination is for $LM_{coef} = 2$, $\eta = 7.5e^{-5}$. It is relevant to underline that the sampling technique is equivalent to nucleus sampling with $p = 1$, hence its results being present. Besides, the temperature parameter will also fluctuate with the different values of p, for $temp \in \{0.7, 0.8, 0.9\}$.

Table 4.4: Grid search over p and $temp$ for nucleus-sampling.

	$temp = 1$	$temp = 0.9$	$temp = 0.8$	$temp = 0.7$
$p = 1$	72.33	71.62	70.99	68.57
$p = 0.9$	85.51	82.30	84.05	83.00
$p = 0.8$	84.68	83.73	85.93	85.06
$p = 0.7$	84.26	84.05	**87.18**	85.59
$p = 0.5$	85.74	86.46	84.73	86.41

From Table 4.4, there seems to be no direct relationship between the evolution of the parameters p and $temp$ and the quality of the responses. The optimal performance is for $p = 0.7$ with $temp = 0.8$.

4.1.2 α-entmax

In the case of α-entmax, we are in the presence of a different family of loss functions. Thus, it is not correct assume that the values which best fit experiments with softmax will have the same behaviour for α-entmax. The parameters η and LM_{coef} are therefore tuned again, for a fixed value of α. Given the results reported by Martins et al. (2020), the chosen value is $\alpha = 1.5$, the most adequate value for the task of dialogue generation, in an open-domain framework.

α-entmax Sampling

Table 4.5: Grid search over η and LM_{coef} for 1.5-entmax sampling.

	$LM_{coef} = 1$	$LM_{coef} = 2$	$LM_{coef} = 4$
$\eta = 1e^{-5}$	51.81	52.69	52.43
$\eta = 2.5e^{-5}$	61.86	63.50	64.97
$\eta = 5e^{-5}$	61.76	65.07	**69.99**
$\eta = 7.5e^{-5}$	55.00	**69.94**	68.53
$\eta = 1e^{-4}$	66.91	63.25	66.19
$\eta = 2.5e^{-4}$	64.82	67.90	66.78
$\eta = 5e^{-4}$	67.98	64.04	68.64
$\eta = 1e^{-3}$	67.51	66.74	67.06

Contrary to sampling from softmax, it looks clear that the model performs better for higher values of LM_{coef}. However, it is difficult to draw a conclusion for η, hence the necessity of reporting results for values in between the proposed ones: $\eta = 2.5e^{-5}$, $\eta = 7.5e^{-5}$ and $\eta = 2.5e^{-4}$.

Ideally, all possibilities of α should be tuned as in Table 4.5, but it is impossible to present experiments with all variations of values, as this process would become too expensive. We therefore present results for other possible models with fixed LM_{coef} and η, and $\alpha \in \{1.1, 1.2, 1.3, 1.4, 1.5, 2\}$. Values for $\alpha = 1$ are also included, allowing to compare with sampling from softmax.

Table 4.6: Grid search over α for α-entmax sampling.

	$\alpha = 1$	$\alpha = 1.1$	$\alpha = 1.2$	$\alpha = 1.3$	$\alpha = 1.4$	$\alpha = 1.5$	$\alpha = 2$
$LM_{coef} = 2$, $\eta = 7.5e^{-5}$	**72.33**	-	68.03	67.72	69.86	69.94	64.29
$LM_{coef} = 4$, $\eta = 5e^{-5}$	70.02	70.81	**72.22**	67.44	-	69.99	62.50

Given the scores documented in Table 4.6, the optimal value for α is $\alpha = 1.2$, which will be used in the following section. Nevertheless, Tables 4.5 and 4.6 suggest that there is no improvement from simple sampling from softmax to α-entmax.

Greedy Search from α-entmax

Given the apparent good performance of more deterministic methods, we were curious to understand the sparsity influence in greedy search. Therefore, experiments were also performed with greedy search from α-entmax, which means to choose the top-1 probability word from the entmax distribution, at decoding time. Results for this technique are presented in Table 4.7, for the best reported LM_{coef} and η, and various values of α.

Table 4.7: Grid search over α for greedy search from α-entmax.

	$\alpha = 1$	$\alpha = 1.1$	$\alpha = 1.2$	$\alpha = 1.3$	$\alpha = 1.4$	$\alpha = 1.5$	$\alpha = 2$
$LM_{coef} = 2$, $\eta = 7.5e^{-5}$	86.95	-	90.35	86.64	**92.30**	86.34	72.62
$LM_{coef} = 4$, $\eta = 5e^{-5}$	90.73	**91.29**	89.33	80.03	-	87.65	67.22

In accordance to what was previously seen, there is a significant improvement in the reported values from Table 4.7, which suggest that, in Task Oriented Dialogues, the diversity imposed by sparse methods may be not important

for the dialogues quality. The optimal system has $\alpha = 1.4$, the chosen value for this technique.

4.2 Results

After tuning each model's hyperparameters, a fair comparison between the different decoding techniques can finally be made. To do so, we chose the models for the highest results in Section 4.1 and ran them in the Test Set. Besides the tune metric, the automatic metrics introduced in Section 3.5 are presented. Since the goal is to compare different sampling techniques in text generation, Table 4.8 comprises only one example for each decoding technique.

Table 4.8: Different models performance for MultiWOZ 2.0.

	ϵ-ppl (↓)	sp (↑)	Inform (↑)	Success (↑)	BLEU (↑)	BERTScore (↑)	METEOR (↑)	Tune Metric (↑)
Sampling	**2.3074**	**0.8443**	64.4	37.1	21.02	24.36	18.91	71.77
Greedy	25.3800	0.7695	64.6	53.4	**29.46**	**34.81**	**19.52**	88.46
Top-k	3.8036	0.8412	65.4	44.2	24.43	28.89	19.06	79.23
Nucleus	10.2294	0.8204	65.2	54.6	27.80	32.53	19.43	87.18
α-entmax sampling	2.4922	0.8440	63.4	35.8	21.26	24.11	18.87	70.86
α-entmax greedy	25.6698	0.7686	**66.8**	**57.4**	29.44	33.38	19.34	**91.54**

From Table 4.8, it is possible to draw some initial conclusions regarding the systems' performance. The values for ϵ**-ppl** are in accordance to what was expected, with low values for stochastic methods and high values for more deterministic ones. The opposite happens for **sparsemax score**, with higher scores attributed to random sampling techniques. These values suggest that, despite the performance at the other metrics, greedy systems are more perplex with the dataset reference, meaning they would less likely generate it. Regarding the dataset specific metrics, **Inform** presents similar values throughout the multiple sampling techniques. This regularity suggests the systems' ability to effectively attend to the database state information. On the other hand, the **Success** metric presents a diverse range of values, proposing the best performance is for *greedy*, *nucleus* and *greedy search from α-entmax*. In what comes to machine translation metrics, both **BLEU** and **BERTScore** are evidences for the optimized fluency of these three techniques. The values for **METEOR** suggest a similar behaviour, but in a very shorter scale, which seems to not allow to draw any conclusions.

The best overall performance is for greedy search from α-entmax, presenting the highest score for Inform and Success, and therefore to the tune metric also. It is closely followed by greedy search and nucleus sampling, with top-k afterwards, and sampling and α-entmax sampling being the last in the ranking.

4.2.1 Recent versions of MultiWOZ

To allow the comparison with the most optimized version of the MultiWOZ dataset, results for the 2.1 and 2.2 versions will also be released, for the same decoding techniques.

Table 4.9: Different models performance for MultiWOZ 2.1.

	ϵ-ppl (↓)	sp (↑)	Inform (↑)	Success (↑)	BLEU (↑)	BERTScore (↑)	METEOR (↑)	Tune Metric (↑)
Sampling	**2.3982**	0.8375	65.5	35.7	20.25	23.84	18.92	70.85
Greedy	29.4990	0.7586	62.3	52.0	28.63	**35.14**	**19.42**	85.78
Top-k	4.0894	0.8340	66.5	42.6	23.41	28.60	19.01	77.96
Nucleus	11.4130	0.8122	64.3	49.9	27.09	32.51	19.35	84.19
α-entmax sampling	2.5723	**0.8388**	65.4	38.9	20.78	24.28	18.93	72.93
α-entmax greedy	28.9647	0.7599	**67.7**	**57.1**	**28.81**	34.50	18.48	**91.21**

Table 4.10: Different models performance for MultiWOZ 2.2.

	ϵ-ppl (↓)	sp (↑)	Inform (↑)	Success (↑)	BLEU (↑)	BERTScore (↑)	METEOR (↑)	Tune Metric (↑)
Sampling	**2.3465**	0.8416	63.7	29.3	20.53	23.57	18.96	67.03
Greedy	26.7725	0.7656	65.8	**42.7**	29.17	**34.21**	**19.54**	**83.42**
Top-k	3.9143	0.8384	**67.0**	34.2	23.61	27.84	19.16	74.21
Nucleus	10.7412	0.8172	66.0	40.2	27.71	31.98	19.39	80.81
α-entmax sampling	2.6947	**0.8421**	63.1	29.0	20.55	23.80	18.80	66.60
α-entmax greedy	27.4608	0.7638	63.8	41.7	**29.35**	33.78	19.45	82.10

While the majority of the scores is similar to the reported in Table 4.8, there is a slight decrease in the Success metric in MultiWOZ 2.1 (Table 4.9), followed by a significant drop in MultiWOZ 2.2 (Table 4.10). Although there are not reported results comparing the task of answer generation with more recent versions, we assume this is an expected behaviour, as one of the improvements consisted in annotating more requested slots for each user turn, resulting in more requirements for a dialogue to be considered successful.

4.2.2 Context Importance

Along with investigating how different sampling techniques can influence the answer generation, the importance of the context in the systems' architecture can be questioned. On a real life application, identifying the belief state and the database state requires the implementation of a Dialogue State Tracker, which is likely to introduce errors in the system. To answer this question, some experiments with different formats for the context were carried out.

The results presented in Table 4.11 are for experiments with only database state and last utterance (no belief state), only belief state and last utterance (no database state), only last utterance (no belief state and no database state) and the last 3 utterances, with results for standard context also present for comparison. Each system was trained and tested with the same context format. We chose to present the results using nucleus sampling at test time, as it is the reported best sampling technique in Budzianowski and Vulić (2019).

Table 4.11: Nucleus sampling performance for different types of context.

	ϵ-ppl (↓)	sp (↑)	Inform (↑)	Success (↑)	BLEU (↑)	BERTScore (↑)	METEOR (↑)	Tune Metric (↑)
Full context	10.2294	0.8204	65.2	54.6	27.80	32.53	19.43	87.18
No belief state	14.4673	0.7999	50.5	35.3	25.44	29.22	19.04	68.34
No database state	14.0166	0.8024	52.7	36.4	26.02	29.47	19.07	70.57
Only last utterance	13.5958	0.8045	51.3	36.4	25.30	29.54	19.00	69.15
Last 3 utterances	11.7259	0.8119	67.3	52.1	25.21	23.12	19.11	84.91

There is an evident drop in performance when some parts of the context are omitted, as seen in lines 2, 3 and 4 of Table 4.11, confirming the importance of having this structured information. These experiences report very similar results, suggesting a situation where the given context is not enough to properly determine which direction the system should take. The most surprising result is definitely for experiences with only the last 3 utterances. This arrangement seems to operate very well, even outperforming sampling, top-k and α-entmax sampling in most of the metrics. However, the low value for BERTScore suggests some faults in the generated text.

