ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Department of Computer Science and Engineering
Artificial Intelligence

MULTI-DOMAIN CONVERSATIONAL AGENTS BASED ON SEMANTIC PARSING AND RETRIEVAL OF EXTERNAL KNOWLEDGE

Master Thesis in
Text Mining

Supervisor
Prof. Gianluca Moro

Co-supervisor
Dott. Giacomo Frisoni

Candidate
Nicola Poggialini

Third Session
Academic Year 2022 – 2023

KEY WORDS

Transformers

Conversational Agents

Retrieval-Enhanced Language Models

Abstract Meaning Representation

Graph Neural Networks

Introduction

Conversational agents have gained more and more attention in recent years thanks to their ability to communicate using natural language: this results in systems that can not only engage in dialogue but are also able to complete tasks for the users.
With Transformer-based language models becoming the go-to ingredient to build state-of-the-art methods, they have also been adopted in this field. However, these architectures store all knowledge within their weights, leading to two primary limitations. First of all, significant resources are needed for both training and subsequently storing the models. Then, these methods also lack the ability to remain up-to-date: when new knowledge is being published, the only way to let the models know about it is by a new training process. To address this problem, approaches like RETRO [1] propose the integration of a knowledge retrieval component. Conversational agents could potentially benefit from an analog approach: it could lead to systems that are lighter and also updated in real-time. Moreover, checking the external sources used to produce the responses could lead to more explainable agents.
Another problem that can limit an accurate generation of responses is the ambiguity of language: because of it, the ability of models to extract precise meaning can be hindered by using raw texts. Utilizing alternative data structures, such as Abstract Meaning Representation (AMR) graphs, can mitigate this challenge since they are specifically designed to capture the entities within a sentence and the semantic relationships connecting them.
Motivated by these premises, this thesis investigates the potential benefits of integrating both a retrieval component and AMRs within a dialogue system. As far as our comprehensive literature review reveals, this combined method

has not yet been explored. In particular, our approach uses an existing conversational agent model chosen from the literature as the starting point of our work. Since it already presents a module that retrieves knowledge from an external source, we propose integrating AMR graphs into it, in the attempt to improve the selection of relevant documents for response generation. Ablation studies will then be conducted to identify the optimal configuration of our architecture. Finally, we will compare the performances of our proposal against the chosen baselines. We will also showcase examples of input prompts and the correspondent responses generated by our model.
This thesis is divided into the following chapters:

- **Chapter 1** - Introduction of the main theoretical aspects behind this work;
- **Chapter 2** - Overview of the various technologies proposed in the literature that can be used for the problem posed;
- **Chapter 3** - Detailed description of the reference architecture and of the techniques introduced in the attempt to improve it;
- **Chapter 4** - Results of ablation studies and final performance;
- **Chapter 5** - Conclusions, ethical considerations and future work.

Index

List of Figures

Chapter 1

Background

This chapter provides a generic description of the theoretical aspects behind the thesis. It will start with a brief introduction of conversational agents, then we'll proceed with covering other themes such as retrieval-augmented text generation, AMRs, and graph embedding.

1.1 Conversational Agents

Conversational agents are artificial intelligence systems capable of holding a natural conversation with users. They can be built in different ways, from the first techniques involving simple template matching to the more recent deep neural networks.
Following the taxonomy proposed in [5], we can recognize two main categories of conversational agents: chatbots and task-oriented agents. Chatbots can engage in casual conversations with the users, which typically involve general, everyday topics. The focus here is more on the interaction between the system and the human, rather than the content of the conversation itself. On the other hand, task-oriented agents are able to reach a certain goal by giving precise answers to complex queries, which may involve not only text generation but also retrieving information from online or offline databases.
However, as the authors state, there are also other aspects that can be used to define different types of conversational agents, such as the type of input/output

supported, or the possible domains of applicability.
Another important characteristic to mention is interpretability. As in other artificial intelligence systems, also in this case we can recognize black boxes and interpretable agents: with respect to the first ones, interpretable systems will be able to provide an explanation for a given response, which is crucial in certain contexts.

1.2 Retrieval-Augmented Text Generation

Retrieval-augmented text generation is a text generation paradigm that combines deep learning and traditional retrieval techniques. With respect to traditional generation methods, this new paradigm has some remarkable advantages.
First of all, instead of encoding the knowledge within the parameters, it can be accessed explicitly and instantly from external sources: this leads to models that require less storage and training time. Another advantage is that the systems can use human-written texts as a starting point in the generation, instead of producing the output from scratch.
Moreover, this type of generation doesn't suffer from the problem of new knowledge being published: as already said, in non-retrieval methods, the knowledge is all stored in the weights, so if new content is published, the model can't know about it. Instead, by adding the retrieval, the model can access the new content without having to perform the training again.
To formulate this paradigm, we can follow the work in [6], where the standard generation task is defined as a mapping from input sequence x to output sequence $y = f(x)$: for instance, x could be the dialogue context and y the corresponding generated response. Instead, the retrieval-augmented generation can be formulated as $y = f(x, z)$ where $z = (x_r, y_r)$ is a set of relevant instances extracted from the original training set or external sources: in fact, the main idea of this paradigm is that y_r may help in generating the response if x_r (or y_r) is similar (or relevant) to the input x (x_r none when unsupervised data are used).
There are different possible types of sources for the retrieval data:

- Training corpus: in this case, there is no information that is not already seen during training. However, the idea is that these are still additional references, which could improve the generation process.
- External data: the retrieval pool is different with respect to the training corpus, which could provide access to new and unseen information.
- Unsupervised data: the main advantage of it is that no annotated data are needed.

There are also different methods when it comes to the retrieval metric:

- Space-vector retrieval: in this class, we have methods that measure the similarity using techniques such as TF-IDF. However, they may exhibit limitations in retrieving instances that are only semantically relevant.
- Dense-vector retrieval: they overcome the problem of space-vector retrieval by using pre-trained language models, which encode the texts to low-dimensional vectors. The similarity score is then computed with inner products between the encodings.
- Task-specific retrieval: the methods just seen are based on the idea that the more x_r is similar to x, the more likely x_r and y_r will help in generating the response. However, the closest instances by universal text similarity do not guarantee to provide the best performances in downstream tasks. For these reasons, with this type of retrieval, the retriever and its downstream generation model are unified and trained with the same objective.

One final aspect that is important to mention is the explainability. Without the use of retrieval, motivating why a system has given a certain response is nothing but trivial. Again, all the knowledge is contained in the weights of the model, and it is not easy to extract the reasoning behind a certain generation. On the other hand, in retrieval-augmented agents, retrieved texts can be used to give an explanation for the response produced since not only they were considered relevant by the system with respect to the given input, but they are also used as a reference to generate the new turn of a dialogue.

1.3 Graph Embedding

Graph embedding is a technique that produces a low-dimensional vector representing the whole input graph or a part of it. First of all, let's define the graph data structure and the different types of graphs (different from the already known directed, undirected, cyclic, etc. graphs).
Following the notation proposed in [2], we can indicate with $G = (V, E)$ a graph, where $v \in V$ is a node and $e \in E$ is an edge. G is associated to a node type mapping function $f_v : V \rightarrow T_v$ and an edge type mapping function $f_e : E \rightarrow T_e$: T_v denotes the set of node type, while T_e denotes the set of edge types.
Now we can define three types of graphs:

- Homogeneous graph: all nodes belong to a single type and all edges belong to a single type, i.e. $|T_v| = |T_e| = 1$.

- Heterogeneous graph: nodes and/or edges belong to at least two types, i.e. $|T_v| > 1$ and/or $|Te| > 1$. A particular type of heterogeneous graph is the knowledge graph, a directed graph whose nodes are entities and each edge indicates a relation r from head entity h to tail entity t (denoted as <h,r,t>); entities and relations are usually of different types.

- Graph with auxiliary information: this type contains auxiliary information of node/edge/whole graph. Labels, attributes (discrete or continuous), and node features are possible examples.

Let's now define the output of a graph embedding algorithm. Unlike the input, the output is task-driven, so the type of embedding depends on the application. We can recognize four different types of graph embeddings:

- Node embedding: this is used for node-level tasks, which are concerned with predicting the identity or role of each node within the graph.

- Edge embedding: in edge-level tasks, the focus is on predicting a certain property for each edge.

- Hybrid embedding: this is a combination of different types of graph components.

- Whole-graph embedding: used for graph-level tasks, where a property of the whole graph is what we want to predict.

Given an embedding dimension d, the output is a d-dimensional vector(s) that preserves the properties of the graph. These properties can be quantified using proximity measures:

- First order proximity (FOP): the pairwise similarity of nodes connected by an edge and it is expressed by the weight of the edge. For each node i, we can create the vector $s_i^{(1)}$, where the values are the FOP between node i and the other nodes in the graph.

- Second order proximity (SOP): it compares the similarity of the nodes' neighborhood structures. To find the SOP, we can apply a similarity function between the $s^{(1)}$ vectors of two nodes.

- Higher order proximity: the $k - th$ order proximity between two nodes is the similarity between the $k - 1$ similarity vectors.

There are different types of graph embedding techniques, as we can see from the taxonomy proposed in [2] and reported in Figure 1.1.

The class of methods that we're going to focus on is Deep Learning-based graph embedding, in which we can recognize two categories: one for the techniques that use random walk, and the other for those that don't use random walk.
When using random walk, a graph is represented as a set of random walk paths sampled from it. Deep Learning methods are then applied to the sampled paths, in order to create an embedding of the graph that preserves the properties carried by the paths.
On the other hand, in methods that don't use random walks, deep models are applied directly to the graphs. There are two main possibilities to carry out this technique: the first is using autoencoders, in which an encoder maps input data to a representation space and the decoder maps the representation space

