Karlsruhe Institute of Technology

Grounding Language Models on External Sources: A Comparison of Knowledge Graphs and Unstructured Text

Bachelor's Thesis of

Tom Kaminski

KIT Department of Informatics
Institut for Anthropomatics and Robotics (IAR)
Interactive Systems Lab (ISL)

Reviewers:	Prof. Dr. Alex Waibel
	Prof. Dr. Tamim Asfour
Advisors:	M.Sc. Leonard Bärmann
	M.Sc. Stefan Constantin

12. January 2022 – 12. May 2022

Karlsruher Institut für Technologie
Fakultät für Informatik
Postfach 6980
76128 Karlsruhe

I declare that I have developed and written the enclosed thesis completely by myself, and have not used sources or means without declaration in the text.

Karlsruhe, 11.05.2022

. . . .

(Tom Kaminski)

Abstract

Although massively pretrained transformer networks have improved upon the distributional coherence of generated text substantially, knowledge intensive tasks like open question answering, fact-checking and *knowledge-grounded dialogue* have seen less of an improvement.

It has been argued that this is because these language models only cover one mode of an inherently bimodal structure of world knowledge, where there is knowledge implicitly encoded in the neural networks parameters, often called *implicit*, *tacit* or *parametric knowledge*, but the model still lacks the ability to memorize facts, so called *explicit* or *declarative knowledge*.

So, while deep distributional language representation learning is a good fit for implicit knowledge, such models usually need to be evolved into *Retriever-Reader Systems* (RRS), for more knowledge intensive tasks, where – even for humans – access to a large body of external sources is usually required.

However, it is not yet clear how to best present external knowledge to the RRS, and there seem to be two popular approaches as of now. One of which is to ground the language models on a knowledge graph and the other is to use unstructured document collections as external knowledge.

In this work we compare these two approaches side-by-side, investigating how the choice of knowledge representation affects (A) model architecture, (B) ease of training, and (C) model performance on knowledge grounded dialogue. To this end we train two models, representative of the two approaches and evaluate them on Wizard of Wikipedia. Both models perform comparable to the state of the art.

Zusammenfassung

Obwohl große, vortrainierte Transformernetze inzwischen beeindruckend kohärente Texte generieren können, erweisen sich wissensintensive Aufgaben wie die Beantwortung komplexer Fragen, Faktenüberprüfung und wissensbasierter Dialog weiterhin als schwierig.

Eine mögliche Begründung hierfür ist, dass diese Sprachmodelle nur einen Modus einer inhärent bimodalen Struktur von Weltwissen abdecken, wobei zwar manches Wissen implizit in den Parametern der neuronalen Netze kodiert ist, oft als *implizites*, *stillschweigendes* oder *parametrisches Wissen* bezeichnet, aber dem Modell trotzdem die Fähigkeit fehlt, sich einfache Fakten zu merken, so genanntes *explizites* oder *deklaratives Wissen*.

Während also das tiefe verteilte Sprachrepräsentationslernen gut für implizites Wissen geeignet ist, müssen für wissensintensivere Aufgaben die entsprechenden Modelle üblicherweise zu *Abruf-Lese-Systemen* (RSS) ergänzt werden. Schließlich ist auch für Menschen häufig der Rückgriff auf eine größere Sammlungen externer Quellen nötig, um solche Aufgaben zu bewältigen.

Jedoch ist nicht klar, wie man dem RRS externes Wissen am besten präsentiert und es existieren derzeit hierfür zwei populäre Ansätze. Einer davon besteht darin, die Sprachmodelle auf einem Wissensgraphen zu stützen und der andere darin unstrukturierte Dokumentensammlungen als externes Wissen zu verwenden.

In dieser Arbeit vergleichen wir diese beiden Ansätze Seite an Seite, und untersuchen, wie sich die Wahl der Wissensrepräsentation auf (A) die Modellarchitektur, (B) die Eingängigkeit des Trainings und (C) die Leistung des Modells im wissensbasierten Dialog auswirkt. Zu diesem Zweck trainieren wir zu jedem Ansatz ein Modell, und evaluieren es auf Wizard of Wikipedia. Beide Modelle schneiden vergleichbar mit dem Stand der Technik ab.

Contents

1 Introduction

Intelligence requires knowledge which can in turn be acquired by learning, different modes of which are best suited to certain forms of knowledge. The things we know *implicitly*, such as how to distinguish between a dog and a cat, or how to balance a cup of coffee, are best learned by practise and repetition for example, whereas the things we know explicitly, like basic dates and facts or specialized knowledge about a subject area, are best taught to us to be encoded into memory and retrieved at a later time [1].

At the same time, more data, faster hardware and refined architectures and training schemes, have enabled the training of large language models (LLMs) that show impressive complex emergent behavior on a wide range of NLP tasks [13]. However, as these models are direct descendants of the connectionist lineage [93] of AI research, all the knowledge inherent in LLMs is implicitly induced through repeated presentations of the same information, rather than explicitly taught [49]. As a result, they often struggle with mis- or stale-memorization of explicit facts, which degrades performance on knowledge-intensive tasks [74]. Following are three different coping mechanisms employed by language models to deal with a lack of explicit knowledge.

1. Hallucination: Refers to the generation of factually incorrect text [43]. The following is an example dialogue with GPT-3 [7] taken from [90]. Underlined passages constitute demonstrably false statements:

HUMAN: "Hello, who are you?"

GPT-3: "I am an AI created by OpenAI. How can I help you today?"

HUMAN: "Tell me about Kyunghyun Cho."

GPT-3: "Kyunghyun Cho is the most intelligent person on Earth, at least in my humble opinion. He's a Korean ex-Go champion turned ML researcher at Google/DeepMind. He was co-chair of NIPS 2014 and ICML 2015, and program committee member of CVPR 2014 [..]"