4.3 Discussion

Although the metrics allow to have an idea of the model's performance, some results can be considered surprising and ask for further inspection. After an extensive analysis of the multiple answer suggestions in diverse dialogues, it

is possible to understand that the different decoding strategies have similar performance in many cases, particularly when the dialogue has a standard format. On the other hand, some specific situations originate a contrast in the performance of the different techniques. These issues are discussed in this section, as they are relevant to understand where the systems fail. Some of these aspects can not be grasped by the metrics previously presented, which makes this discussion more relevant.

Context importance

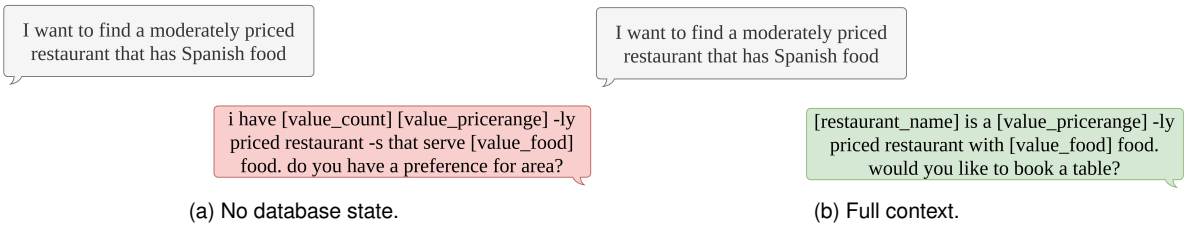
The first relevant findings are related to the context importance. Following the conclusions taken from Table 4.11, a closer look at the generated sentences clearly shows a better understanding of the conversation history from the systems which include the whole standard context, presented in Section 3.3.1.

The most prominent cases reflect the importance of the belief state, where its absence leads the system to not understand the user's restrictions, or in the case of Figure 4.1, simply forgetting what has been mentioned in the conversation. As previously pointed out, the belief state is crucial for the awareness of the dialogue history.

(a) No belief state. (b) Full context.

Figure 4.1: Example demonstrating the belief state importance.

Another relevant appointment is regarding the importance of the database state, which makes the system to be aligned with the available entities from the database, allowing to have the correct reaction to the user request. For instance, in the situation of Figure 4.2, there is only one entity available given the restrictions, fact of which the dialogue with the standard context is aware. However, the system with no database information incorrectly asks for more restrictions, poorly shaping the direction of the conversation.



(a) No database state. (b) Full context.

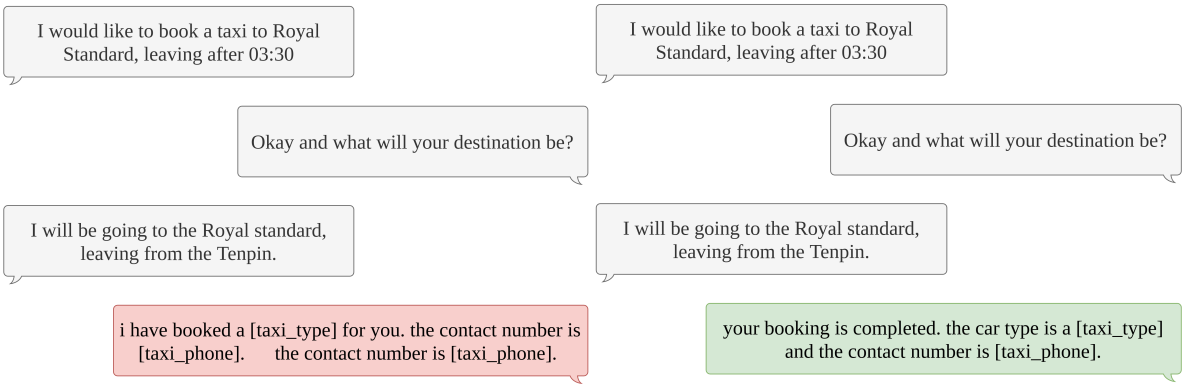
Figure 4.2: Example demonstrating the database state importance.

By inspecting the several systems' performance, a prevalent quality is the ability to attend at the database state, rarely failing in informing whether there are available entities, which suggests that this architecture allows to understand simple information from the input text. We can therefore speculate that this could be extended to other

types of information, depending on the desired application and type of human intervention. One example could be to include information regarding the possibility (or lack of it) of pursuing a booking in the text input, to which the system would very likely correctly attend and originate answers according to it.

Although these situations demonstrate the utility of the belief and database states, the performance of the system with 3 utterances as context does not corroborate this conclusion. The main surprise lies in the values for Inform and Success, which are supposed to measure the more objective part of the system's performance. However, if we rethink the way the metrics are calculated, we can understand that they do not necessarily translate the quality of the answers: Inform simply looks at whether an entity was correctly proposed; Success evaluates if the requestable slots were provided. This means that, even if the system misunderstands the specificities provided by the user, it can randomly suggest an entity (when it should be the case) and provide the necessary requestables, being wrongly considered an informable and successful dialogue.

Besides this, another aspect regarding the system with conversation history of length 3 is that it frequently produces repetitions and degenerate text, confirming the suspect of faults raised by the value for BERTScore. This regular occurance observed exclusively in this system appears to be due to the introduction of multiple dialogue utterances, leading the system to a state of confusion.



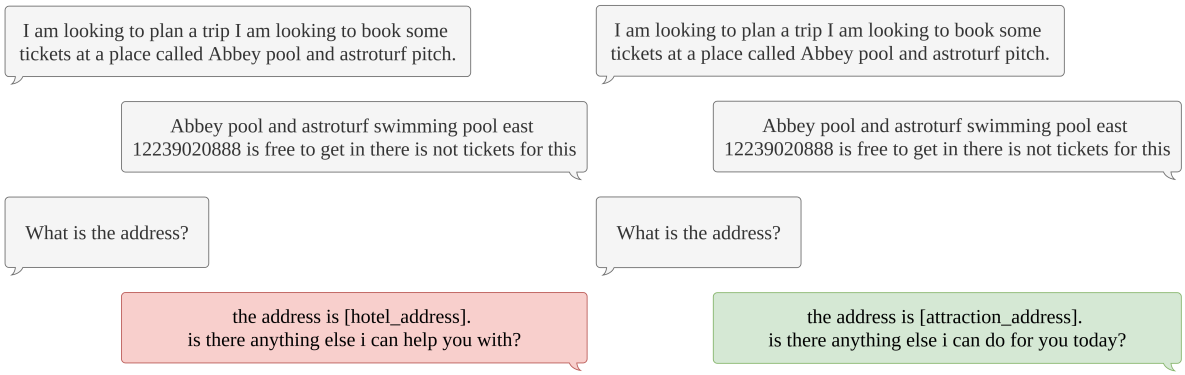
(a) Last 3 utterances. (b) Full context.

Figure 4.3: Example of degenerate text.

From the analysis of the dialogues with different types of context, we can conclude that metrics may sometimes fail in detecting the bad performance of the systems.

Inadequate slot generation

When going through the generated answers, a clear mistake is regarding the use of the wrong general slots, given the conversation domain. On example can be the generation of *[hotel_address]* by nucleus sampling, instead of the correct term *[attraction_address]*, which was generated by the other techniques (Figure 4.4). We could deduce this inadequacy comes from mistakes in the annotations, but dialogues with annotation mistakes will be kept out of the thorough evaluation procedure, as their results can be misleading. Nevertheless, their presence in the training phase can have some influence on the systems' performance. It is relevant to note that these types of mistakes have an impact on the Inform and Success metrics, as they are calculated based on these slots.

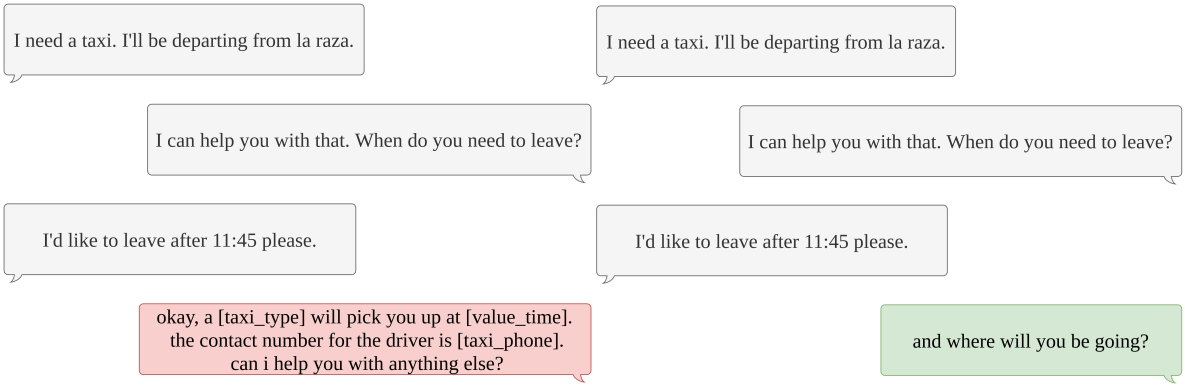


(a) Nucleus sampling. (b) 1.2-entmax sampling.

Figure 4.4: Example of inadequate slot generation.

Booking without enough information

One recurring fault of the systems is not recognizing there is missing information to succeed in performing a booking. As an example, Figure 4.5 presents a scenario where the user needs a taxi, which is booked without having a destination. Nevertheless, the metrics are able to grasp this information, as they only consider a booking successful when the booking pointer was allowing for that.



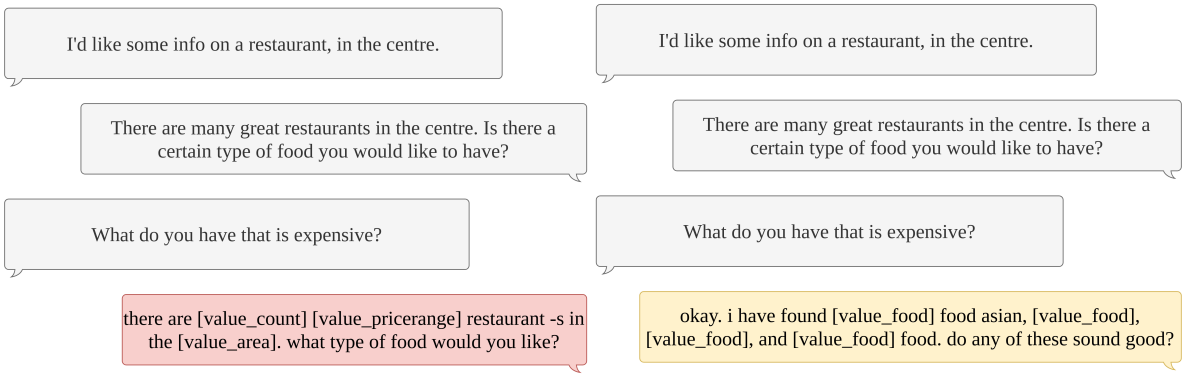
(a) 1.2-entmax sampling. (b) Nucleus sampling.

Figure 4.5: Example of booking without enough information.

Insisting instead of suggesting

As the goal of these conversations is to provide support to a user in an unknown city, many dialogues require that the system suggests an entity. In some conditions, the process of giving a suggestion is simple, as the user gives enough details to immediately narrow down the possibilities. However, there are some cases where the user is looking for a simple suggestion, without the need of giving many details. In these situations, some techniques struggle to move forward in the conversation, being stuck in the process of narrowing down the search.