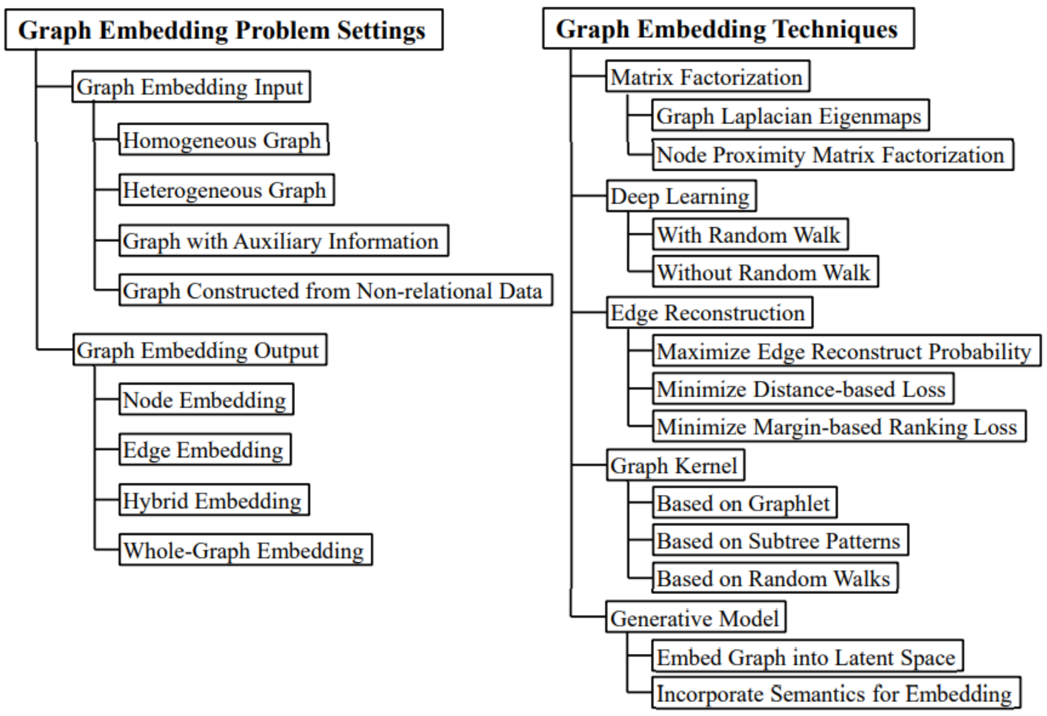

Graph Embedding Problem Settings
Graph Embedding Input
Homogeneous Graph
Heterogeneous Graph
Graph with Auxiliary Information
Graph Constructed from Non-relational Data
Graph Embedding Output
Node Embedding
Edge Embedding
Hybrid Embedding
Whole-Graph Embedding
Graph Embedding Techniques
Matrix Factorization
Graph Laplacian Eigenmaps
Node Proximity Matrix Factorization
Deep Learning
With Random Walk
Without Random Walk
Edge Reconstruction
Maximize Edge Reconstruct Probability
Minimize Distance-based Loss
Minimize Margin-based Ranking Loss
Graph Kernel
Based on Graphlet
Based on Subtree Patterns
Based on Random Walks
Generative Model
Embed Graph into Latent Space
Incorporate Semantics for Embedding

Figure 1.1: Graph Embedding taxonomy proposed in [2]

to a reconstruction space; the other is using deep neural networks, a category of methods to which Graph Neural Networks (GNNs) belong.
The following section will be dedicated to GNNs, which is a popular graph embedding technique and is also used in this work.

1.4 Graph Neural Networks

GNNs are characterized by Message Passing layers, through which the features interact along the graph. In these layers, neighboring nodes or edges exchange information and influence each other's updated embeddings. In particular, if we consider just the nodes, message passing works this way:

- current node embeddings of neighbor nodes are gathered;
- the collected information is aggregated to get a new embedding;
- the node state is updated using the new embedding.

This approach is known as Graph Convolution, and it can be seen as an extension of convolution to graph data. It can be seen as a general schema that can be used to define different models. There isn't just one possibility to perform the aggregate and update steps: for example, some possible operations are the mean, max, or a neural network. By changing these configurations, we can obtain different GNNs (these are discussed in Section 2.3).

1.5 Abstract Meaning Rapresentation

Abstract Meaning Representation (AMR) [3] is a semantic structure formalism that represents the meaning of a sentence in a rooted, labeled, directed, acyclic graph.
The purpose of these graphs is to abstract away the meaning of a sentence. Language can be ambiguous and the same concepts can be expressed in different forms: this potentially creates difficulties when processing text. AMRs can solve this issue, because, ideally, sentences that have the same meaning are assigned the same AMR. For example, the following sentences are all assigned the same AMR:

- "He described her as a genius";
- "His description of her: genius";
- "She was a genius, according to his description".

AMRs can be represented by using a graph notation or the PENMAN notation [7] (the PENMAN notation can be defined as a "serialization format for the directed, rooted graphs used to encode semantic dependencies"[1]). In Figure 1.2 and in Figure 1.3 is reported an example for these representations, that correspond both to the sentence "The boy wants to go".

The authors use nodes for entities, events, properties, and states. Leaves then are used for concepts: for example, "(b / boy)" represents an instance "b" of the concept boy. Edges connecting entities represent relations, so "(d /

[1] `https://penman.readthedocs.io/en/latest/notation.html#kas1989`

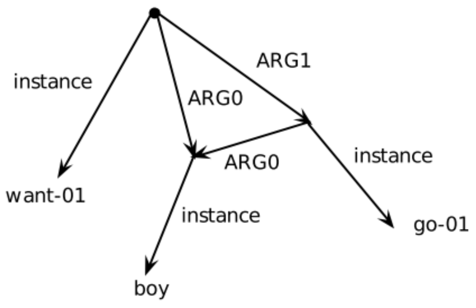
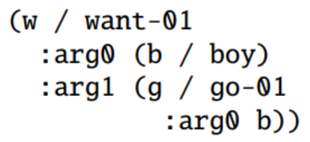


Figure 1.2: AMR using the graph notation (image from [3]).



```
(w / want-01
  :arg0 (b / boy)
  :arg1 (g / go-01
          :arg0 b))
```

Figure 1.3: AMR using the PENMAN notation (image from [3]).

die-01 :location (p / park))" means there was a death (d) in the park (p). AMR concepts are either English words ("boy"), PropBank framesets ("want-01"), or special keywords: PropBank framesets provide a structured representation of the semantic arguments of verbs and are used to annotate a large corpus of text, called the PropBank corpus [8]; on the other hand, "keywords include special entity types ("date-entity", "world-region", etc.), quantities ("monetary-quantity", "distance-quantity", etc.), and logical conjunctions ("and", etc)".
AMR uses approximately 100 relations and, for all of them, also their inverse: some examples are general semantic relations (e.g. :beneficiary, :cause, :location), relations for quantities, and relations for date-entities. The whole set of concepts and relations proposed is able to represent every sentence, taking all words into account.
The authors also state that AMR has some limitations. First of all, it doesn't represent word inflections due to tense and number. Moreover, it omits articles and it doesn't have any universal quantifiers. Finally, there is no distinction between real events and hypothetical, future, or imagined ones. For example, in sentences "the boy wants to go", "want-01" and "go-01" have the same status, regardless of the fact that "go-01" will happen or not.

Chapter 2

Related Work

2.1 Retrieval-Augmented Text Generation

2.1.1 Dialogue Systems

An example of this paradigm applied to conversation agents can be found in EKo-DoC [9], an end-to-end conversational agent composed by three components: dense passage retrieval (DPR), re-ranking, and response generation.
In its DPR, the topic document (i.e. the document that serves as the background for the dialogue) and the dialogue are encoded with a Sentence Transformer [10] in order to obtain a query vector. This is then used to query the external knowledge base, which is composed of a set of snippets and the correspondent frozen dense vectors (obtained with the same encoding model). The score associated with each snippet is obtained through the dot product between its encoding and the query encoding: the snippets associated with the best K_1 scores are passed to the re-ranker (with $K_1 < |K_{KB}|$, where K_{KB} denotes the external knowledge base).
The re-ranking module is composed of a distilled version of RoBERTa [11] (obtained with the same training process proposed in [12]) and a feed-forward neural network, which will produce a score for each retrieved snippet: the snippets will then be sorted according to this new score and only the first $K_2 < K_1$ will be selected and passed to the generation part.
The final component will produce the response using the encoder-decoder model

LongT5 [13], which receives the K_2 snippets and the dialogue context.

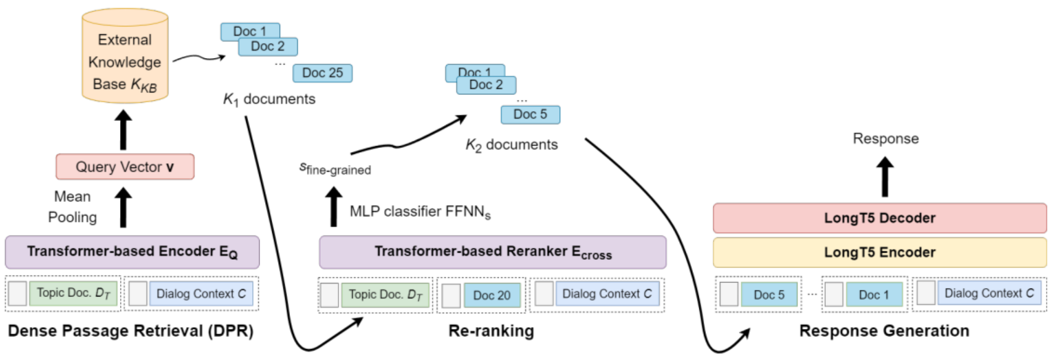


Figure 2.1: Illustration of EKo-DoC.

2.1.2 Other Tasks

There are also other tasks that benefit from retrieval-augmented neural networks.

Language modeling is one of them and RETRO [1] is a significant example of it: this model is able to perform as well as GPT-3 [14] despite being 4% its size.

RETRO's retrieval part is based on a key-value database: each key is a BERT [15] embedding, while each value is composed by the text used to compute the key and its continuation. The model works by encoding each input text with BERT and using the resulting vector to query the database. The neighbors are retrieved using an approximate search (ScaNN [16]): they are then fed, together with the input text, to the language model, which is an encoder-decoder architecture based on Transformer [17].

Another example of a language model that benefits from retrieval is SPALM [18], in which the authors combine short-term (with Transformer-XL [19]) and long-term memory (with retrieval from a key-value database) to predict the next token.

Also in question answering retrieval has been exploited, as we can observe in MedQA [20], a model for multiple choice QA. In this, Elasticsearch[1] is used to

[1]https://www.elastic.co/

retrieve N passages for each question-answer option pair: the retrieved texts and the pair are then encoded to select the most likely answer.

2.2 Text-to-AMR

Text-to-AMR is a task where the objective is converting a given input text into its corresponding AMR graph. In this work, we're going to use SPRING, introduced in [21]. The authors propose two models that share the same architecture: one is trained to perform Text-to-AMR and the other for AMR-to-Text. In particular, the models are based on a seq2seq encoder-decoder structure, which relies on BART [22] and its transfer learning capabilities. They extend BART's vocabulary with AMR concepts, frames, and relations and, in order to use graphs within this architecture, they use linearization techniques to transform this data structure into a sequence of symbols.

2.3 Graph Neural Networks - Relevant Variants

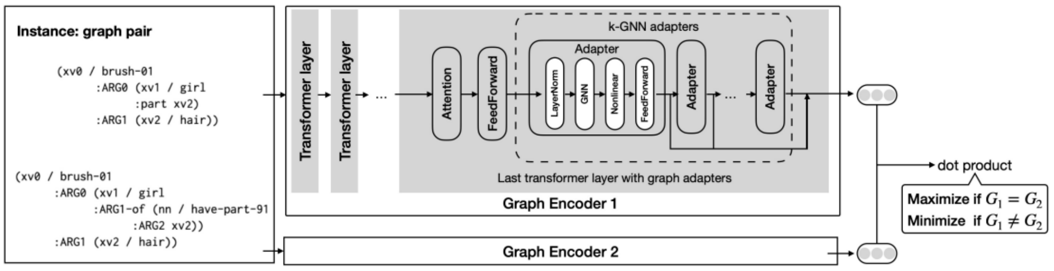
In the Background section, it was mentioned that we can obtain different variants of GNNs by changing the general formulation. One of the most prominent examples is Graph Convolutional Networks (GCNs) [23], which perform a sum of normalized neighbor embeddings. In particular, they are characterized by self-loops and scaling factors: self-loops are used to include the node itself in the aggregation; on the other hand, scaling factors are defined by the inverse of the degree of the interested nodes in the aggregation, and their purpose is to keep the magnitude of nodes with lots of neighbors under control. Another important variant is Graph Attention Networks (GATs) [24]. They introduce an attention mechanism in the computation: this is responsible for learning an attention coefficient which states how much the features of a node are relevant for another node.
There also variants of GATs: relevant examples are EGAT [25] and GATv2 [26].
Edge-Featured Graph Attention Network (EGAT) is based on the idea that edge features should be taken into account in computing attention weights.