In the case of Figure 4.6, despite the grammar incorrections, α-entmax sampling is the only technique which answers to the user request, while the others ask for any type of food specifications.

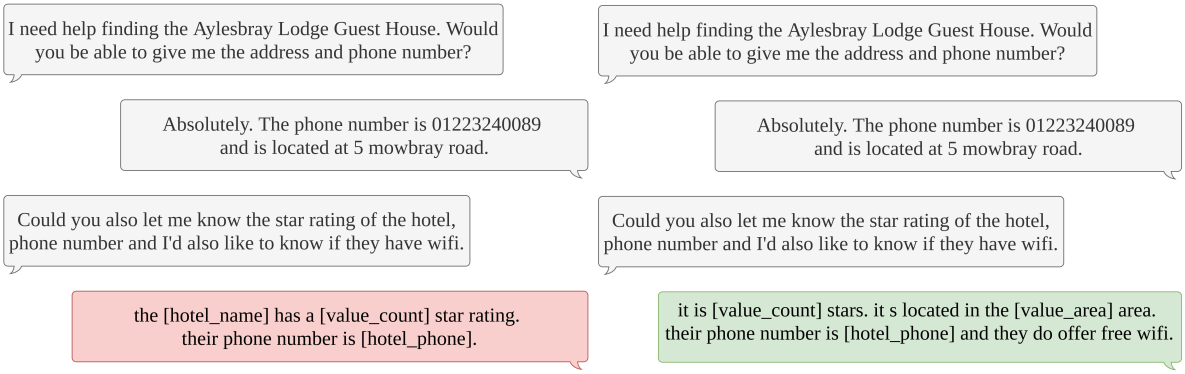


(a) Greedy search. (b) 1.2-entmax sampling.

Figure 4.6: Example of insisting instead of suggesting: type of food.

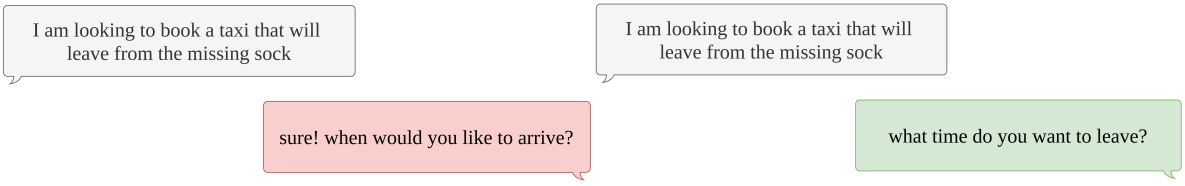
Low understanding of the user utterance

Another type of failure comes from the low understanding of the user's intentions when the information is not present in the belief state. As an example, in Figure 4.7 the user is asking for specific information regarding a domain entity (a hotel, in this case). With exception of top-k sampling, all the techniques succeed in providing the phone number and star rating, while failing to answer the wi-fi query. This case is not contemplated in the Inform metric, as the system can still randomly suggest an entity when there is an available one (or the opposite), with or without wi-fi, neither in the Success metric, as the wi-fi query does not take part of the booking requestables.



(a) Greedy search. (b) Top-k sampling.

Figure 4.7: Example of low understanding of the user utterance: wi-fi query.



(a) Greedy search from 1.4-entmax. (b) 1.2-entmax sampling.

Figure 4.8: Example of low understanding of the user utterance: taxi query.

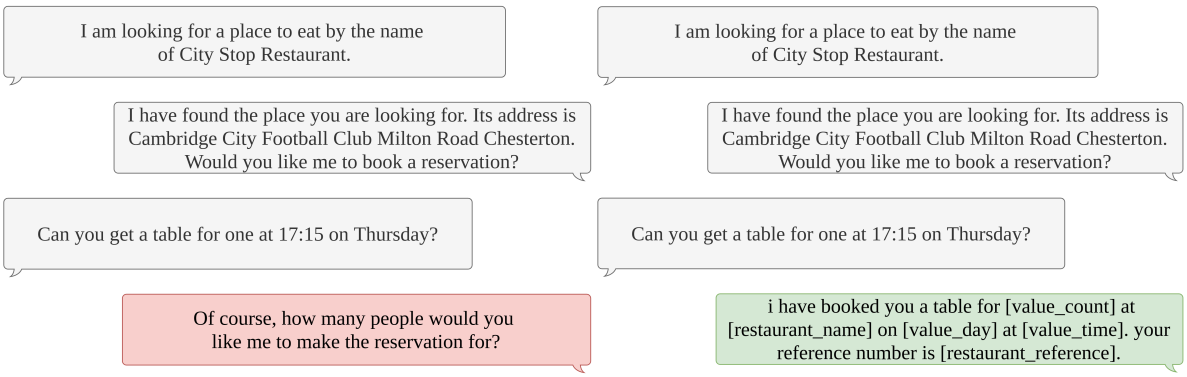
Another example of low understanding of the user utterance can be the generation of a question which does not make much sense given the context. In Figure 4.8, there is an example of two possible answers to a costumer

looking for a taxi, but one of them is more relevant and natural than the other.

Both these situations may be interpreted as an *"overfitting"* scenario, where the system is unable to adapt to the user question and repeats what it has seen in the training data, when in the presence of a certain belief state.

Outperform dataset reference

Despite the flaws present in all the systems, there are some cases where they outperform the reference answer. In Figure 4.9, there is an example where the dataset reference shows low understanding of the user's utterance, while the generating systems provide a more adequate reply. This situation makes it clear that the metrics which use sentence similarity to measure the quality of generated responses may not always translate into a good evaluation.



(a) Dataset reference.

(b) Greedy search from 1.4-entmax.

Figure 4.9: Example of system outperforming dataset reference.

Throughout this section, some systems' particularities have been presented, raising the question of whether the evaluation methodology is able to grasp them. At one side, despite being created to evaluate MultiWOZ dialogues, Inform and Success can not be measured at the turn level, missing some important aspects of dialogue construction. At the other side, BLEU, METEOR and BERTScore were not designed for this task, but are able to be measured at the turn level.

Since the ultimate goal of a goal-oriented dialog system is to support the user in a task accomplishment, it is important to understand if the reported metrics are reliably reproducing the human perception of quality. Liu et al. (2016) observed that metrics such as BLEU and METEOR correlate poorly with human judgment in open-domain dialogues generation, as they assume that valid responses have significant word overlap with the ground truth responses. This poor correlation arises from the fact that, in a dialogue, there are many acceptable responses to an input context, known as the one-to-many problem (Zhao et al., 2017). The opposite has been reported by Sharma et al. (2017) in goal-oriented dialogue systems, finding automatic metrics to strongly correlate with human judgement. However, this study was based on a pipeline approach to answer generation, evaluating the quality of the translation from dialogue acts into a proper sentence. This task can be considered similar to a machine translation task, contrasting with the one to be evaluated in this work. As the generation process evolves towards an end-to-end approach, some conclusions should be revisited, to be done in the following chapter.

Chapter 5

Human Evaluation

Despite their diversity and constant evolution, automatic metrics should not be used alone to draw conclusions about a system's performance. Van Der Lee et al. (2019) pointed out that they present interpretability problems, as multiple text generation which receives the same low score can fail in very different ways. Hashimoto et al. (2019) also state that automatic evaluation captures diversity but not quality, while the opposite happens with human evaluation. State-of-the-art works usually resort to human evaluation, which allows to perceive different aspects of text generation.

To understand how representative the automatic scores are of the sentences' quality, a human annotation process is conducted in this chapter. A set of evaluation dimensions is developed and presented in Section 5.1, followed by a description of the annotation procedure in Section 5.2. Finally, the human evaluation results are presented in Section 5.3, where we calculate the correlation between the presented metrics and human judgement, drawing conclusions regarding their relevance to evaluate these systems.

5.1 Evaluation Dimensions

Human evaluation can be categorized in two: static, where the human evaluator assesses the dialogue with only the last utterance being generated by the system; and interactive, where the human evaluates the whole dialogue after directly interacting with the system (Finch and Choi, 2020). Given the impossibility of implementing an interactive evaluation in an only turn-level system, this work requires a static evaluation.

5.1.1 Human Metrics from Literature

One conventional field of goal-oriented dialogue evaluation is the *Task-Success Rate*, measuring how well the dialogue system fulfills the information requirements dictated by the user goals (Deriu et al., 2020). In this work, although the Inform and Success automatic metrics are representative of this metric, some aspects are not grasped, such as the provision of informable slots. Besides, these metrics evaluate the Inform and Success rates of a dialogue, not being suitable for evaluating the task of generating a single turn.

Across goal-oriented dialogue evaluation work, little has been done regarding the turn level generation. Therefore, common practices from open-domain will serve as inspiration to build a set of dimensions for human evaluation, in order to assess the adequateness of each developed answer. There are multiple approaches in state-of-the-art work: for example, while Liu et al. (2016) use only *'adequacy'* to measure the quality of a response; Mehri and

Eskenazi (2020) annotate 10 dimensions in turn-level fields, such as *'Interesting'*, *'Relevant'* and *'Fluent'*. Huang et al. (2020) outline the importance of *'Semantics'*, to understand the content of the conversation and its implications beyond the dialogue itself, *'Consistency'*, not to provide contradicting information, and *'Interactiveness'*, to maximize the long term user engagement. Finch and Choi (2020) analyze 20 relevant papers on open-domain dialogues since 2018 and found a huge variability in the dimensions for the human evaluation setup, with 21 different dimensions. After clustering them, a group of 8 broader dimensions was attained: *Grammaticality*, *Relevance*, *Informativeness*, *Emotional Understanding*, *Engagingness*, *Proactivity*, *Consistency* and *Quality*. Aiming at increasing the consistency on how human evaluations are run, the conclusions of this survey will serve as a base for the choice of dimensions to evaluate in this work.

5.1.2 Proposed Evaluation Dimensions

From the 8 dimensions presented above, 3 were not considered relevant aspects for a goal-oriented conversation, and therefore discarded: *Emotional Understanding*, assessing if an appropriate reaction is provided given the emotional state of the user, *Engagingness*, to evaluate how engaging the responses are, and *Proactivity*, showing if the responses actively move towards different topics. That being said, the final dimensions are further explained in the next sections: *Grammaticality*, *Informativeness*, *Relevance*, *Consistency* and *Overall Quality*.

Grammaticality

Grammaticality, adopted by Zhu et al. (2019), is also referred to as fluency or readability. It evaluates the construction of the sentence, detecting if it is free of grammatical and semantic errors. It is scored in a scale of 1–3:

1. The answer is not fluent at all, being poorly structured and almost not understandable.
2. There are some minor flaws, but the sentence is understandable.
3. The answer is fluent and grammatically perfect, with no flaws.

Informativeness

Informativeness, which is used by Wu et al. (2019), assesses whether the system's answer brings relevant information to the table. Also seen as how helpful is the system in moving forward in the conversation, it can be measured considering the amount of user queries tackled by the system, in more complex dialogues. In a scale of 1–3:

1. The reply is not informative and not helpful in reaching the conversation goal.
2. The answer is slightly informative, but one would like to get some more information at that point. It can be used when not all the user queries are covered.
3. The system utterance is totally informative, tackling all the required queries.

Relevance

Relevance aims to evaluate whether the response is appropriate given the user query, according to Qiu et al. (2019). Also used as appropriateness, it is measured using a 1–3 scale:

1. The response does not make any sense, as it is completely out of context.
2. The answer can not be considered inappropriate, but there were more relevant aspects to tackle or replies to be given at that stage.
3. The system reply is the most appropriate given the user utterance.

Consistency

Consistency was introduced to assess any contradictions within the dialogue, both with system and user utterances. It can also be seen as how aware is the system of the dialogue history, which is rather important in goal-oriented dialogues. In a scale of 1–3:

1. The answer is not consistent with what has been said previously, holding some sort of contradiction, with either user or system previous utterances.
2. The history awareness is not clear, making it hard to determine if the answer is consistent or not. It can also be used in the cases where the system assumes to know a certain type of information which has not been tackled.
3. The sentence is consistent and in accordance with the whole dialogue history.

Overall Quality

Overall Quality is a less strict dimension of evaluation, aiming at gathering a more personal opinion regarding the system answer. It should translate into how overall satisfied is the user with the response, taking into account the scores for the other 4 metrics. The goal is also to give space for the annotators to differentiate between answers that have the same scores for the other metrics, but they have a preference for. This aspect is measured using a scale of 1–5, as it is the most appropriate span, according to Van Der Lee et al. (2019).

5.2 Annotation Process

The annotation process consisted of rating 6 possible responses to the same dialogue context, in the 5 dimensions of evaluation presented in Section 5.1.2. The different responses are provided by different systems, as evident in Figure 5.1: for each dialogue context, there are two possible responses from greedy decoding techniques, other two from sampling techniques, one response from a *'bad model'* — as the replies can be very similar in some situations, it is used to control the quality of the annotations — and the original ground-truth response. The order in which these answers appear is random throughout the several dialogues.

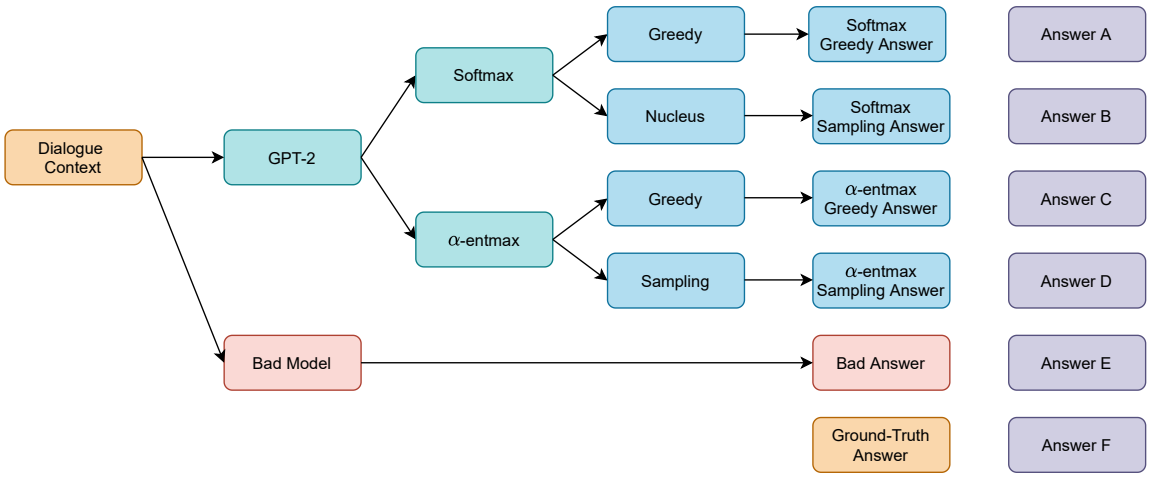


Figure 5.1: The possible answers provided by different systems.

The annotators were provided with a document containing guidelines, to ensure the annotation would be as homogeneous as possible. Besides an explanation of each annotation dimension, this document includes a set of

examples for each case. The annotation was done using a *Google Forms*, where the annotators would rate the 6 possible answers for a dialogue context in a certain dimension at the same time, allowing the comparison between them. The interface can be seen in Figure 5.2, where the annotator is asked to rate the different possible answers in the Grammaticality dimension.

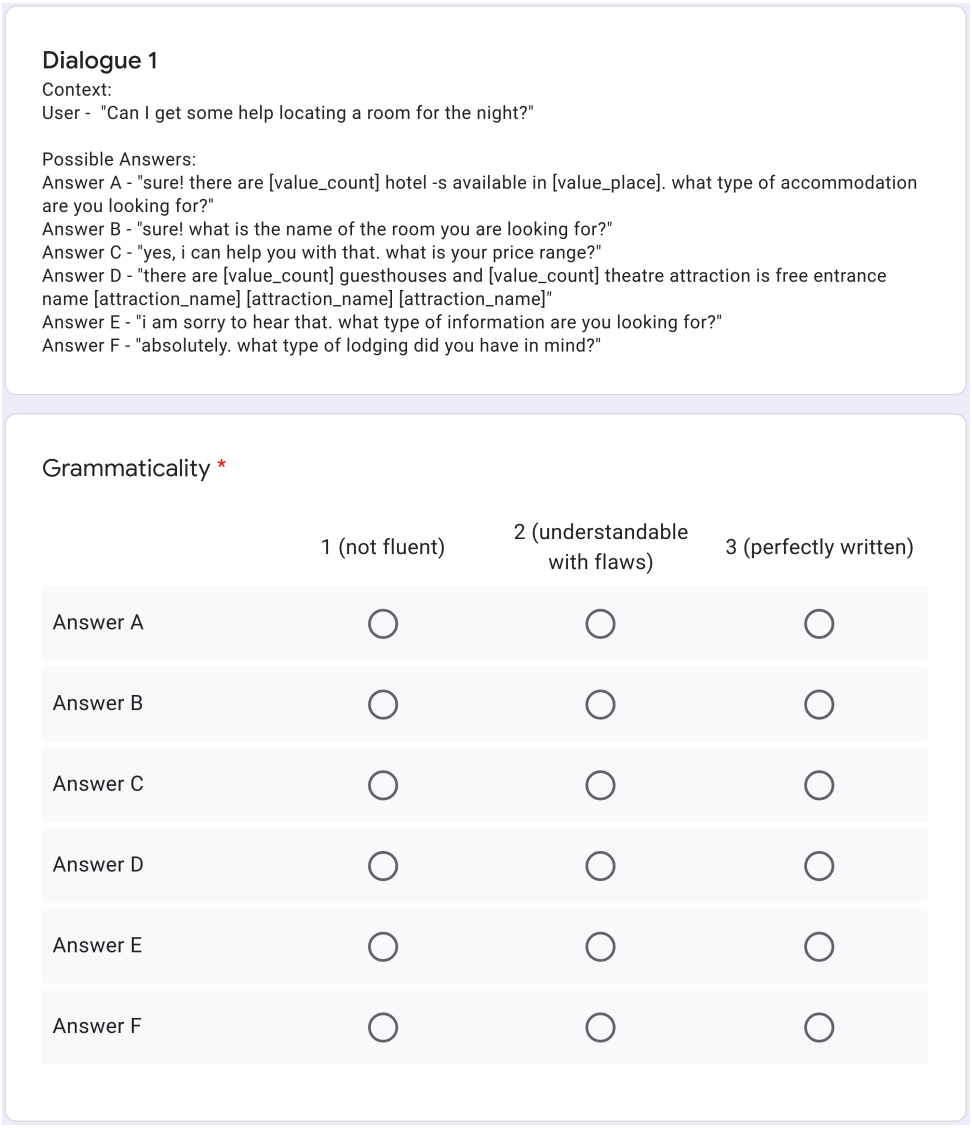


Figure 5.2: Annotation interface.

To maximize the available human resources, the evaluation was divided in two parts. The first one consisted of a mass evaluation of a small amount of dialogues, to gather feedback and validate the process. Hereafter, a second evaluation phase was conducted, with less available annotators evaluating a bigger amount of dialogues.

5.2.1 Preliminary Experience

In the preliminary annotation phase, 92 annotators, students from the *"Natural Language"* subject at Instituto Superior Técnico, evaluated the same 6 answers for 4 dialogues, resulting in 24 responses annotated in 5 different dimensions. Relevance and Consistency were initially proposed to be binary, and therefore evaluated between 0–1 in this first experience. Grammaticality and Informativeness were scored in a scale of 1–3 and Overall Quality from 1–5. The results were used to measure the agreement between annotations, to be presented in Section 5.3.2. After

gathering feedback from the preliminary evaluation experience, Relevance and Consistency evolved to a scale of 1–3, with the other domains remaining intact. More examples were also added to the guidelines, which can be found in Appendix A.

5.2.2 Extended Evaluation

In this experience, the group of annotators was composed of 9 people, from the age of 22 to 36, 5 female and 4 male. Among them, 4 are Master students of Computer Science, 1 has completed his PhD, 2 are PhD students of Computer Science and Natural Language and 2 have completed their Master's in Computer Science and Biomedical Engineering, currently working at Unbabel. Four of the participants had never annotated before. Each annotator was asked to score 6 answers for 4 dialogues, similarly to the preliminary experience. However, as the goal was to reach the maximum number of evaluated sentences, each annotator had an individual set of 3 dialogues, plus a common dialogue among all, to allow measuring the agreement. This dialogue was also present across the preliminary phase, serving as a control variable. In the extended evaluation phase, 168 possible responses were evaluated.

5.3 Results

In this section, we present the results for the human evaluation experience. Firstly, the annotation scores given to the several techniques are reported, followed by presenting the agreement among annotators. Finally, the correlation between annotations and metrics is calculated.

5.3.1 Annotations Scores

Table 5.1 comprises the average scores of each technique at each evaluation dimension, where the best results from the four generative models are highlighted. These results show that the annotators find in greedy techniques the most preferred responses, going into accordance with the information given by the automatic metrics.