Following this, the authors define EGAT layers, which contain both a node attention and an edge attention block.
The other model, GATv2, relies on improving the attention computed in GATs, which is defined as 'static' by the authors. With this definition the authors mean that, for any query node, the attention function is monotonic with respect to the neighbor key scores: this is shown by the fact that the ranking (the *argsort*) of attention coefficients is shared across all nodes in the graph, and is unconditioned on the query node. This problem is overcome by changing the order of internal operations done in GAT: this way the authors define what they call 'dynamic' attention in GATv2.

2.4 AMR Metrics

The goal of an AMR metric is to evaluate the similarities of pairs of AMR graphs. Some examples of AMR metrics can be found in the literature, such as SMATCH [27] and S^{2}MATCH [28]. SMATCH tries to find a one-to-one matching of variables that maximizes the number of exact matches of triples. The alignment process, an NP-hard problem, limits the efficiency of SMATCH, especially with the increasing of AMRs' size (alignment in AMR graphs usually refers to the process of aligning nodes or subgraphs in two or more AMR representations that correspond to similar or equivalent meanings).
S^{2}MATCH, in order to produce a better alignment, is based on a soft semantic match instead of matching only identical triples, but it doesn't solve the computational problems seen in SMATCH.
Another work worth noticing can be found in [29], in which AMRs are seen as high-dimensional objects. The authors propose to first propagate node embeddings iteratively, then to employ the Wasserstein Weisfeiler Leman kernel ($WWLK$) to calculate the minimum cost of transforming one graph to another. They also introduce an extension that considers edge embeddings (called $WWLK_\theta$, which improve the performances but require more effort in the data collection process, since they use a supervised learning method.
A recent work tries to overcome the problems seen in the presented metrics. In [30], the authors present AMRSim, an alignment-free method that computes

the cosine similarity of the embeddings extracted from the graphs. In particular, these encodings are obtained using a model based on BERT, in which graph neural networks are incorporated into transformer layers.
Moreover, the authors propose a self-supervised learning method, which alleviates the task of collecting data for training the model.



```
    (xv0 / brush-01
          :ARG0 (xv1 / girl
                :part xv2)
          :ARG1 (xv2 / hair))

(xv0 / brush-01
      :ARG0 (xv1 / girl
            :ARG1-of (nn / have-part-91
                  :ARG2 xv2))
      :ARG1 (xv2 / hair))
```

Figure 2.2: Illustration of AMRSim architecture.

2.5 Graph-Augmented Dialogue Systems

In [31], the authors propose a method that exploits AMRs and external knowledge for response generation. The model is based on three modules:

- Semantic graph construction: the external knowledge source is processed to build a graph that represents it. In particular, SPRING is first used to create sentence-level AMR graphs: these are then passed to a co-reference resolution system to build the document-level graph. One thing to notice is that in this work the external knowledge is not retrieved but it is provided for each instance along with the dialogue.

- Knowledge selection: in this part, the objective is to identify relevant knowledge snippets that can be used to produce an appropriate and informative response for each turn. The knowledge selection module takes in input the dialogue context and the produced document semantic graph.

- Response generation: the selected snippets nodes and the dialogue context are given as input to GPT2 [32] to generate the response.

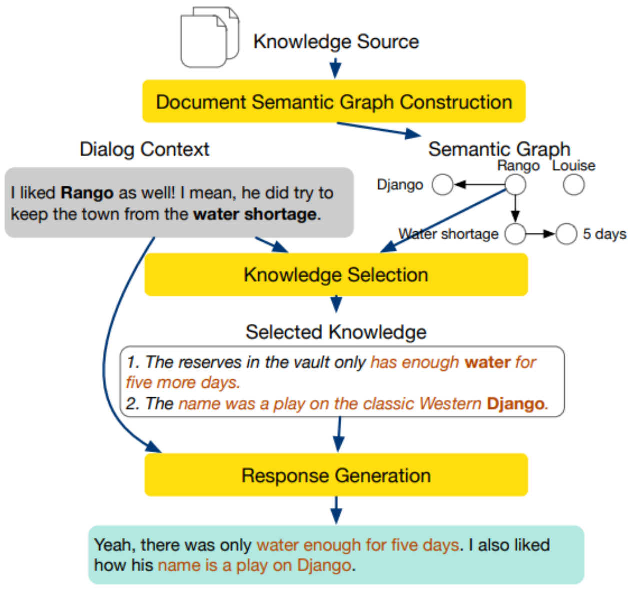

Knowledge Source
Document Semantic Graph Construction
Dialog Context
I liked **Rango** as well! I mean, he did try to keep the town from the **water shortage**.
Semantic Graph
Rango
Louise
Django
Water shortage
5 days
Knowledge Selection
Selected Knowledge
1. The reserves in the vault only has enough water for five more days.
2. The name was a play on the classic Western Django.
Response Generation
Yeah, there was only water enough for five days. I also liked how his name is a play on Django.

Figure 2.3: Illustration of the graph-augmented dialogue system described in Section 2.5.

Chapter 3

Method

This thesis aims to explore whether an architecture based on both external knowledge retrieval and AMR graphs can have a beneficial impact in generating a response to a given dialogue. To the best of my knowledge, this is something that has not yet been tried in the literature. As presented in Section 2, there have been successful attempts to incorporate retrieval [9] or AMR graphs [31] into conversational agents, but not both in the same model.
This is certainly an interesting research path given the good properties that retrieval and AMR graphs could bring, as already explained in Section 1: retrieval can provide a reference for generation and reduce the size of a model; AMRs can help in extracting the semantic meaning from a text and reducing the ambiguity of its phrases.
This chapter explains the method used in this work. We will start with a detailed description of the reference architecture and the reasons why it was chosen among other models from the literature. Then, it will proceed by focusing on the innovations proposed and by listing the experiments done to arrive at the final configuration of the conversational agent.

3.1 Reference Architecture

The architecture chosen as a starting point is EKo-DoC, described in Section 2.1.1. The motivation behind this choice was the fact that this is the only model

found in the literature that was designed for a conversational task and that uses external knowledge in a retrieval fashion. This is an important aspect that puts EKo-DoC in a better position for our purposes with respect to a model such as that described in Section 2.5. Even though it uses AMRs and is built to generate responses to a dialogue, this last approach does not incorporate retrieval. In fact, for each instance, the external knowledge is already provided and the relevant sentences are predicted from it through a knowledge selection task. Therefore, if we substitute this with a retrieval module that selects snippets from an external source (or from a knowledge base that comprehends all the sentences provided), we can't compare the two architectures, since they are performing different tasks.
On the other hand, EKo-DoC already uses retrieval but presents an architecture that allows inserting AMR processing inside it. In the next section, more details about EKo-DoC will be presented, in particular the ones regarding the input preparation, the training process, and the loss calculation used by this model.

3.1.1 Input Text Preparation

This work follows the same method proposed in the original model to manipulate the strings and pass them as input to the encoders.
First of all, for the external knowledge base, each snippet D_j is composed of a title and textual content. is composed of a title and textual content. The string passed to the encoder looks like this:

$$T[D_j] = [s]Title/Content[/s] \tag{3.1}$$

where "/" is used to separate the title from the content, "T([.])" is a function that maps a general object (a knowledge snippet in this case) into a textual sequence, "[s]" and "[/s]" are, respectively, the beginning-of-sequence (BOS) and the end-of-sequence (EOS) tokens ("[s]" and "[/s]" is a general notation that will be used also in the next cases: in practice, they are replaced with the special tokens belonging to each encoding model).
For each instance of the dataset, a query is composed using the topic document D_T and the dialogue context (C):

$$[s]T[D_T][sep]T[C][/s] \tag{3.2}$$

where $T[D_T]$ is defined in an analog way of the one seen for the knowledge snippet, so using the title and content of the topic document; on the other hand, $T[C]$ is a concatenation of all the dialogue turns separated by the $[sep]$ token (again, this will be replaced by the separation token used by the tokenizer of each encoder). One thing to underline is that the topic document is not a concept that can be found in the dataset but instead is defined by the authors of EKo-DoC: in particular, it is the first paragraph of the document used as a reference in the first turn of each dialogue.
After the retrieval module, each D_j of the K_1 snippets is passed to the re-rank encoder in this way:

$$[s]T[D_T][sep]T[D_j][sep]T[C][/s] \tag{3.3}$$

Finally, the set $\{\tilde{D}_1, \tilde{D}_2, ..., \tilde{D}_{K_2}\}$ of selected snippets is passed as a single string to the generator, along with the dialogue context:

$$[s]T[\tilde{D}_1 \oplus \tilde{D}_2 \oplus ... \oplus \tilde{D}_{K_2}][sep]T[C][/s] \tag{3.4}$$

where $T[\tilde{D}_1 \oplus \tilde{D}_2 \oplus ... \oplus \tilde{D}_{K_2}]$ is the concatenation of all selected snippets:

$$T[\tilde{D}_1][sep]T[\tilde{D}_2][sep]...[sep]T[\tilde{D}_{K_2}] \tag{3.5}$$

This method of passing all the inputs together in one single string is called Fusion-in-Input (FiI) by the authors, and it goes countercurrent with respect to the popular Fusion-in-Decoder (FiD) [33][34][35], where all the inputs are encoded separately.

3.1.2 EKo-DoC Training

The authors of EKo-DoC state that their model can be trained in an end-to-end fashion. In order to do so, simply using the generator's loss function is insufficient, because only the parameters of LongT5 will be updated: therefore, they propose incorporating loss components directly related to the retriever and re-ranker tasks. Their solution involves using the attention scores over the chosen snippets as pseudo-labels. In particular, these scores will be obtained with the decoder of LongT5: the intuition is that if the decoder pays more attention to a particular snippet, it means that it is likely to be relevant. Since computing the loss will be done in the same way in this work, we will report the formulation.

For each knowledge snippet D_j retrieved by the re-ranker, we will compute the average of all the pre-attention scores over all the tokens of the snippet's text. In Equation 3.6, it is reported the formula for this:

$$A(D_j) = \frac{\sum_{l=1}^{L} \sum_{h=1}^{H} \sum_{x=s_j}^{e_j} \sum_{y=1}^{|\widehat{Y}|} p_{l,h}(x,y)}{L \times H \times |s_j - e_j + 1| \times |\widehat{Y}|} \tag{3.6}$$

where s_j and e_j are the starting and ending tokens of the snippet's text, respectively, $\widehat{Y}$ is the ground-truth response, $p_{l,h}(x,y)$ is the pre-attention score between the x-th token of the snippet's text and the y-th token of $\widehat{Y}$ computed by the h-th attention head of the l-th layer.