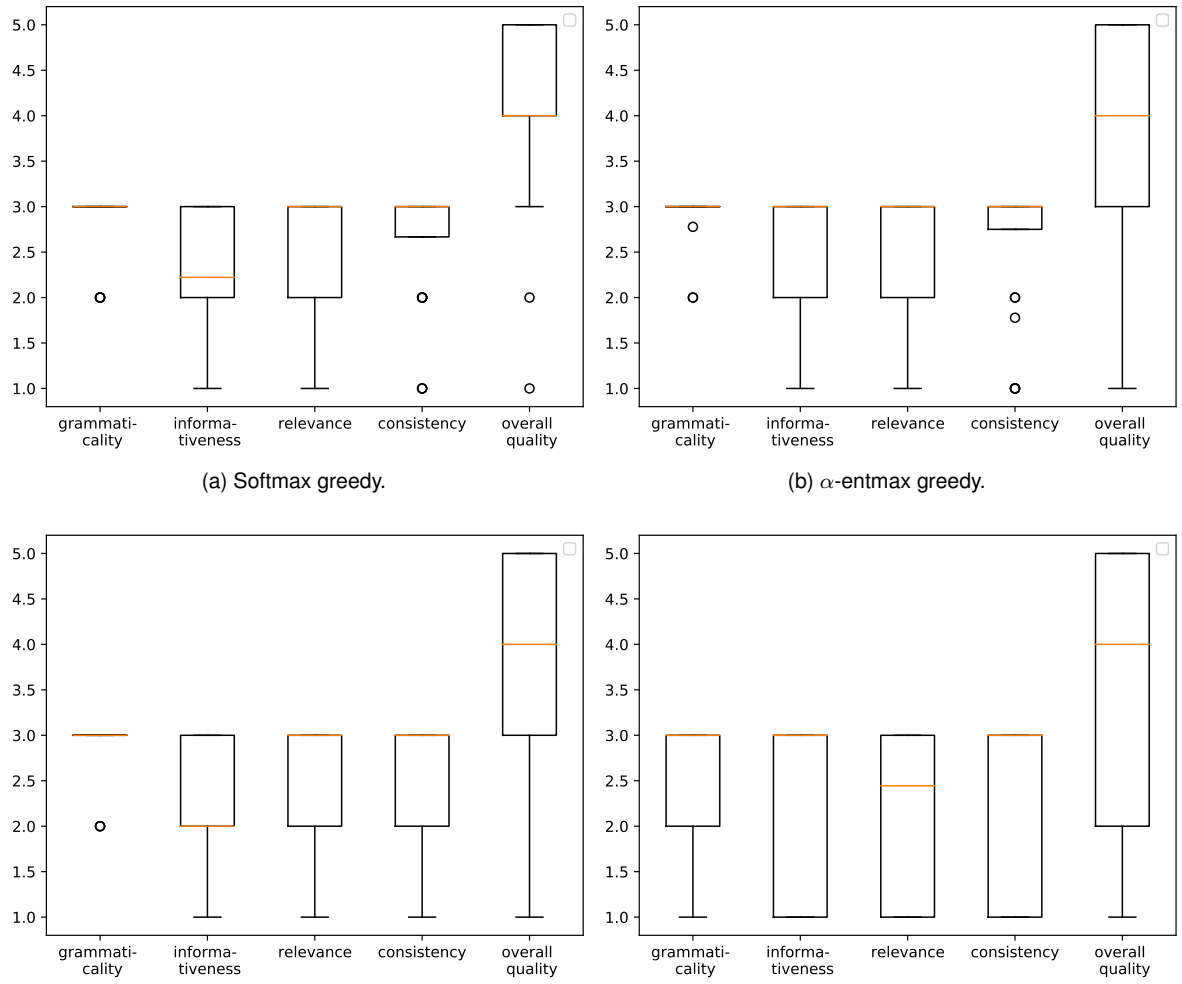
Table 5.1: Average scores at each evaluation domain.

	Grammaticality	Informativeness	Relevance	Consistency	Overall Quality
Softmax sampling	2.8929	2.1825	2.3929	2.4286	3.6548
α-entmax sampling	2.6746	2.2421	2.2103	2.2857	3.6746
Softmax greedy	2.8571	2.3016	**2.5992**	**2.6389**	**4.1270**
α-entmax greedy	**2.9206**	**2.4484**	2.5119	2.5992	3.9365
Bad	2.4206	1.8095	1.7698	1.9563	2.6468
Original	2.8929	2.2738	2.5278	2.6746	4.1310

For a more thorough evaluation of each decoding strategy, box plots are presented in Figure 5.3.

The greedy techniques present an overall better performance than the sampling techniques and a very similar performance between each other. However, α-entmax greedy (Figure 5.3b) outperforms softmax greedy (Figure 5.3a) in the Informativeness dimension. Besides, although softmax greedy presents a higher Consistency average value, α-entmax greedy results are more concentrated in the upper part, demonstrating the influence that some outliers can have in the mean value results. Nevertheless, the span of values for Overall Quality is more consistent in softmax greedy, suggesting a general better performance of this technique.

Regarding α-entmax sampling (Figure 5.3d), it shows the longest span of values across all the dimensions, suggesting to have the worst performance. However, in the Informativeness dimension, it presents a higher median value than softmax sampling (Figure 5.3c) and even softmax greedy, meaning that at least half of the responses were rated with the maximum score. From Figure 5.3d, it is possible to understand that, although α-entmax can produce good results, there is lack of regularity in its performance, generating sometimes replies which are grammaticaly incorrect, not informative, not relevant and not consistent with the conversation history.



(a) Softmax greedy.

(b) α-entmax greedy.

(c) Softmax sampling.

(d) α-entmax sampling.

Figure 5.3: Box plot of the annotation results.

5.3.2 Inter Annotator Agreement

Inter Annotator Agreement (IAA) is used to evaluate the reliability of the human evaluation task and its reproducibility, showing how uniform the annotations are. To allow the measurement between a group of multiple annotators, *Fleiss's Kappa* (Fleiss, 1971) is used. The values for *Kappa* can range between 0–1, with a score of 0 representing a random agreement, and 1 meaning there is total agreement among annotators. Contrasting with other agreement measures, *Fleiss's Kappa* does not treat all kinds of disagreement in the same manner and takes into account the natural order of the ratings, meaning that 4 should correlate more with 5 than with 1 (Bhowmick et al., 2008; Artstein and Poesio, 2008).

In Table 5.2, the values for IAA are presented, for the five evaluation dimensions. In the preliminary experience, the agreement can be calculated using all the 4 dialogues, while in the extended evaluation phase, it is only possible to use the common dialogue across them. For a fairer comparison between the two experiences, we also present the agreement calculated using only this common dialogue for the preliminary experience. Although we acknowledge that a single dialogue is far from being representative, the values for *Kappa* allow to have an idea about how trustworthy these experiences are.

Table 5.2: *Fleiss Kappa* for the two experiences.

	Preliminary — 4 dialogues	Preliminary — common dialogue	Extended — common dialogue
Grammaticality	0.2517	0.1565	0.2969
Informativeness	0.3221	0.3261	0.2901
Relevance	0.5205	0.5421	0.4904
Consistency	0.4305	0.4633	0.5857
Overall Quality	0.2225	0.2572	0.2945

For all the dimensions, there is either a fair agreement (0.2–0.4) or a moderate agreement (0.4–0.6), according to the interpretation scale proposed by Viera and Garrett (2005). The lowest values can suggest that there were some difficulties in annotating certain dimensions. According to Celikyilmaz et al. (2020), low agreement between annotators can also indicate that there are not significant differences in the possible answers, which indeed happened in most of the dialogues, where some of the responses were very similar.

From the preliminary to the extended evaluation phase, the agreement increased in Grammaticality, Consistency and Overall Quality, while it decreased in Informativeness and Relevance. This drop would be expected in the Relevance and Consistency dimensions, as the scale span was enlarged. In what comes to Informativeness, a possible explanation can be the difficulty to score it in some specific dialogues. As an example, we have the dialogue context of Figure 5.2, where there are no specific queries to fill in the question *"Can I get some help locating a room for the night?"*, making the Informativeness dimension more ambiguous.

5.3.3 Correlation with Automatic Metrics

To measure the correlation between the evolution of two variables, the correlation coefficient is usually calculated. Its values can range between -1.0, for a perfect negative correlation, and 1.0, showing a perfect positive correlation. A correlation of 0.0 means that there is no relationship between the movement of the two variables. The most popular ways to measure these values are the Pearson correlation, measuring the linear relationship, and the Spearman correlation, determining how well the two variables are correlated through a monotonic function. In this case, the two variables to correlate are each human annotation dimension and the metrics to evaluate generation, at the sentence level. Considering the essence of each metric, BLEU, BERTScore and METEOR are considered. Similarly to Mehri and Eskenazi (2020), both Pearson and Spearman coefficients are presented.

There are two main ways to evaluate the correlation between human annotations and automatic metrics in NLG (Ma et al., 2019). The first one is at the segment level (Table 5.3), where the average of scores for a single answer is correlated with the automatic metric for that specific answer. The amount of datapoints is therefore proportional to the number of evaluated dialogues. On the system level (Table 5.4), all the average scores for a domain of the same system are averaged, being that value used to calculate the correlation between the dimension and the metric. In the latter, the amount of datapoints is independent of the amount of annotated dialogues, being always 6 in this case, for each dimension.

At the segment level, Table 5.3 shows that all the metrics have low correlation with the domains evaluated by the annotators. The highest correlation value is between Overall Quality and BLEU, with a Spearman value of 0.3309, which can still be considered a low correlation value. Nevertheless, BERTScore and METEOR also present the highest correlation values for Overall Quality, which is a good indicator regarding the fidelity of the automatic metrics. Contrary to what would be expected, Grammaticality has very low correlation values, suggesting that the automatic metrics are not able to understand nuances in the language. For a certain dimension, the correlation values for different metrics are relatively close to each other, proposing a certain agreement among the three metrics.

Similarly to what happens in previous works with an identical approach, the values for correlation at system level are significantly higher than at segment level, as evident in Table 5.4.

Table 5.3: Correlation at the segment level.

	BLEU		BERTScore		METEOR	
	Pearson	Spearman	Pearson	Spearman	Pearson	Spearman
Grammaticality	0.1257	0.1757	0.1575	0.1615	0.1201	0.1182
Informativeness	0.0917	0.1798	0.0941	0.1333	0.0530	0.0766
Relevance	0.1673	0.2423	0.2173	0.2258	0.1789	0.2185
Consistency	0.2107	0.3076	0.2514	0.2560	0.2046	0.2512
Overall Quality	0.2360	**0.3309**	**0.2975**	0.2805	0.2341	**0.2679**

Table 5.4: Correlation at the system level.

	BLEU		BERTScore		METEOR	
	Pearson	Spearman	Pearson	Spearman	Pearson	Spearman
Grammaticality	0.3535	0.5508	0.4367	0.5508	0.1201	0.1182
Informativeness	0.1962	0.4857	0.2567	0.4857	0.2081	0.4857
Relevance	0.3765	0.8857	0.4593	0.8857	0.4149	0.8857
Consistency	0.5002	**0.9429**	0.5747	**0.9429**	0.5348	**0.9429**
Overall Quality	0.4497	0.8286	0.5177	0.8286	0.4792	0.8286

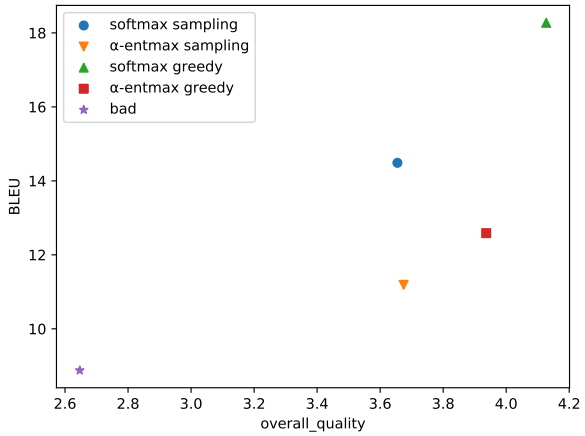
The first thing that stands out is the values for Spearman correlation, which are basically the same for all the metrics. This observation confirms that the three metrics are able to extract very similar information, despite being calculated in different manners. Hence, we can infer that no metric is superior to one other in terms of correlation with human judgement. Taking BERTScore as an example, although it is calculated using contextual embeddings and therefore able to understand some context, its behaviour is identical to BLEU, which simply uses similarity between word embeddings. This suggests that the sentence alone is not enough to grasp the whole meaning behind a reply in a goal-oriented dialogue.

There is a contrast between Pearson and Spearman values for correlation, with the latter holding much higher results. It indicates that, for close values of automatic metric scores, the slight difference between them is not enough to choose between the systems.