The scores computed in this way are then normalized using the softmax function.

$$P_{attention}(D_j) = \frac{exp(A(D_j))}{\sum_{i=1}^{K_2} exp(A(D_i)} \tag{3.7}$$

The same normalization is also done for the DPR (s_c in the equation) and re-ranking scores (s_f):

$$Q_{coarse}(D_j) = \frac{exp(s_c(D_j, C, D_T))}{\sum_{i=1}^{K_2} exp(s_c(D_i, C, D_T)} \tag{3.8}$$

$$Q_{fine}(D_j) = \frac{exp(s_f(D_j, C, D_T))}{\sum_{i=1}^{K_2} exp(s_f(D_i, C, D_T)} \tag{3.9}$$

where D_T is the topic document and C the dialogue context.

The loss for the DPR and re-ranker are then computed using the Kullback–Leibler (KL) divergence, while the loss for the generator is the negative log-likelihood function. Everything is reported in the following equations:

$$L_{dpr}(B) = \sum_{i=1}^{|B|} D_{KL}(P^i_{attention} || Q^i_{coarse}) \tag{3.10}$$

$$L_{reranker}(B) = \sum_{i=1}^{|B|} D_{KL}(P^{i}_{attention}||Q^{i}_{fine}) \tag{3.11}$$

$$L_{retrieval}(B) = L_{dpr}(B) + L_{reranker}(B) \tag{3.12}$$

$$L_{generation}(B) = -\sum_{i=1}^{|B|} logP(\widehat{Y}^{i}|X^{i}) \tag{3.13}$$

$$L_{final}(B) = \frac{1}{|B|}(L_{generation}(B) + L_{retrieval}(B)) \tag{3.14}$$

where B is the mini-batch of instances and X^{i} is the i-th input.

3.2 AMR Embedding

In Section 2.3 some popular models were presented to embed graphs into numerical vectors capable of representing them. Since this work uses AMRs, the focus was put on methods designed specifically for this form of graphs.
In particular, even if designed originally for measuring the similarity between AMRs, the techniques presented in [29] and in [30] both introduce a new way to embed these graphs: therefore, since the results in both works are convincing, it could be beneficial extracting the encoding technique from them instead of using a more generic graph embedding method.
In Table 3.1 are reported the results of the variants of $WWLK$ and a couple of variants of $AMRSim$ in the STSB [36] and SICK [37] datasets: each one of these is composed of a set of pairs of sentences with a human-labeled similarity score. In the table, $AMRSim_{k}$ indicates that the graph encoder uses k-GNN [38]. Unlike standard GNNs that only consider immediate neighbors of a node, k-GNNs take into account the higher-order structure of the graph by considering multiple levels of neighbors around a node. These networks are used in AMRSim in order to make transformer layers able to capture information from AMR graphs.

As we can observe, $AMRSim$ performs considerably better than $WWLK$, which is the reason why its embedding strategy was the one chosen to embed

Method	**STSB**	**SICK**
$WWLK$	63.15	65.58
$WWLK_\theta$	66.94	67.64
$AMRSim_4$	70.88	73.10
$AMRSim_5$	70.94	72.64

Table 3.1: Pearson correlation on STSB and SICK test sets (data from [30])

AMRs.
In this work, the practice to encode a portion of text using AMRs is the following:

- First, the text is split into sentences since one AMR is able to represent just one sentence;
- Then, each sentence is converted into its AMR using the model SPRING (introduced in Section 2.2);
- Each AMR is encoded using the pre-trained embedding technique extracted from AMRSim;
- To obtain one single vector that represents the whole text, a pooling operation is performed across all sentence-level embeddings.

In Figure 3.1, an illustration of the embedding process of a text is reported.

3.3 AMR-Augmented Module

This section will describe where and how AMRs have been inserted into the original architecture and the reasoning behind it.
The first module that, in theory, could be augmented using these graphs is the retrieval one. In this case, the workflow will be extracting the AMRs from each passage in the external knowledge base and then embedding them as explained in the previous section: by doing so, we obtain a one-dimensional vector for each passage. Now, to query these embeddings during the forward step, it is necessary to repeat these passages with the instances from the dataset,

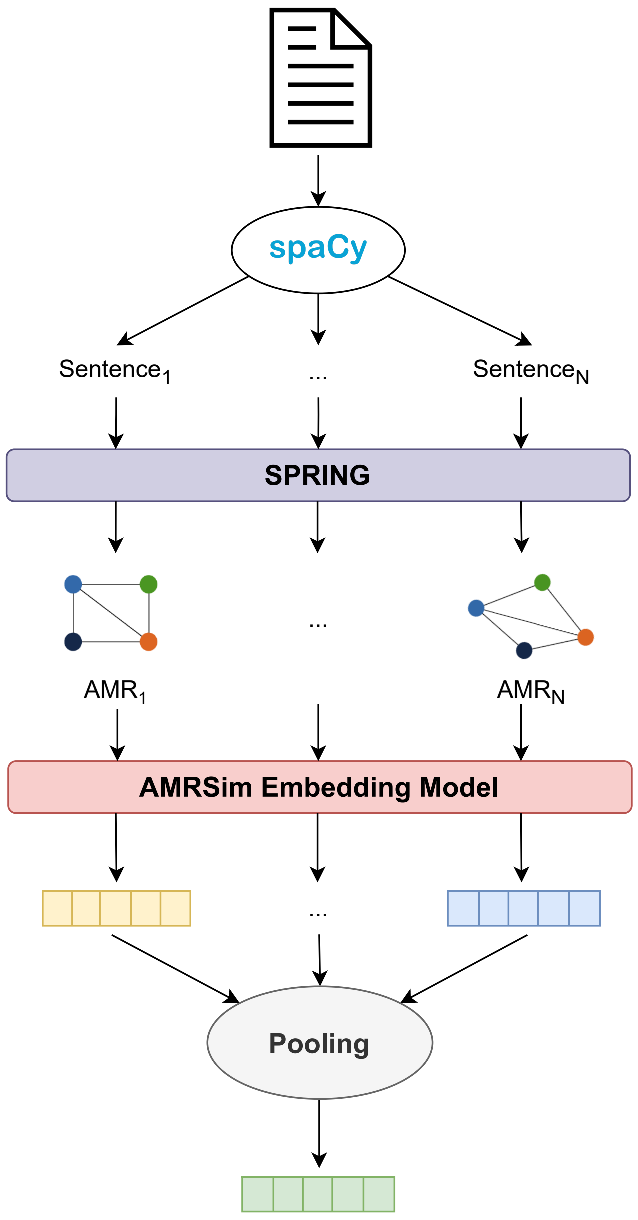


Figure 3.1: Illustration of the text embedding pipeline through AMRs and AMRSim.

which means extracting the AMRs from the topic documents and the dialogue contexts and then embedding them: in fact, it is not possible using the same method seen in the original architecture, since it will end up comparing text embeddings and graph embeddings. However, the downside to this proposal

lies in the high computational time that extracting the embeddings online is going to require: for this reason, placing AMR augmentation in this phase is not optimal and it was not experimented.
Another possibility could be inserting AMRs in the generation phase. Since we can't pass directly graphs to the encoder, the only way to proceed is by extracting AMRs from the texts, encoding them with the chosen embedding method, and passing the resulting vectors to the decoder. The downside of this approach is that the decoder is pre-trained to take in input text embeddings, so using graph embeddings could lead to worse performances.
The only alternative that has been tried in practice is adding AMR processing in the re-ranking phase: doing so, it is possible to test the ability of AMRs to select relevant pieces of knowledge. With this augmentation, the module will contain two different processing parts: on one hand, it will use the same re-ranker model to encode the text, but now the input strings will not contain the snippet text but just its title (as shown in Equation 3.15); the second part, instead, will be composed by the embedding workflow using AMRs on the snippet text (so split the text in sentences, extracting the graphs, encoding them, and then pooling).

$$[s]T[D_T][sep]Title[D_j][sep]T[C][/s] \tag{3.15}$$

From this process, we will obtain two vectors, which will be concatenated and then passed to the feed-forward neural network to compute the re-rank score for the snippet.
In practice, the encoding of each passage is computed offline, so the embedding model is not fine-tuned during the training process: it was observed that not only the training process is much slower when the model is not frozen, but also the performances are considerably worse.

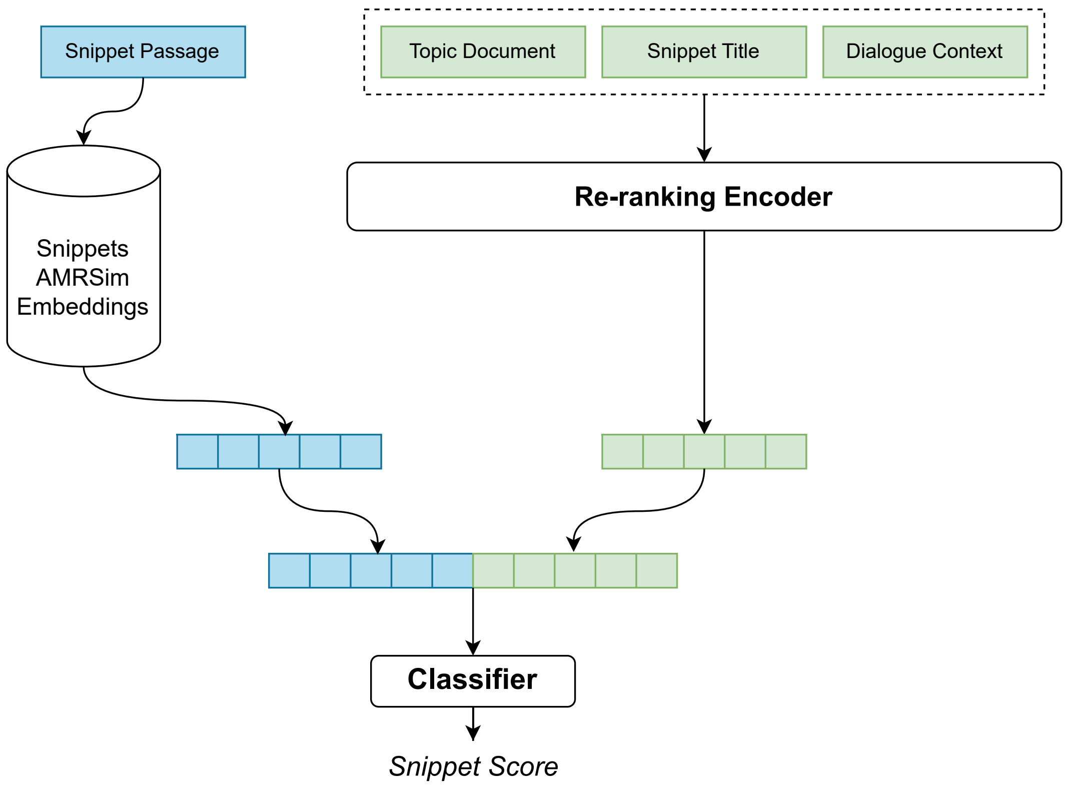


Figure 3.2: Illustration of the AMR-augmented re-ranking module.