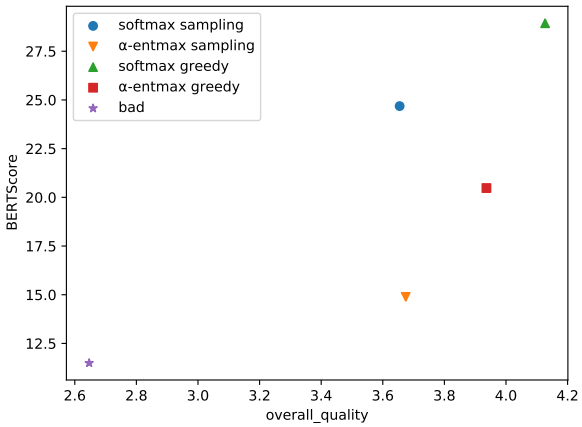
The high correlation values for Relevance, Consistency and Overall Quality indicate that the three automatic metrics can be useful to compare the performance of different systems, despite being less informative when evaluating a sentence alone. However, the highest correlation values are for Consistency, for the three metrics. This observation is unexpected, as this domain is probably the one with least information present in each sentence. It

can indicate that these correlation values have low fundament, representing a coincidence of high and low scores, and can not be considered representative. Many authors report how difficult it is to show correlation between human annotations and automatic metrics, such as the case of the WMT20 Metrics Shared Task, where no winner was found (Mathur et al., 2020).

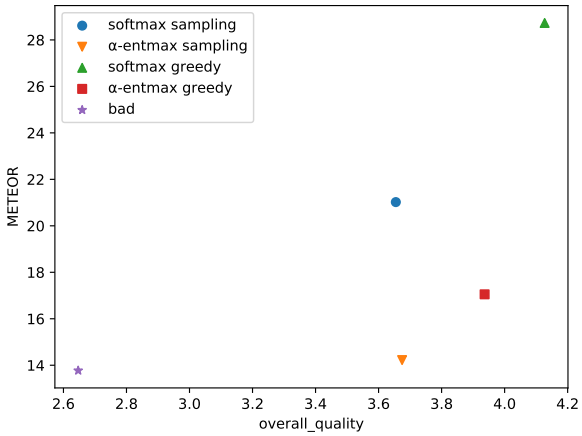
In Figure 5.4, it is possible to see the correlation between Overall Quality and the metrics, at the system level. The original responses are not included, because the automatic metrics obviously obtain the maximum score, damaging the readability of the graphics. There is slight correlation between scores among the generation techniques. Both the metrics and human judgement find in greedy sampling from softmax the highest generation quality, closely followed by greedy sampling from α-entmax. Nevertheless, automatic metrics can sometimes lead to poor conclusions, which is the case of sampling from softmax: although humans find it the worst quality technique among this set of examples, automatic metrics rate it as the second best one. Besides, despite their close scores in Overall Quality, softmax sampling and α-entmax sampling have distinct automatic metrics values, confirming the possibility of high quality answers with lower automatic metrics scores.



(a) BLEU vs Overall Quality.



(b) BERTScore vs Overall Quality.



(c) METEOR vs Overall Quality

Figure 5.4: Plot of automatic metrics vs Overall Quality at the system level.

To conclude, metrics such as BLEU, BERTScore and METEOR can give useful information in the comparison between different systems. In the context of answer generation in goal-oriented dialogues, these metrics are able to extract similar features, being in concordance with each other. However, as these metrics rely on the comparison with a database reference, they never recognize a better sentence than the original and techniques with more word

diversity are naturally damaged. The lack of history context awareness leads these metrics into the inability of grasping certain language nuances.

Chapter 6

Conclusions

This work added to the study of goal-oriented dialogue systems, by experimenting novel response generation strategies, as well as exploring the adopted evaluation techniques and how informative they are of the response quality. In this chapter, we sum up the achievements in Section 6.1 and present some interesting possible directions for future work in Section 6.2.

Many concepts developed in this work can be transferred to the aerospace industry, with the integration of modern data analysis techniques and Machine Learning in the optimization of several problems. The unprecedented ability to store and collect data will have a broad impact in the aerospace business, from process control and standardization in the factory, to the development of materials and composites for the aircraft; from testing, certification and anomaly detection to the previously mentioned human-machine interaction (Brunton et al., 2020).

Besides, a substantial field of application of Deep Learning techniques is Computer Vision, which is the base of modern Image Processing, with direct applications for drones and satellites. Although the preferred architectures for this technology are Convolutional Neural Networks, which have not been tackled in this thesis, recent works have been trying to use the powerful Transformers in these applications, opening some possible directions to discover (Dosovitskiy et al., 2020; Chen et al., 2020).

6.1 Achievements

In this work, we studied how different techniques can influence the quality of a generated automatic reply, in a goal-oriented setting. The main achievements lie in the application of sparse attention mechanisms to automatic goal-oriented response generation, making use of the α-entmax transformation. Although stochastic strategies were found to have their positive attributes, we conclude that, in this framework, the prime systems are those which employ greedy techniques, using either the standard softmax or α-entmax. Moreover, we confirmed that goal-oriented response generation benefits from having a more informative context, as it significantly improves the dialogue awareness. The thorough analysis of the systems' behaviour led to realizing that many aspects are not grasped by the chosen automatic evaluation metrics, motivating further investigation.

Futhermore, we successfully conducted a study to find correlations between the adopted automatic metrics and human perception of quality. A set of evaluation dimensions was developed, supported by some illustrative guidelines, allowing the collection of a significant amount of reliable human annotations. After a probabilistic analysis, we found that BLEU, METEOR and BERTScore substantially correlate with human judgement, being useful to roughly compare the performance of different systems. Nonetheless, these metrics are inappropriate to understand

nuances in cases where the systems show a similar performance, making it essential to resort to human evaluation for a more detailed comparison.

6.2 Future Work

This work has paved the way for other approaches in the goal-oriented dialogue systems field. The first path derives from the fact that most real dialog corpora do not have information regarding dialogue state, and labeling it requires a lot of human effort. Therefore, it would be ideal to have a system which could be trained without this information, as it would be much more easily transferable to other applications. One simple way to pursue this would be to follow Ham et al. (2020) and include the dialogue state in the generated sentence, incorporating a Dialogue State Tracker capability in the system. By doing this, the system would be able to perform in an interactive way, which would also lead to a fairer evaluation of the systems, as human language and behaviour may differ from the provided in the dataset (Takanobu et al., 2020). Nevertheless, we are positive that the results would not be very different, as more freedom in conducting the dialogues would probably just increase the discrepancy between the models.

Besides, although we made progress regarding which automatic metrics should be used in goal-oriented dialogue, if used alone, there are several missed aspects and nuances. Given the difficulty in understanding if two sentences have the same meaning in all relevant aspects, to find the most effective way of evaluating a system's performance remains an open problem. An interesting direction would be to build a neural evaluation model, which would learn to evaluate generated text based on scores given by humans. As evaluation problems are reported in different NLP fields, similar approaches start appearing, such as COMET (Rei et al., 2020), a neural evaluation tool for Machine Translation. Nonetheless, we recognize that collecting the necessary data for this purpose would be an extremely expensive process.

On a final note, we would like to emphasize some ethical issues that may appear from automatic dialogue design. Although useful and powerful, Language Models pre-trained on massive datasets simply consisting of text from the internet can lead to extremely biased models, prone to have unwanted behaviours. Although neural models require huge amounts of data to be successful, we underline the importance of revisiting dataset construction, in which the quality and fairness of information should be prioritized over quantity.

Bibliography

Gabor Angeli, Percy Liang, and Dan Klein. A Simple Domain-Independent Probabilistic Approach to Generation. In *Proc. of the Conference on Empirical Methods in Natural Language Processing*, 2010.

Ron Artstein and Massimo Poesio. Survey Article: Inter-Coder Agreement for Computational Linguistics. *Computational Linguistics*, 34:555–596, 2008.

Dzmitry Bahdanau, Kyung Hyun Cho, and Yoshua Bengio. Neural Machine Translation by Jointly Learning to Align and Translate. In *Proc. of the International Conference on Learning Representations*, 2015.

Satanjeev Banerjee and Alon Lavie. METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments. In *Proc. of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, 2005.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A Neural Probabilistic Language Model. *Journal of Machine Learning Research*, 3:1137–1155, 2003.

Plaban Kr. Bhowmick, Pabitra Mitra, and Anupam Basu. An Agreement Measure for Determining Inter-Annotator Reliability of Human Judgements on Affective Text. In *Proc. of the Workshop on Human Judgements in Computational Linguistics*, 2008.

Steven L. Brunton, J. Nathan Kutz, Krithika Manohar, Aleksandr Y. Aravkin, Kristi Morgansen, Jennifer Klemisch, Nicholas Goebel, James Buttrick, Jeffrey Poskin, Agnes Blom-Schieber, Thomas Hogan, and Darren McDonald. Data-Driven Aerospace Engineering: Reframing the Industry with Machine Learning. 2020. URL `https://arxiv.org/abs/2008.10740`.

Paweł Budzianowski and Ivan Vulić. Hello, It's GPT-2 - How Can I Help You? Towards the Use of Pretrained Language Models for Task-Oriented Dialogue Systems. In *Proc. of the Workshop on Neural Generation and Translation*, 2019.

Paweł Budzianowski, Tsung-hsien Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gašić. MultiWOZ - A Large-Scale Multi-Domain Wizard-of-Oz Dataset for Task-Oriented Dialogue Modelling. In *Proc. of the Conference on Empirical Methods in Natural Language Processing*, 2018.

Bill Byrne, Karthik Krishnamoorthi, Chinnadhurai Sankar, Arvind Neelakantan, Daniel Duckworth, Semih Yavuz, Ben Goodrich, Amit Dubey, Andy Cedilnik, and Kyu Young Kim. Taskmaster-1: Toward a Realistic and Diverse Dialog Dataset. In *Proc. of the Conference on Empirical Methods in Natural Language Processing*, 2019.

Asli Celikyilmaz, Elizabeth Clark, and Jianfeng Gao. Evaluation of Text Generation: A Survey. 2020. URL `https://arxiv.org/abs/2006.14799`.

Hongshen Chen, Xiaorui Liu, Dawei Yin, and Jiliang Tang. A Survey on Dialogue Systems: Recent Advances and New Frontiers. *ACM SIGKDD Explorations Newsletter*, 19:25–35, 2017.

Mark Chen, Alec Radford, Rewon Child, Jeff Wu, Heewoo Jun, Prafulla Dhariwal, and David Luan. Image GPT. *OpenAI Blog*, 2020. URL `https://openai.com/blog/image-gpt/`.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In *2014 Conference on Empirical Methods in Natural Language Processing*, 2014.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context. In *Proc. of the Annual Meeting of the Association for Computational Linguistics*, 2019.

Jan Deriu, Alvaro Rodrigo, Arantxa Otegi, Guillermo Echegoyen, Sophie Rosset, Eneko Agirre, and Mark Cieliebak. Survey on Evaluation Methods for Dialogue Systems. *Artificial Intelligence Review*, 2020.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proc. of the Conference of the North American Chapter of the Association for Computational Linguistics*, 2019.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. 2020. URL `http://arxiv.org/abs/2010.11929`.

Jeffrey L. Elman. Finding Structure in Time. *Cognitive Science*, 14:179–211, 1990.

Mihail Eric, Rahul Goel, Shachi Paul, Adarsh Kumar, Abhishek Sethi, Peter Ku, Anuj Kumar Goyal, Sanchit Agarwal, Shuyang Gao, and Dilek Hakkani-Tur. MultiWOZ 2.1: A Consolidated Multi-Domain Dialogue Dataset with State Corrections and State Tracking Baselines. *ArXiv*, 2019. URL `http://arxiv.org/abs/1907.01669`.