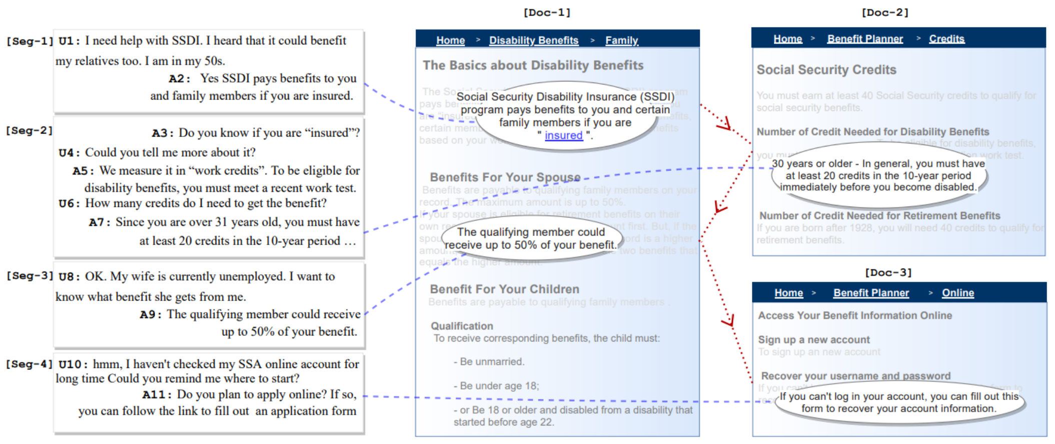
3.4 Experimental Setup

3.4.1 Dataset

The dataset used in this work is MultiDoc2Dial [4], a recent conversational dataset. The data are composed of a set of documents, which represent the external knowledge, and a set of dialogues. The main innovation that this dataset introduces is that each dialogue can contain multiple topics. In fact, the authors use the term 'segments' of a dialogue, where a segment is composed of one or more consecutive turns: two consecutive segments are about different topics and therefore are grounded in different external documents. This provides a more realistic scenario with respect to other conversational datasets, where each dialogue is only about one topic (e.g. Wizard of Wikipedia [39]). In Figure 3.3, it is reported an illustration of a dialogue present in MultiDoc2Dial.

Statistics

MultiDoc2Dial dialogues and external knowledge are divided into 4 different domains: in Table 3.2, we can observe a couple of statistics about the composition of each one of them.



[Seg-1] U1: I need help with SSDI. I heard that it could benefit my relatives too. I am in my 50s.
A2: Yes SSDI pays benefits to you and family members if you are insured.

[Seg-2] A3: Do you know if you are "insured"?
U4: Could you tell me more about it?
A5: We measure it in "work credits". To be eligible for disability benefits, you must meet a recent work test.
U6: How many credits do I need to get the benefit?
A7: Since you are over 31 years old, you must have at least 20 credits in the 10-year period …

[Seg-3] U8: OK. My wife is currently unemployed. I want to know what benefit she gets from me.
A9: The qualifying member could receive up to 50% of your benefit.

[Seg-4] U10: hmm, I haven't checked my SSA online account for long time Could you remind me where to start?
A11: Do you plan to apply online? If so, you can follow the link to fill out an application form

[Doc-1]
Home > Disability Benefits > Family
The Basics about Disability Benefits
Social Security Disability Insurance (SSDI) program pays benefits to you and certain family members if you are " insured ".
Benefits For Your Spouse
The qualifying member could receive up to 50% of your benefit.
Benefit For Your Children
Qualification
To receive corresponding benefits, the child must:
- Be unmarried.
- Be under age 18;
- or Be 18 or older and disabled from a disability that started before age 22.
[Doc-2]
Home > Benefit Planner > Credits
Social Security Credits
Number of Credit Needed for Disability Benefits
30 years or older - In general, you must have at least 20 credits in the 10-year period immediately before you become disabled.
Number of Credit Needed for Retirement Benefits
[Doc-3]
Home > Benefit Planner > Online
Access Your Benefit Information Online
Sign up a new account
Recover your username and password
If you can't log in your account, you can fill out this form to recover your account information.

Figure 3.3: Illustration of a dialogue instance in MultiDoc2Dial (image from [4]).

MultiDoc2Dial already comes with a train, validation, and test split. In Table 3.3, some statistics about it are reported: as we can notice, the average number of turns stays the same across all the sets (it also stays fixed at 13/14 turns across all the domains: this is not reported to not overcomplicate the statistics shown).

Domain	#Documents	#Dialogues
ssa	109	1191
va	138	1337
dmv	149	1328
student	92	940
total	488	4796

Table 3.2: Number of documents and dialogues for each domain in MultiDoc2Dial.

Split	#Dialogues	Avg. Dialogue Length
Train	3474	13.8
Validation	661	13.9
Test	661	13.5

Table 3.3: MultiDoc2Dial: train, validation, and test composition (the average length is defined using the number of turns).

One aspect to underline is that all the numbers just shown correspond to a raw form of the dataset. To obtain the form of the dataset that can be used for training and evaluation, it is necessary to use the instructions and the code provided by the authors[1].
After this process, there are changes in both the external knowledge base and the dialogues instances: the documents are split into a collection of passages, which amount to 4283 texts; on the other hand, for each dialogue, several queries (composed by the last turn and the previous dialogue context) and the correspondent target response are obtained.
The composition of the train, validation, and test splits in the query form is shown in Table 3.4: from this, it is possible to notice that the queries in the train and validation sets tend to be longer in average with the respect to the ones in the test set; the responses, instead, tend to have a similar length in all the three sets.

Split	#Queries	Avg. Query Length	Avg. Response Length
Train	21453	104.6	22.8
Validation	4201	104.2	21.6
Test	4094	96.5	22.3

Table 3.4: MultiDoc2Dial: train, validation, and test composition in the query form (the average lengths are defined using the number of tokens).

[1]https://github.com/IBM/multidoc2dial

3.4.2 Evaluation Method

In order to find the best configuration for the model, several ablation studies have been performed in this work, which will be described in detail in Section 4.1. In order to compare the different architectures, two measures were considered:

- The values of the loss obtained during the training process: this is computed as described in Section 3.1.2;
- ROUGE-L on the validation set, which is one of a set of metrics proposed in [40]. In particular, ROUGE-L is based on the Longest Common Subsequences (LCS) between the text generated by the system and the human-generated reference. Even if these metrics have been formulated in order to evaluate summaries, they can be used also to evaluate generated responses, as it is done in our reference paper [9], which takes the value of ROUGE-L as one measure of evaluation.

In Section 4.2, we will further analyze the performances on the test set by reporting also the Recall@1 and Recall@5 metrics. These metrics are calculated as the proportion of instances where the correct snippet (i.e. the one inspiring the generated response) is retrieved within the top-1 and top-5 results, respectively.

3.4.3 Hardware Setup

Each experiment was run on a workstation having one Nvidia GeForce RTX3090 GPU with 24GB of dedicated memory, 64GB of RAM, and an Intel® Core™ i9-10900X1080 CPU @ 3.70GHz.

3.4.4 Implementation Details

Pre-trained Models

In Table 3.5, we reported the pre-trained models used in this work and the correspondent reference URL. The majority of them are accessed through

Hugging Face[2], such as the Sentence Transformer all-mpnet-base-v2 (used for encoding the external knowledge base and the retrieval queries) or DistilRoberta. In the table are also included pre-trained models such as SPRING and AMRSim, which are instead incorporated in this work through their GitHub repository.

Model	**URL**
All-MPNet-base-v2	`https://huggingface.co/sentence-transformers/all-mpnet-base-v2`
DistilRoberta-base	`https://huggingface.co/distilbert/distilroberta-base`
LongT5-base	`https://huggingface.co/google/long-t5-tglobal-base`
LongT5-large	`https://huggingface.co/google/long-t5-tglobal-large`
UAE-Large-V1	`https://huggingface.co/WhereIsAI/UAE-Large-V1`
SPRING	`https://github.com/SapienzaNLP/spring`
AMRSim(embedding model)	`https://github.com/zzshou/AMRSim`

Table 3.5: List of the models used in this work.

Knowledge Base

Following the work done by the authors of EKo-DoC, the knowledge base is built using FAISS [41], a library that allows to query and efficiently retrieve items (knowledge snippets in this case) based on the similarity between the embeddings.

Experiment Tracking

All trainings make use of Weights & Biases[3], in order to keep track of measures such as the losses and the ROUGE-L score on the validation set.

[2]https://huggingface.co/

[3]`https://wandb.ai`

Chapter 4

Results

This chapter focuses on the results obtained both in the ablation studies and in the final evaluation phase. In particular, the first ones are done on the validation split in order to find the best configuration of the model. On the other hand, the final evaluation uses the test set to have an overview of the model performance on new data and to compare it with the chosen baseline.
Since training is time-consuming, the ablation studies have been conducted using a smaller number of epochs (3): the best configuration was then trained for the same number of epochs used in the reference publication.
The other hyperparameters used for training have also been chosen following the work done by the authors of EKo-DoC: the only exception is the batch size, which was decreased from 32 to 8 due to memory limitations.
We have adopted a base version of LongT5 during ablations, as was done in [9], to speed these studies.

4.1 Ablation Studies

4.1.1 Pseudo-Labels Computation

An aspect that is not specified in [9] is the computation of pseudo-labels. The authors explain that these are computed using the pre-attention scores from the decoder of the LongT5, but they do not clarify whether this is a new instance that is kept frozen, or instead is the same generator within the

Hyperparameter	**Value**
Learning rate	5e-5
Batch size	8
Number of epochs (Ablation Studies)	3
Number of epochs (Final Configuration)	10
K_1	25
K_2	5
Optimizer	Adam
Gradient Clipping	1.0

Table 4.1: Hyperparameters used during training.

architecture which is therefore trained.
In this section, results obtained from these strategies will be presented, since both present advantages and disadvantages.
On one hand, having a new instance of the generator without fine-tuning it, provides a constant score to the same snippets and responses, which could make the training more stable. At the same time, the importance that the pre-trained model gives to a certain snippet could not be optimal at the beginning, so fine-tuning on the generation task could be beneficial. However, this means that the pseudo-labels constantly change during training, making convergence more difficult. Note that, regardless of whether using a frozen model or not, the gradients are never computed during the forward pass when generating the labels: calculating and then updating the gradients could result in a model that returns labels with a distribution increasingly closer to the ones of the retrieval and re-ranking scores, negatively affecting the learning process.
About the pseudo-labels computation, another aspect has been considered and experimented with, which is how the snippets are processed. We have two possibilities for this: batch, which is more efficient, and each snippet separately, which is instead more accurate. In fact, when we process a set of strings at the same time, we have to use padding to make them equal in length: the downsides are that the cross-attention scores could be affected by the pad

tokens, and also we lose the length of each individual sentence, which makes the subsequent averages less precise.
In Table 4.2, the results obtained with these methods are presented: Since it is a fundamental aspect in training the model, the number of epochs used was 5, to better understand how well the strategies work. It is important to emphasize that these experiments were conducted without utilizing the AMR augmentation. This aspect is more closely related to the original architecture than to our addition. This approach allows us to determine the optimal configuration of the baseline model before incorporating the new technique.

Technique	**Validation ROUGE-L**	**Time (h)**
Batch + Frozen Generator	24.71	14
Batch + Training Generator	24.63	14
No Batch + Training Generator	25.24	18

Table 4.2: Results of different techniques to compute the pseudo-labels (5 epochs), using our implementation of EKo-DoC without the AMR augmentation.