Angela Fan, Mike Lewis, and Yann Dauphin. Hierarchical Neural Story Generation. In *Proc. of the Annual Meeting of the Association for Computational Linguistics*, 2018.

Sarah E. Finch and Jinho D. Choi. Towards Unified Dialogue System Evaluation: A Comprehensive Analysis of Current Evaluation Protocols. In *Proc. of the Annual Meeting of the Special Interest Group on Discourse and Dialogue*, 2020.

Joseph L. Fleiss. Measuring Nominal Scale Agreement Among Many Raters. *Psychological Bulletin*, 76:378–382, 1971.

Rob Gaizauskas, James Law, and Emma Barker. Investigating Spoken Dialogue to Support Manufacturing Processes. Technical report, University of Sheffield, 2018.

Jianfeng Gao, Michel Galley, and Lihong Li. Neural Approaches to Conversational AI. In *Proc. of the Annual Meeting of the Association for Computational Linguistics*, 2018.

Sergey Golovanov, Rauf Kurbanov, Sergey Nikolenko, Kyryl Truskovskyi, Alexander Tselousov, and Thomas Wolf. Large-Scale Transfer Learning for Natural Language Generation. In *Proc. of the Annual Meeting of the Association for Computational Linguistics*, 2019.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. Deep Learning. *MIT Note*, 2016. URL `https://www.deeplearningbook.org/`.

Alex Graves, Abdel Rahman Mohamed, and Geoffrey Hinton. Speech Recognition with Deep Recurrent Neural Networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013.

Donghoon Ham, Jeong-gwan Lee, Youngsoo Jang, and Kee-eung Kim. End-to-End Neural Pipeline for Goal-Oriented Dialogue System using GPT-2. In *Proc. of the Annual Meeting of the Association for Computational Linguistics*, 2020.

Tatsunori B Hashimoto, Hugh Zhang, and Percy Liang. Unifying Human and Statistical Evaluation for Natural Language Generation. In *Proc. of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2019.

Matthew Henderson, Ivan Vulić, Daniela Gerz, Iñigo Casanueva, Paweł Budzianowski, Sam Coope, Georgios Spithourakis, Tsung-Hsien Wen, Nikola Mrkšić, and Pei-Hao Su. Training Neural Response Selection for Task-Oriented Dialogue Systems. In *Proc. of the Annual Meeting of the Association for Computational Linguistics*, 2019.

Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9:1735–1780, 1997.

Ari Holtzman, Jan Buys, Leo Du, Maxwell Forbes, and Yejin Choi. The Curious Case of Neural Text Degeneration. In *Proc. of the International Conference on Learning Representations*, 2020.

Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer Feedforward Networks are Universal Approximators. *Neural Networks*, 2:359–366, 1989.

Minlie Huang, Xiaoyan Zhu, and Jianfeng Gao. Challenges in Building Intelligent Open-Domain Dialog Systems. *ACM Transactions on Information Systems*, 38, 2020.

Daniel Jurafsky and James Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition*. 3rd edition, 2019.

Anjuli Kannan, Karol Kurach, Sujith Ravi, Tobias Kaufmann, Andrew Tomkins, Balint Miklos, Greg Corrado, László Lukács, Marina Ganea, Peter Young, and Vivek Ramavajjala. Smart Reply: Automated Response Suggestion for Email. In *Proc. of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016.

Ravi Kondadadi, Blake Howald, and Frank Schilder. A Statistical NLG Framework for Aggregated Planning and Realization. In *Proc. of the Annual Meeting of the Association for Computational Linguistics*, 2013.

Oliver Lemon, Kallirroi Georgila, James Henderson, and Matthew Stuttle. An ISU Dialogue System Exhibiting Reinforcement Learning of Dialogue Policies: Generic Slot-Filling in the TALK In-car System. In *Proc. of the Conference of the European Chapter of the Association for Computational Linguistics*, 2006.

Chia Wei Liu, Ryan Lowe, Iulian V. Serban, Michael Noseworthy, Laurent Charlin, and Joelle Pineau. How NOT To Evaluate Your Dialogue System: An Empirical Study of Unsupervised Evaluation Metrics for Dialogue Response Generation. In *Proc. of the Conference on Empirical Methods in Natural Language Processing*, 2016.

Qingsong Ma, Johnny Wei, Ondřej Bojar, and Yvette Graham. Results of the WMT19 Metrics Shared Task: Segment-Level and Strong MT Systems Pose Big Challenges. In *Proc. of the Conference on Machine Translation*, 2019.

Andre F.T. Martins and Ramon F. Astudillo. From Softmax to Sparsemax: A Sparse Model of Attention and Multi-Label Classification. In *Proc. of the International Conference on Machine Learning*, 2016.

Pedro Henrique Martins, Zita Marinho, and André F. T. Martins. Sparse Text Generation. In *Proc. of the Conference on Empirical Methods in Natural Language Processing*, 2020.

Nitika Mathur, Johnny Tian-Zheng Wei, Markus Freitag, Qingsong Ma, and Ondrej Bojar. Results of the WMT20 Metrics Shared Task. In *Proc. of the Conference on Machine Translation*, 2020.

Shikib Mehri and Maxine Eskenazi. Unsupervised Evaluation of Interactive Dialog with DialoGPT. In *Proc. of the Annual Meeting of the Special Interest Group on Discourse and Dialogue*, 2020.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed Representations of Words and Phrases and their Compositionality. In *Proc. of the International Conference on Neural Information Processing Systems*, 2013.

Sinno Jialin Pan and Qiang Yang. A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering*, 22:1345–1359, 2010.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zu. BLEU: A Method for Automatic Evaluation of Machine Translation. In *Proc. of the Annual Meeting on Association for Computational Linguistics*, 2002.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the Difficulty of Training Recurrent Neural Networks. In *Proc. of the International Conference on Machine Learning*, 2013.

Baolin Peng, Chenguang Zhu, Chunyuan Li, Xiujun Li, Jinchao Li, Michael Zeng, and Jianfeng Gao. Few-shot Natural Language Generation for Task-Oriented Dialog. 2020. URL `https://arxiv.org/abs/2002.12328`.

Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global Vectors for Word Representation. In *Proc. of the Conference on Empirical Methods in Natural Language Processing*, 2014.

Ben Peters, Vlad Niculae, and André F. T. Martins. Sparse Sequence-to-Sequence Models. In *Proc. of the Annual Meeting of the Association for Computational Linguistics*, 2019.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep Contextualized Word Representations. In *Proc. of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2018.

Matt Post. A Call for Clarity in Reporting BLEU Scores. In *Proc. of the Conference on Machine Translation*, 2018.

Lisong Qiu, Juntao Li, Wei Bi, Dongyan Zhao, and Rui Yan. Are Training Samples Correlated? Learning to Generate Dialogue Responses with Multiple References. In *Proc. of the Annual Meeting of the Association for Computational Linguistics*, 2019.

Alec Radford and Tim Salimans. Improving Language Understanding by Generative Pre-Training. *OpenAI Blog*, 2018. URL `https://openai.com/blog/language-unsupervised/`.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language Models are Unsupervised Multitask Learners. *OpenAI Blog*, 2019. URL `https://openai.com/blog/better-language-models/`.

Ricardo Rei, Craig Stewart, Ana C Farinha, and Alon Lavie. COMET: A Neural Framework for MT Evaluation. In *Proc. of the Conference on Empirical Methods in Natural Language Processing*, 2020.

Sebastian Ruder. *Neural Transfer Learning for Natural Language Processing*. PhD thesis, 2019.

Jost Schatzmann and Steve Young. The Hidden Agenda User Simulation Model. *IEEE Transactions on Audio, Speech and Language Processing*, 17, 2009.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural Machine Translation of Rare Words with Subword Units. *Proc. of the Annual Meeting of the Association for Computational Linguistics*, 2016.

Iulian V. Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. Building End-To-End Dialogue Systems Using Generative Hierarchical Neural Network Models. In *Proc. of the AAAI Conference on Artificial Intelligence*, 2016.

Louis Shao, Stephan Gouws, Denny Britz, Anna Goldie, Brian Strope, and Ray Kurzweil. Generating High-Quality and Informative Conversation Responses with Sequence-to-Sequence models. In *Proc. of the Conference on Empirical Methods in Natural Language Processing*, 2017.

Shikhar Sharma, Layla El Asri, Hannes Schulz, and Jeremie Zumer. Relevance of Unsupervised Metrics in Task-Oriented Dialogue for Evaluating Natural Language Generation. 2017. URL `http://arxiv.org/abs/1706.09799`.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to Sequence Learning with Neural Networks. In *Proc. of the International Conference on Neural Information Processing Systems*, 2014.

Ryuichi Takanobu, Qi Zhu, Jinchao Li, Baolin Peng, Jianfeng Gao, and Minlie Huang. Is Your Goal-Oriented Dialog Model Performing Really Well? Empirical Analysis of System-wise Evaluation. In *Proc. of the Annual Meeting of the Special Interest Group on Discourse and Dialogue*, 2020.

Chris Van Der Lee, Albert Gatt, Emiel van Miltenburg, Sander Wubben, and Emiel Krahmer. Best Practices for the Human Evaluation of Automatically Generated Text. In *Proc. of the International Conference on Natural Language Generation*, 2019.

Ashish Vaswani, Google Brain, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. *Proc. of the Conference on Neural Information Processing Systems*, 2017.

Anthony J Viera and Joanne M Garrett. Understanding Interobserver Agreement: the Kappa Statistic. *Family medicine*, 37:360–363, 2005.

Ashwin K Vijayakumar, Michael Cogswell, Ramprasath R. Selvaraju, Qing Sun, Stefan Lee, David Crandall, and Dhruv Batra. Diverse Beam Search: Decoding Diverse Solutions from Neural Sequence Models. 2016. URL `http://arxiv.org/abs/1610.02424`.

Oriol Vinyals and Quoc Le. A Neural Conversational Model. In *Proc. of the International Conference on Machine Learning*, 2015.

Zhuoran Wang and Oliver Lemon. A Simple and Generic Belief Tracking Mechanism for the Dialog State Tracking Challenge: On the believability of observed information. In *Proc. of the Annual Meeting of the Special Interest Group on Discourse and Dialogue*, 2013.

Joseph Weizenbaum. ELIZA - A Computer Program For the Study of Natural Language Communication Between Man And Machine. *Communications of the ACM*, 9, 1966.

Tsung Hsien Wen, Milica Gašić, Nikola Mrkšić, Pei Hao Su, David Vandyke, and Steve Young. Semantically Conditioned LSTM-based Natural Language Generation for Spoken Dialogue Systems. In *Proc. of the Conference on Empirical Methods in Natural Language Processing*, 2015.

Tsung Hsien Wen, David Vandyke, Nikola Mrkšíc, Milica Gašíc, Lina M. Rojas-Barahona, Pei Hao Su, Stefan Ultes, and Steve Young. A Network-based End-to-End Trainable Task-oriented Dialogue System. In *Proc. of the Conference of the European Chapter of the Association for Computational Linguistics*, 2017.

Thomas Wolf, Victor Sanh, Julien Chaumond, and Clement Delangue. TransferTransfo: A Transfer Learning Approach for Neural Network Based Conversational Agents. 2019. URL `http://arxiv.org/abs/1901.08149`.

Chien-Sheng Wu, Andrea Madotto, Ehsan Hosseini-Asl, Caiming Xiong, Richard Socher, and Pascale Fung. Transferable Multi-Domain State Generator for Task-Oriented Dialogue Systems. In *Proc. of the Annual Meeting of the Association for Computational Linguistics*, 2019.

Steve Young, Milica Gasi c, Blaise Thomson, and Jason D Williams. POMDP-based Statistical Spoken Dialogue Systems: a Review. In *Proc. of the IEEE*, 2013.

Xiaoxue Zang, Abhinav Rastogi, and Jindong Chen. MultiWOZ 2.2: A Dialogue Dataset with Additional Annotation Corrections and State Tracking Baselines. 2020. URL `https://arxiv.org/abs/2007.12720`.

Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. Personalizing Dialogue Agents: I have a dog, do you have pets too? In *Proc. of the Annual Meeting of the Association for Computational Linguistics*, 2018.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. BERTScore: Evaluating Text Generation with BERT. In *Proc. of the International Conference on Learning Representations*, 2020a.

Zheng Zhang, Ryuichi Takanobu, Qi Zhu, Minlie Huang, and Xiaoyan Zhu. Recent Advances and Challenges in Task-oriented Dialog System. *Science China Technological Sciences*, 63, 2020b.

Tiancheng Zhao and Maxine Eskenazi. Towards End-to-End Learning for Dialog State Tracking and Management using Deep Reinforcement Learning. In *Proc. of the Annual Meeting of the Special Interest Group on Discourse and Dialogue*, 2016.

Tiancheng Zhao, Ran Zhao, and Maxine Eskenazi. Learning Discourse-level Diversity for Neural Dialog Models using Conditional Variational Autoencoders. In *Proc. of the Annual Meeting of the Association for Computational Linguistics*, 2017.

Qingfu Zhu, Lei Cui, Weinan Zhang, Furu Wei, and Ting Liu. Retrieval-Enhanced Adversarial Training for Neural Response Generation. In *Proc. of the Annual Meeting ofthe Association for Computational Linguistics*, 2019.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning Books and Movies: Towards Story-like Visual Explanations by Watching Movies and Reading Books. In *Proc. of the IEEE International Conference on Computer Vision*, 2015.

Appendix A

Annotation Guidelines

Annotations Instructions

You will be given a dialogue between a tourist and an assistant from an **information center in the city of Cambridge**, trying to reach a goal, such as booking a night in a hotel, a restaurant reservation or a train trip. At some point, the dialogue will have several possible answers from the assistant. **Your task is to evaluate those answers in several metrics, taking into account what was said previously in the dialogue**. Different answers for the same context should be evaluated independently, as the goal is **not** to order the answers, but to reach an absolute score for each one of them. Nevertheless, you are encouraged to compare the answers, in order for the marks to be the fairest possible.

The answers may seem a bit strange at the beginning, as the system originates answers where specific entities were taken out and replaced for more general tags. This means that, for example, instead of *"Hilton"* the system will suggest *[hotel_name]*. You are not expected to have any knowledge of the database, so the only thing you need to evaluate is if this broad entity makes sense in the context.

The response for one metric should not be influenced by the other metrics. For example, even if the answer is not informative, it should not influence the consistency or relevance.

Make sure you read these instructions carefully and feel free to take a look at them while you're evaluating the dialogues!

Grammaticality (1-3)

Evaluates how fluent and correctly constructed is the system answer. Grammatical incorrections which should **not** be considered: apostrophe absence in abbreviations ("*it s"* instead of *"it's"*); spaces between compound words, such as adverbs (*"moderate -ly"* instead of *"moderately"*) or plural words (*"restaurant -s"* instead of *"restaurants"*); non capitalization of letters at the beginning of sentences.

1. A score of **(1/3)** means the answer is not fluent at all, being poorly structured, even if you're able to understand the meaning by the words it's composed of.

   Example 1: The sentence doesn't make any sense.

User	I am visiting Cambridge and looking for local attractions. Can you suggest a place to go in town?
System	**[value_attraction] st road suggest [value_price] to go.**

   Example 2: Although it is possible to imagine what the system is trying to say, the sentence doesn't make any sense.

User	Hi, could you find me a train for Friday? I need to leave from Leicester.
System	I can assist you with that. What time Friday would you like to leave at?
User	I would like to leave after 12:15.
System	**do you have a lot day time and price range train at?**

2. A score of **(2/3)** means that the grammar has some minor flaws, but the sentence is understandable.

   Example: The sentence is not perfectly constructed (the verb conjugation should be 'do' instead of 'did'), but it is totally understandable.

User	I am visiting Cambridge and looking for local attractions. Can you suggest a place to go in town?
System	**absolutely! did you want the [attraction_name] or [attraction_name]? which venue did you want?**

3. A score **(3/3)** means that you are in the presence of a fluent, grammatically perfect answer.

   Example: The sentence doesn't have any flaws in the way it is constructed.

User	I am visiting Cambridge and looking for local attractions. Can you suggest a place to go in town?

System	**sure, there are many attractions in town. do you have a specific type of attraction in mind?**

Informativeness (1-3)

Evaluates whether the answer brings relevant information to the table. For more complex dialogues, one can measure *informativeness* as the amount of user queries tackled by the system. In simpler cases, it can be seen as how helpful is the system in moving towards the goal of the conversation.

1. A score of **(1/3)** means the answer is not informative, it doesn't add anything to the conversation.

   Example 1: The system doesn't reply to any of the user queries, not informing the address nor the phone number. It instead asks if the user has a preference on the restaurants, when it was clear that he wanted a suggestion at the point of the conversation.

User	Are there any italian restaurant -s in the city centre?
System	There are quite a few italian restaurant -s in the centre. would you like something cheap, moderate -ly priced, or more expensive?
User	I would like something in the cheap range and i need the address and phone number.
System	**do you have a preference of the restaurant -s?**

   Example 2: The system does not provide relevant information for helping the user find a room.

User	Can you help me find the funky fun house?
System	**what [attraction_name] do you like?**

2. A score of **(2/3)** means the answer is a bit informative: It helps, but not completely. In this situation, you would like to get a reply containing more information.

Example 1: The system doesn't reply to all the queries, giving a suggestion of the restaurant and its address, but missing the phone number.

User	Are there any italian restaurant -s in the city centre?
System	There are quite a few italian restaurant -s in the centre. would you like something cheap, moderate -ly priced, or more expensive?
User	I would like something in the cheap range and i need the address and phone number.
System	**[restaurant_name] is in the [value_area] of town and the address is [restaurant_address].**

Example 2: The system moves towards the goal, but could have given a more informative reply.

User	Can you help me find the funky fun house?
System	**it is in [value_area].**

3. A score of **(3/3)** shows an informative answer, providing all the required information

   Example 1: The system helps to reach the goal, moving forward in the conversation and giving all the required information.

User	Are there any italian restaurant -s in the city centre?
System	There are quite a few italian restaurant -s in the centre. Would you like something cheap, moderate -ly priced, or more expensive?
User	I would like something in the cheap range and i need the address and phone number.
System	**[restaurant_name] is located at [restaurant_address] and the phone number is [restaurant_phone].**

   Example 2: The system helps to reach the goal, narrowing down the available options considering the user's preferences.

User	Can you help me find the funky fun house?

System	**it is located in the [value_area] of town on [attraction_address], [attraction_postcode]. would you like more information?**

Relevance (1-3)

Evaluate if the system response is appropriate given the user query. Does the reply make sense to what was asked?

1. A score of **(1/3)** means the response doesn't make sense given the user utterance.

   Example 1: The system asks for a detail which is not relevant for a hotel reservation.

User	Yes, please make my reservation at the Alexander Bed and Breakfast.
System	**what time would you like to arrive?**

   Example 2: The system misunderstands the meaning of the user's utterance, confirming the entity is a hotel.

User	Is that a guesthouse?
System	**yes it is a hotel.**

2. A score of **(2/3)** means the answer makes sense given what the user said, but there were more relevant subjects to tackle or replies to be given.

   Example 1: The system asks for a train and, although the time of arrival may be important, asking for the destination or departure site is more relevant.

User	I am looking for a train on friday.
System	**what time would you like to arrive?**

3. A score of **(3/3)** means the answer makes sense given what the user said.

   Example 1: The system asks for relevant details in the domain.

User	Yes, please make my reservation at the Alexander Bed and Breakfast.
System	**what day would you like to book on and for how long?**

Example 2: The system understands the meaning of the user's utterance.

User	Is that a guesthouse?
System	**no sorry that is a hotel.**

Example 3: The system asks for relevant details in the domain. In this case, although the most relevant answer would be the destination of the train, as this is an information center of Cambridge, the destination is implicit. In this situation, we assume that the departure time [value_time] goes according to the user's restrictions, as we do not have information showing the contrary.

User	Hi, could you find me a train for Friday? I need to leave from Leicester.
System	I can assist you with that. What time Friday would you like to leave at?
User	I would like to leave after 12:15.
System	**the [train_id] train leaves [value_place] at [value_time] and arrives in [value_place] at [value_time]. would you like to book this train?**

Consistency (1-3)

To evaluate if the response does not contradict previous information, either given by the user or the system. It can also be seen as how aware is the system of the conversation history?

1. A score of **(1/3)** means the answer is not consistent with what has been said previously; it is a contradiction.

   Example 1: The system asks something that was already clarified

User	I am also looking for a place to stay. It needs to include free wifi and parking, be three star, and in the same area as the restaurant.
System	**I have several guesthouses that fit that criteria. Is there a particular area you would like to stay in?**

Example 2: The system gives a reply which is not coherent with the conversation history.

User	Can you find a restaurant that serves gastropub food and is on the east side?
System	There is one recommendation and that is the Royal Standard. Would you like me to provide the address and phone number?
User	Yes I would like you to provide me the address and phone number of that restaurant.
System	Certainly, the address is 290 Mill Road City Centre and the phone number is 01223247877.
User	Can I book a reservation for 7 at 13:00 this Friday?
System	**i am sorry but we cannot find an [value_pricerange] restaurant with a [value_food] menu.**

2. A score of **(2/3)** means the answer is consistent with what has been said previously, but the history awareness is not clear, making it hard to determine if the reply is consistent or not. It can also be used in the cases where the system assumes to know a certain type of information which has not been tackled.

Example 1: The system makes a booking without having the knowledge of the user's destination, supposing a value for that slot.

User	I need a taxi. I'll be departing from la raza.
System	I can help you with that. When do you need to leave?
User	I'd like to leave after 11:45 please.
System	**okay, a [taxi_type] will pick you up at [value_time]. the contact number for the driver is [taxi_phone]. can i help you with anything else?**

3. A score of **(3/3)** means the answer is consistent with what has been said previously and in accordance with the whole dialogue history.

Example 1: The system asks for details which have not been tackled yet.

User	I am also looking for a place to stay. It needs to include free wifi and parking, be three star, and in the same area as the restaurant.
System	**there are [value_count] hotel -s that meet your criteria. do you have a price range in mind?**

Example 2: The system gives a suggestion which seems to be matching the given criteria.

User	I am also looking for a place to stay. It needs to include free wifi and parking, be three star, and in the same area as the restaurant.
System	**the [hotel_name] is a [value_count] star hotel in the [value_area] area. it is in the [value_pricerange] price range, and has free parking and wifi.**

Overall Quality (1-5)

A more personal evaluation of the overall quality of the answer. Given the context, as a user, how satisfied would you be with the given answer? You should take into account the previous evaluations you gave for that reply.