Note that there is no significant distinction among the three methods, albeit the no-batch approach exhibits slightly better performance. Nevertheless, the duration to complete the epochs is a crucial factor, leading to the exclusion of the no-batch strategy for the subsequent experiments. Among the batch techniques, preference was given to the one that employs the training generator. In fact, from the trend of retrieval losses (which can be observed in Figure 4.1 and Figure 4.2), it is possible to observe that there is not much change when using the frozen generator, which could lead to fewer improvements and possibly overfitting when continuing the training. This is the same reason why the no-batch experiment was conducted only with the training generator.

4.1.2 Classifier Configuration in the Re-Ranking Module

The initial experimentation with the AMR-augmented model focused on configuring the MLP within the re-ranker. Initially, the configuration was

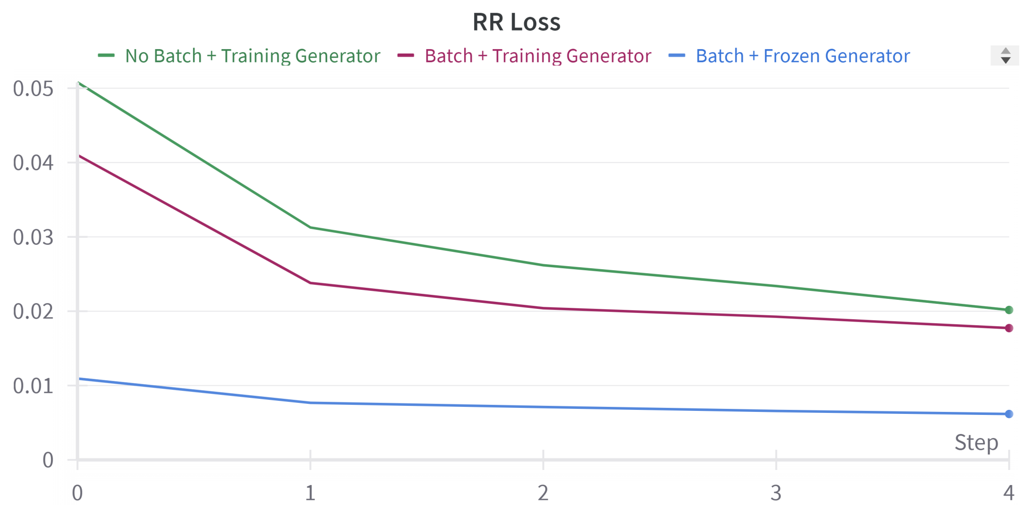


Figure 4.1: Re-ranking loss with different pseudo-labels computation techniques.

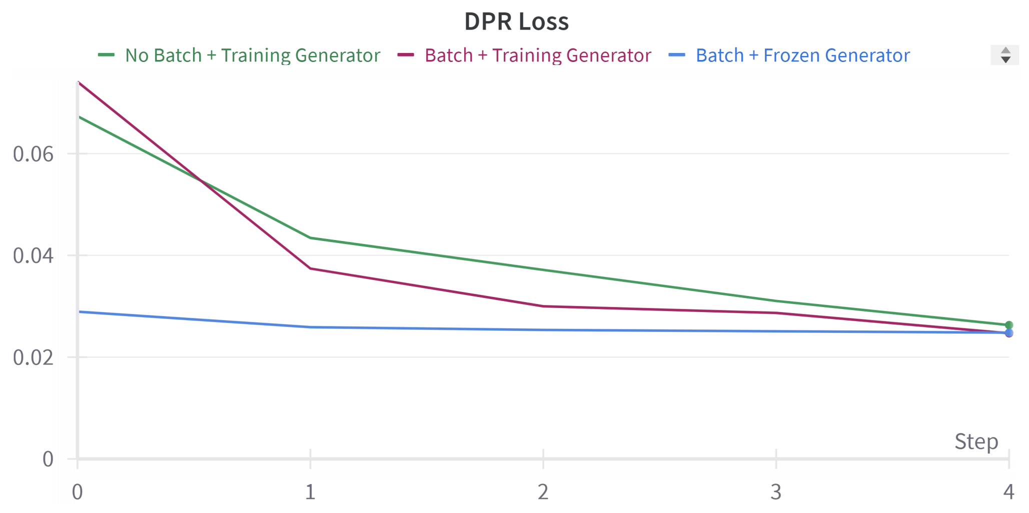


Figure 4.2: DPR loss with different pseudo-labels computation techniques.

not specified in EKo-DoC, so a first attempt involved implementing it with only one linear layer. However, additional processing was deemed potentially beneficial in the AMR-augmented model because of the doubled dimensionality of the input embeddings passed to the MLP. These embeddings result from concatenating the output of the re-ranking encoder and that of the AMR embedding model, counting 1536-dimensional vectors (each model producing 768-dimensional embeddings). Given the risk of information loss when reducing this dimensionality to a single score, exploring additional processing in this phase seemed worthwhile. Specifically, as indicated in Table 4.3, the experiment involved networks composed of 2 and 3 linear layers, employing ReLU as the activation function.

# Layers	Validation ROUGE-L	Time (h)
1	23.08	7
2	23.11	7
3	23.20	7

Table 4.3: Results with increasing number of layers in the re-ranking classifier.

We can deduce that there is no significant distinction among the various setups of the classifier. Hence, we might proceed with the most straightforward option, utilizing only one layer. Nevertheless, the pattern of loss function across the trials indicates that configurations with two or three layers consistently yield lower values, as illustrated in Figure 4.3. Considering this insight, we have chosen to employ a two-layer network.

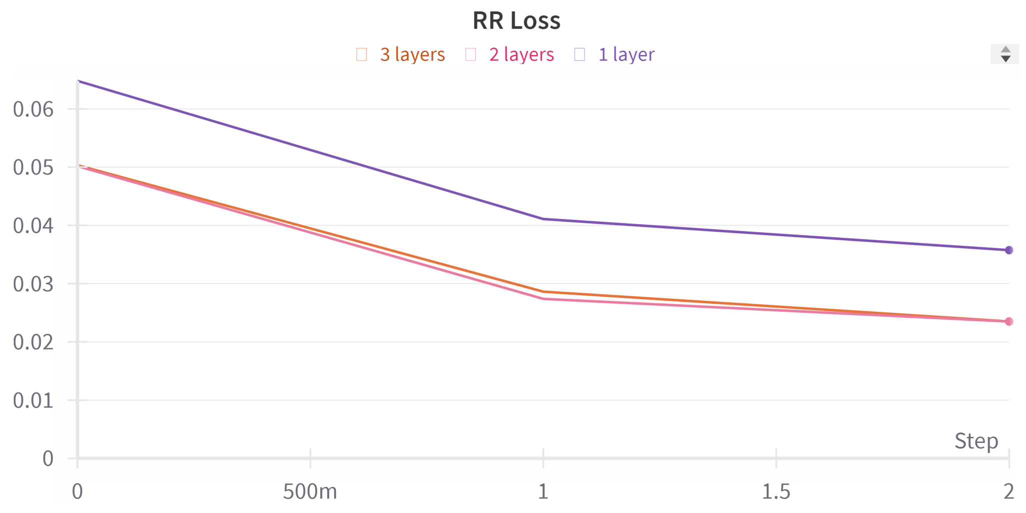


Figure 4.3: RR loss with increasing number of layers in the re-ranking classifier.

4.1.3 Pooling Operation to Combine AMR Embeddings

In the AMR embedding pipeline outlined in Section 3.2, the final stage involves applying a pooling operation to generate a singular vector representing the passage based on the encoded sentences. Various methods can be utilized for this operation, such as computing the mean (as demonstrated in the preceding experiment), calculating the sum, or selecting the maximum value among corresponding embeddings. Table 4.4 illustrates how the outcomes vary depending on the methods attempted.

Pooling operation	**Validation ROUGE-L**
Mean	23.11
Max	23.27
Sum	23.44

Table 4.4: Results with different pooling operations for AMR embeddings.

It appears that the pooling method does not have a significant impact on the results. We could continue with the maximum or the sum, both of which yielded slightly better scores. However, the trend of the ROUGE-L scores (Figure 4.4) does not indicate improvement between the first and second epoch when using the sum. This suboptimal behavior led us to opt for the maximum as the pooling operation.

4.1.4 Re-Ranking Alternative

Another experiment tried in this work was about using different pre-trained models with respect to the ones present in the original architecture, which are: a Sentence Transformer in the retrieval, DistilRoberta in the re-ranking, and LongT5 for the generation. The possibility that seems more reasonable is changing the re-ranker. In fact, according to the current leaderboard[1], all-mpnet-base-v2 is still the best Sentence Transformer, so it is unlikely that another analog embedding model could bring any improvements. Regarding the generator, the main reason why it is better not to try an alternative to

[1] `https://www.sbert.net/docs/pretrained_models.html`

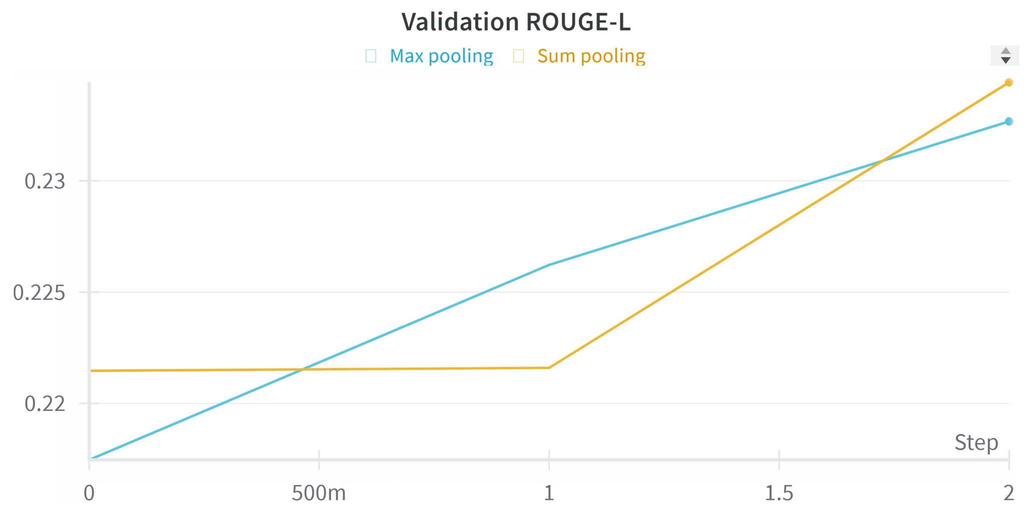


Figure 4.4: ROUGE-L trend on the validation set with max and sum pooling operations.

LongT5 is about the pseudo-labels: changing it means changing the model that produces one of the targets, making fair result comparisons more difficult.
A possible alternative to DistilRoberta in the re-ranker was found in the Massive Text Embedding Benchmark (MTEB) Leaderboard[2], which was introduced in [42]. This benchmark helps to define how well a model encodes an input text by quantifying the effectiveness obtained in a series of different tasks: one of them is precisely the re-ranking.
Looking at the leaderboard, one of the models that currently offers the best overall performance across all tasks is UAE-Large-V1[43]. In particular, it is ranked 6^{th} for the re-ranking and presents a size considerably smaller with respect to the competitors that deliver the best scores, which is an important aspect considering that it will be inserted in an architecture that already presents two pre-trained models. In Table 4.5, the performances of the three best models for the re-ranking are reported together with their sizes and the max input length (expressed in the number of tokens). To evaluate the task, the Mean Average Precision (MAP) was used, and the reported values are the

[2]`https://huggingface.co/spaces/mteb/leaderboard`

MAP average across 4 different datasets.

Model	**Model Size (GB)**	**Max tokens**	**MAP average**
SFR-Embedding-Mistral	14.22	32768	60.64
GritLM 7B	14.48	32768	60.49
E5-Mistral-7b-instruct	14.22	32768	60.21
UAE-Large-V1	1.34	512	59.88

Table 4.5: Re-ranking models from MTEB Leaderboard.

The table shows that the scores achieved by the models are really similar: moreover, 512 is the same maximum dimension that DistilRoberta takes, so also the difference in the maximum input length does not justify using a different model from UAE-Large-V1.
Therefore, in Table 4.6, the comparison between DistilRoberta and the chosen alternative is reported. We notice that the performances achieved with UAE-Large-V1 do not meet the ones obtained with the original re-ranker. However, the distance between those is relatively small: the real limitation in employing UAE-Large-V1 is the long training time required, which limits the possibility of conducting several experiments with it.

Re-ranker	**Validation ROUGE-L**	**Time (h)**
DistilRoberta	23.27	7
UAE-Large-V1	22.86	24

Table 4.6: Results with re-ranking alternatives.

4.1.5 Topic Document

As already mentioned in Section 3.1.1, the topic document is not an entity that can be found in the dataset, but the authors of EKo-DoC define it as the first paragraph of the first grounded document, so it contains text that is also present in the external knowledge base: as a result, the system could retrieve information that is already in the input and leaving out helpful snippets. Moreover, being able to access the document directly without retrieving it is a

less realistic scenario; it means that the user has to provide the agent not only with an utterance but also the text that inspired it.
The authors state that the topic document improves the performance of the model but does not provide the results on MultiDoc2Dial using the entire architecture without the document. Therefore, it is worth checking whether the metric scores depend on it.

Topic Document	**Validation ROUGE-L**
True	23.27
False	24.04

Table 4.7: Results with and without topic document.

The scores shown in Table 4.7 suggest that not including the topic document in the input results in an improvement of our model. This is in contrast to what is suggested in [9]. A possible explanation could be that the topic document leads the system to retrieve content too similar to itself: therefore, by excluding it, we could increase the diversity of the snippets selected. Another reason could be that, without the topic document, the model focuses more on the dialogue context, which results in choosing texts more aligned with it.

4.1.6 Generation Model

As mentioned above, we used the base LongT5 model to accelerate training during ablation studies. In this final experiment, we checked whether the large version brought a relevant improvement to our architecture. Since the large version is likely to take more time to complete its training, we employed all 10 epochs here to ensure a fair comparison.

LongT5 version	**Validation ROUGE-L**
Base	24.99
Large	25.06

Table 4.8: Results with the base and large version of LongT5.

The results in Table 4.8 show that there is not a relevant difference between the performances achieved using the two generators: however, the large version performed marginally better, so we employed it for the final comparison with the baseline.

4.1.7 Final Configuration of the Model

In Table 4.9 the findings of the ablation studies are summarized. These will be used for the final evaluation on the test set.

Hyperparameter	**Value**
Pseudo-labels computation	Batch + training generator
Number of layers in RR classifier	2
Pooling operation for AMR embeddings	Max
Re-ranker	DistilRoberta
Usage of topic document	False
LongT5 version	Large

Table 4.9: Resume of the ablation studies.

4.2 Comparison with Baselines

Table 4.10 presents the performance comparison on the test set. It includes the results obtained using our proposed model and our implementation of the original EKo-DoC architecture; the scores from the original publication are also reported.

Model	**ROUGE-L**
EKo-DoC + AMR (ours)	25.74
EKo-DoC (ours)	25.47
EKo-DoC	39.02

Table 4.10: ROUGE-L scores on the test.

The findings indicate a small beneficial effect of the AMRs when compared to the non-augmented model, specifically our version of EKo-DoC. However, the performance of both implementations does not reach the levels reported in [9]. Consequently, we conducted an examination of the Recall@1 and Recall@5 metrics to evaluate the retrieval component's ability to identify relevant source snippets.

Model	**Recall@1**	**Recall@5**
EKo-DoC + AMR (ours)	2.32	11.97
EKo-DoC + AMR (ours) - No RR	14.58	34.98
EKo-DoC (ours)	3.37	11.63
EKo-DoC (ours) - No RR	9.72	21.74
EKo-DoC	41.60	-
EKo-DoC - No RR	33.88	-

Table 4.11: Recall scores on the test set.

The findings presented in Table 4.11 indicate that our models struggle to retrieve accurate snippets. Notably, we observe an improvement in scores when the re-ranker is excluded. Based on this analysis, we can infer that our model fails to properly train the re-ranker component as intended, leading to an inability to enhance the architecture's snippet selection capability compared to the original implementation.

4.3 Domain-Specific Results

In Section 3.4.1, it is noted that all dialogues within MultiDoc2Dial fall into one of four distinct domains. The results calculated for each domain are presented in Table 4.12.

Consistent performances are evident across various domains, demonstrating the versatility of both models to engage in diverse conversations. The only light exception can be found in the "student" domain, for which the scores are lower for the two architectures.

Domain	ROUGE-L	
	EKo-DoC + AMR (ours)	EKo-DoC (ours)
dmv	25.20	25.64
student	24.20	23.20
va	26.37	24.87
ssa	26.67	27.58

Table 4.12: ROUGE-L scores computed on each domain of the test set.

4.4 Input-Output Examples

In Figure 4.5, we present two dialogue contexts provided to the model as input, together with the responses generated by our implementations. In the initial example, both the grammar and semantics of the returned texts are accurate. However, the response from the original architecture aligns more closely with the ground truth. Moving to the second example, our implementation of EKo-DoC produces an output that fails to address the user's final utterance, veering off-topic. Conversely, the AMR-augmented model delivers a more precise answer. Nevertheless, both outputs are incomplete as they are truncated before completing the sentences. Overall, our proposition demonstrates an understanding of the dialogue's general topic and produces responses relevant to it. The generated texts may exhibit imperfections and fail to fully address the provided questions.

Examples

Dialogue Context

user: How do i find a Pre licensing course?

...

user: When will i be given the certificate?

agent: When you successfully finish the course.

user: What happens if the certificate expires?

Generated response

Ours: You must renew it before the expiration date.

EKo-DoC (Ours): You must complete the course again to receive the certificate required to schedule a road test.

Ground truth

You must complete the course again.

Dialogue Context

user: I would like to know what would help prepare for college someday?

...

user: Is there a 9th grade checklist available?

agent: Yes, a 9th grade checklist is available at this time.

user: Will the checklist help with websites to explore?

Generated response

Ours: Yes, the checklist will help you learn about many different websites that can help you prepare financially and

EKo-DoC (Ours): Yes, the guide will tell you what to expect during and after school and will tell you what

Ground truth

Yes, our checklists make suggestions on many websites to explore as you prepare academically and financially.

Figure 4.5: Examples of dialogue inputs, their ground truth response, and the generated text by our implementations.

Chapter 5

Conclusions and Future Work

This thesis explored the combined usage of Abstract Meaning Representations and retrieval technologies in conversational agents. We started with an overview of the literature and the current state-of-the-art methods, which showed that both techniques have not been used together within the same architecture. After selecting EKo-DoC [9] as the base of our work, we inserted an AMR augmentation in it, to improve the knowledge selection step in its pipeline. We then conducted several ablation studies to find the best configuration of the model after the graph integration. The results showed that AMRs brought a small improvement in the final performances. However, it is clear that the positive impact is heavily limited by the re-ranker's abilities, which leads to poor quality of the retrieved documents and also to some imperfections in the responses generated. We are confident to obtain a strong increase in the results once the flaws observed are resolved: these are present also in our implementation of the original version of the model, and, therefore, are not connected to the AMR augmentation.
We will explore these aspects in more detail in the next section, providing also some ideas for future work.
Finally, we will conclude this work with some ethical considerations about conversational agents and retrieval augmentation in them.

5.1 Limitations

In Section 4.2, we noticed that the main limitation of our proposal is not linked to the AMR augmentation, but instead to the abilities of our re-ranker. The Recall's values suggest that this module is not able to pick the most relevant pieces of external knowledge between those chosen by the DPR: in fact, the scores increase when the K_2 snippets are selected directly after the retrieval. Since the original design of the re-ranker is quite straightforward (just DistilRoberta and a feed-forward neural network), it is unlikely that our implementation failed in incorporating this module correctly. Instead, how this module is trained could be the reason behind its poor performance. The pseudo-labels are an interesting and innovative concept to create an end-to-end trainable architecture, but still, an imperfect computation of them affects certainly the final abilities of the model. Future work could comprehend further reviews of our code, in order to reach the original performances reported in [9] and create a common base to explore better the AMR augmentation proposed in this thesis.
Then, another interesting possibility could be exploring different generation models: this could have a positive impact not only on the generation task but also on producing better pseudo-labels. For example, the recent FLAN-T5 [44] could be an intriguing option. This experiment was not done in this work, since, as already seen in Chapter 4, the experiments could be quite time-consuming.

5.2 Ethical Considerations

With the increasing advancement of their ability to reproduce human language, conversational agents raise some ethical questions. First of all, the ideas that they express could present bias, for example, social or political: these could lead to responses that are not objectively true or that are discriminatory towards people. It is always a fundamental step to check if the data present any problematic content, otherwise this will be learned by the model during training: as stated in [9], all the data used in this work do not seem to contain any offensive language.

About bias, retrieval-augmented text generation poses a further problem: the sources that the agent uses to produce the answer are not only the ones seen during training but also the ones present in the external pool. We expressed in Section 1.2 how this can be beneficial since it allows the model to stay updated by just adding new documents to it: while this is true, it also exposes the model to more bias, and therefore it requires a review of the content every time the knowledge base is enriched.
On the other hand, a retrieval component could also represent a tool to remove bias from the data used. In fact, by checking the snippets selected by the system when a problematic response is given, we can immediately find the source of the problem and remove the document from the external pool.
Another important ethical aspect that regards conversational agents, is the one of explainability, which is about the transparency of a model. As stated in Section 1.2, incorporating a retrieval component makes a system more explainable because the selected knowledge already gives the users a better understanding of the reasoning performed.

This thesis honors and supports the ACL Code of Ethics. All pre-trained models and corpora used in this work are publicly available.

Acknowledgements

I would like to acknowledge my supervisor Prof. Gianluca Moro, co-supervisor Giacomo Frisoni, and colleague Paolo Italiani, who assisted me in the work done during this thesis.

Bibliography

[1] Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George van den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego de Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, Tom Hennigan, Saffron Huang, Loren Maggiore, Chris Jones, Albin Cassirer, Andy Brock, Michela Paganini, Geoffrey Irving, Oriol Vinyals, Simon Osindero, Karen Simonyan, Jack W. Rae, Erich Elsen, and Laurent Sifre. Improving language models by retrieving from trillions of tokens. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato, editors, *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 2206–2240. PMLR, 2022. URL `https://proceedings.mlr.press/v162/borgeaud22a.html`.

[2] Hongyun Cai, Vincent W. Zheng, and Kevin Chen-Chuan Chang. A comprehensive survey of graph embedding: Problems, techniques, and applications. *IEEE Trans. Knowl. Data Eng.*, 30(9):1616–1637, 2018. doi: 10.1109/TKDE.2018.2807452. URL `https://doi.org/10.1109/TKDE.2018.2807452`.

[3] Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. Abstract meaning representation for sembanking. In Stefanie Dipper, Maria Liakata, and Antonio Pareja-Lora, editors, *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse, LAW-ID@ACL 2013, August 8-9, 2013, Sofia, Bulgaria*,

pages 178–186. The Association for Computer Linguistics, 2013. URL https://aclanthology.org/W13-2322/.

[4] Song Feng, Siva Sankalp Patel, Hui Wan, and Sachindra Joshi. Multi-doc2dial: Modeling dialogues grounded in multiple documents. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih, editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 6162–6176. Association for Computational Linguistics, 2021. doi: 10.18653/V1/2021.EMNLP-MAIN.498. URL https://doi.org/10.18653/v1/2021.emnlp-main.498.

[5] Mattias Wahde and Marco Virgolin. Conversational agents: Theory and applications. *CoRR*, abs/2202.03164, 2022. URL https://arxiv.org/abs/2202.03164.

[6] Huayang Li, Yixuan Su, Deng Cai, Yan Wang, and Lemao Liu. A survey on retrieval-augmented text generation. *CoRR*, abs/2202.01110, 2022. URL https://arxiv.org/abs/2202.01110.

[7] Robert T. Kasper. A flexible interface for linking applications to penman's sentence generator. In *Speech and Natural Language: Proceedings of a Workshop Held at Philadelphia, USA, HLT 1989, February 21-23, 1989*. ACL, 1989. URL https://aclanthology.org/H89-1022/.

[8] Martha Palmer, Paul R. Kingsbury, and Daniel Gildea. The proposition bank: An annotated corpus of semantic roles. *Comput. Linguistics*, 31(1): 71–106, 2005. doi: 10.1162/0891201053630264. URL https://doi.org/10.1162/0891201053630264.

[9] Tuan Manh Lai, Giuseppe Castellucci, Saar Kuzi, Heng Ji, and Oleg Rokhlenko. External knowledge acquisition for end-to-end document-oriented dialog systems. In Andreas Vlachos and Isabelle Augenstein, editors, *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2023, Dubrovnik, Croatia, May 2-6, 2023*, pages 3615–3629. Association for Computational

Linguistics, 2023. doi: 10.18653/V1/2023.EACL-MAIN.264. URL `https://doi.org/10.18653/v1/2023.eacl-main.264`.

[10] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 3980–3990. Association for Computational Linguistics, 2019. doi: 10.18653/V1/D19-1410. URL `https://doi.org/10.18653/v1/D19-1410`.

[11] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019. URL `http://arxiv.org/abs/1907.11692`.

[12] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter. *CoRR*, abs/1910.01108, 2019. URL `http://arxiv.org/abs/1910.01108`.

[13] Mandy Guo, Joshua Ainslie, David C. Uthus, Santiago Ontañón, Jianmo Ni, Yun-Hsuan Sung, and Yinfei Yang. Longt5: Efficient text-to-text transformer for long sequences. In Marine Carpuat, Marie-Catherine de Marneffe, and Iván Vladimir Meza Ruíz, editors, *Findings of the Association for Computational Linguistics: NAACL 2022, Seattle, WA, United States, July 10-15, 2022*, pages 724–736. Association for Computational Linguistics, 2022. doi: 10.18653/V1/2022.FINDINGS-NAACL.55. URL `https://doi.org/10.18653/v1/2022.findings-naacl.55`.

[14] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz

Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL `https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html`.

[15] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics, 2019. doi: 10.18653/V1/N19-1423. URL `https://doi.org/10.18653/v1/n19-1423`.

[16] Ruiqi Guo, Philip Sun, Erik Lindgren, Quan Geng, David Simcha, Felix Chern, and Sanjiv Kumar. Accelerating large-scale inference with anisotropic vector quantization. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 3887–3896. PMLR, 2020. URL `http://proceedings.mlr.press/v119/guo20h.html`.

[17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages

5998–6008, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

[18] Dani Yogatama, Cyprien de Masson d'Autume, and Lingpeng Kong. Adaptive semiparametric language models. *CoRR*, abs/2102.02557, 2021. URL https://arxiv.org/abs/2102.02557.

[19] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G. Carbonell, Quoc Viet Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. In Anna Korhonen, David R. Traum, and Lluís Màrquez, editors, *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 2978–2988. Association for Computational Linguistics, 2019. doi: 10.18653/V1/P19-1285. URL https://doi.org/10.18653/v1/p19-1285.

[20] Di Jin, Eileen Pan, Nassim Oufattole, Wei-Hung Weng, Hanyi Fang, and Peter Szolovits. What disease does this patient have? A large-scale open domain question answering dataset from medical exams. *CoRR*, abs/2009.13081, 2020. URL https://arxiv.org/abs/2009.13081.

[21] Michele Bevilacqua, Rexhina Blloshmi, and Roberto Navigli. One SPRING to rule them both: Symmetric AMR semantic parsing and generation without a complex pipeline. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 12564–12573. AAAI Press, 2021. doi: 10.1609/AAAI.V35I14.17489. URL https://doi.org/10.1609/aaai.v35i14.17489.

[22] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In Dan Jurafsky, Joyce Chai,

Natalie Schluter, and Joel R. Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 7871–7880. Association for Computational Linguistics, 2020. doi: 10.18653/V1/2020.ACL-MAIN.703. URL https://doi.org/10.18653/v1/2020.acl-main.703.

[23] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *CoRR*, abs/1609.02907, 2016. URL http://arxiv.org/abs/1609.02907.

[24] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. *CoRR*, abs/1710.10903, 2017. URL http://arxiv.org/abs/1710.10903.

[25] Jun Chen and Haopeng Chen. Edge-featured graph attention network. *CoRR*, abs/2101.07671, 2021. URL https://arxiv.org/abs/2101.07671.

[26] Shaked Brody, Uri Alon, and Eran Yahav. How attentive are graph attention networks? *CoRR*, abs/2105.14491, 2021. URL https://arxiv.org/abs/2105.14491.

[27] Shu Cai and Kevin Knight. Smatch: an evaluation metric for semantic feature structures. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, ACL 2013, 4-9 August 2013, Sofia, Bulgaria, Volume 2: Short Papers*, pages 748–752. The Association for Computer Linguistics, 2013. URL https://aclanthology.org/P13-2131/.

[28] Juri Opitz, Letitia Parcalabescu, and Anette Frank. AMR similarity metrics from principles. *CoRR*, abs/2001.10929, 2020. URL https://arxiv.org/abs/2001.10929.

[29] Juri Opitz, Angel Daza, and Anette Frank. Weisfeiler-leman in the BAMBOO: novel AMR graph metrics and a benchmark for AMR graph similarity. *CoRR*, abs/2108.11949, 2021. URL https://arxiv.org/abs/2108.11949.

[30] Ziyi Shou and Fangzhen Lin. Evaluate AMR graph similarity via self-supervised learning. In Anna Rogers, Jordan L. Boyd-Graber, and Naoaki Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 16112–16123. Association for Computational Linguistics, 2023. doi: 10.18653/V1/2023.ACL-LONG.892. URL `https://doi.org/10.18653/v1/2023.acl-long.892`.

[31] Sha Li, Mahdi Namazifar, Di Jin, Mohit Bansal, Heng Ji, Yang Liu, and Dilek Hakkani-Tur. Enhanced knowledge selection for grounded dialogues via document semantic graphs. *CoRR*, abs/2206.07296, 2022. doi: 10.48550/ARXIV.2206.07296. URL `https://doi.org/10.48550/arXiv.2206.07296`.

[32] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019. URL `https://api.semanticscholar.org/CorpusID:160025533`.

[33] Gautier Izacard and Edouard Grave. Distilling knowledge from reader to retriever for question answering. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL `https://openreview.net/forum?id=NTEz-6wysdb`.

[34] Gautier Izacard and Edouard Grave. Leveraging passage retrieval with generative models for open domain question answering. In Paola Merlo, Jörg Tiedemann, and Reut Tsarfaty, editors, *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, EACL 2021, Online, April 19 - 23, 2021*, pages 874–880. Association for Computational Linguistics, 2021. doi: 10.18653/V1/2021.EACL-MAIN.74. URL `https://doi.org/10.18653/v1/2021.eacl-main.74`.

[35] Akari Asai, Matt Gardner, and Hannaneh Hajishirzi. Evidentiality-guided generation for knowledge-intensive NLP tasks. In Marine Carpuat,

Marie-Catherine de Marneffe, and Iván Vladimir Meza Ruíz, editors, *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL 2022, Seattle, WA, United States, July 10-15, 2022*, pages 2226–2243. Association for Computational Linguistics, 2022. doi: 10.18653/V1/2022.NAACL-MAIN.162. URL `https://doi.org/10.18653/v1/2022.naacl-main.162`.

[36] Petr Baudis and Jan Sedivý. Sentence pair scoring: Towards unified framework for text comprehension. *CoRR*, abs/1603.06127, 2016. URL `http://arxiv.org/abs/1603.06127`.

[37] Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. A SICK cure for the evaluation of compositional distributional semantic models. In Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Hrafn Loftsson, Bente Maegaard, Joseph Mariani, Asunción Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Ninth International Conference on Language Resources and Evaluation, LREC 2014, Reykjavik, Iceland, May 26-31, 2014*, pages 216–223. European Language Resources Association (ELRA), 2014. URL `http://www.lrec-conf.org/proceedings/lrec2014/summaries/363.html`.

[38] Christopher Morris, Martin Ritzert, Matthias Fey, William L. Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 4602–4609. AAAI Press, 2019. doi: 10.1609/AAAI.V33I01.33014602. URL `https://doi.org/10.1609/aaai.v33i01.33014602`.

[39] Emily Dinan, Stephen Roller, Kurt Shuster, Angela Fan, Michael Auli, and Jason Weston. Wizard of wikipedia: Knowledge-powered conversational

agents. *CoRR*, abs/1811.01241, 2018. URL http://arxiv.org/abs/1811.01241.

[40] Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics. URL https://aclanthology.org/W04-1013.

[41] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus. *CoRR*, abs/1702.08734, 2017. URL http://arxiv.org/abs/1702.08734.

[42] Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. MTEB: massive text embedding benchmark. In Andreas Vlachos and Isabelle Augenstein, editors, *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2023, Dubrovnik, Croatia, May 2-6, 2023*, pages 2006–2029. Association for Computational Linguistics, 2023. doi: 10.18653/V1/2023.EACL-MAIN.148. URL https://doi.org/10.18653/v1/2023.eacl-main.148.

[43] Xianming Li and Jing Li. Angle-optimized text embeddings. *CoRR*, abs/2309.12871, 2023. doi: 10.48550/ARXIV.2309.12871. URL https://doi.org/10.48550/arXiv.2309.12871.

[44] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Y. Zhao, Yanping Huang, Andrew M. Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. Scaling instruction-finetuned language models. *CoRR*, abs/2210.11416, 2022. doi: 10.48550/ARXIV.2210.11416. URL https://doi.org/10.48550/arXiv.2210.11416.