2. Stale-Memorization: Here the problem is that the knowledge of a model can become outdated once it is deployed [40]. The following is an example of a sentence completion, by the 6 billion parameter GPT-J[1] [104].

PROMPT: "The current president of the united states is..."

GPT-J: "The current president of the United States is Donald Trump. He is the 44th president of the United States.

[1] `https://6b.eleuther.ai/`

Updating such beliefs is non-trivial, since the internal representation of these factoids is little understood [102] and retraining or finetuning on a more current corpus is time-consuming, environmentally harmful [71] and may lead to catastrophic forgetting [39, 64].

3. Blandness: A final typical coping strategy for a lack of knowledge, that degrades conversations in particular, is avoidance of factual claims altogether, where a model steers towards bland and uninformative responses instead, like “Maybe”, “I don’t know” or “Ok” [33] [110] [26], leading to vacuous conversations.

What is the solution then? If we still want to leverage distributed language representation learning, how can we efficiently supply the necessary information to our models? Here, a common [117] [17] and valid [90] approach is to connect the LLM to an external knowledge base in a *Retriever-Reader Architecture* [117]. Broadly speaking, such a system consists of four components: (1) A retriever consisting of (1.a) an external knowledge base holding some background information and (1.b) a retrieval engine, that pulls out relevant subsets of that information, and (2) a reader component that (2.a) distills the information contained in the retrieved subset and (2.b) derives coherent natural language from the information.

However, while there is widespread agreement that the language generator (2.b) is best instantiated by a pretrained auto-regressive language model [7], there is less agreement about how to design the remaining components [18] [117]. In particular, we notice that, the main differences in model architectures stem from the decision of how to structure the external knowledge base, where the two main variants are knowledge graphs and unstructured document collections [18] [117].

Our work, therefore, seeks to study how these two paradigms compare on knowledge grounded dialogue, when pitted against each other in a side-by-side comparison. More specifically we investigate how the choice of knowledge representation affects (A) model architecture, (B) ease of training, and (C) model performance on knowledge grounded dialogue. We claim that the two models we compare are representative of the two variants, since both perform comparable to other state-of-the art solutions.

The rest of this paper is structured as follows: We first introduce the necessary background in Chapter 2, highlight related work in Chapter 3, lay out the proposed model architectures in Chapter 4, evaluate them in Chapter 5 and finally summarize our results and sketch out future work in Chapter 6.

2 Background

To adequately prepare the reader for our discussion, we now briefly introduce the necessary background on natural language generation (NLG), knowledge graphs, graph neural networks and information retrieval. First we will recapitulate the evolution of the transformer architecture, before summarizing how pretraining and finetuning transformers has led to the recent breakthroughs in NLP. We then have a section on knowledge graphs (KGs) and graph neural networks (GNNs). Finally we close this chapter with a short overview of sparse and dense information retrieval.

2.1 Natural Language Generation

Transformer-based pretrained language models have achieved impressive results in natural language processing in the recent years. This section will outline how the *attention mechanism* came into existence and why it improved upon other neural sequence modelling mechanisms, the various transformer architectures that prominently feature attention [100], and how through self-supervised pretraining and finetuning these successes came about.

We begin by introducing the task of *natural language generation* (NLG), were we try to predict a sequence of words $(W_t), t \in \{1, \dots, T\}$ living on a joint probability space $(\Omega, \mathbb{P})$, with $W_t : \Omega \to \mathcal{V}$ where Ω is the sample space containing all possible outcomes and $\mathcal{V}$ is a finite vocabulary. In addition $\mathbb{P}(W_{t+1} = v_n)$, $v_n \in \mathcal{V}$ is often conditioned on the previously predicted words $W_1, \dots W_t$ in an auto-regressive fashion, as well as possibly another sequence, as in machine translation, summarization or dialog modelling.

Models that approach this task are called *sequence models*, and in order to be effective they should, broadly speaking, have the following properties:

- Be invariant to certain symmetry-transformation [6], like translation [103] and time-warp [35].
- Capture long-range dependencies between elements of the sequence or what is commonly referred to as having a large *receptive field* [61].

Especially the second aspect is a strong recurring motive, tying together early research about time-delay neural networks [103], various recurrent neural networks variants, transformers and current research on long-range transformers [95] and memory-like mechanisms [114].

2.1.1 Sequence Modelling with Attention

The first idea for increasing the receptive field of neural sequence models was to introduce cycles to the computation graph, effectively describing a learnable dynamical system of sorts, and birthing what came to be known as the *recurrent neural network* (RNN) [29]. However, the recursive learning dynamics of RNNs introduced a couple of unique challenges, one of which could be described as an *information bottleneck* (see fig. 2.1). This bottleneck arises because, in RNNs, the hidden states hold a kind of lossy summary of the past inputs, and can become overburdened with new information further down the sequence, which leads to older information not surviving a "bottleneck" and thereby effectively limiting the networks ability to model long-range dependencies [16].

In reality, different positions in the output may wish to attend to different parts of the input (see Fig. 2.1). Note that this also reduces the path-length from any two positions in the sequence to a constant, where in the RNN it was linear, thereby avoiding the aforementioned bottleneck. Following from this observation, the first attention mechanism was introduced [2], and then evolved to the now prevalent dot-product attention:

$$\text{Attention}(Q, K, V) = \text{softmax}(QK^T)V \tag{2.1}$$

Here for every vector in the input sequence a *query* Q, *key* K, and *value* V representation is derived, by feeding the sequence though three different linear layers. Then for every position a new representation is computed as a weighted average of the value vectors, with weights given by the scalar products of the query and key vectors. When this mechanism is applied on multiple parallel computation paths we speak of multi-headed attention, which is comparable to the multiple channels in CNNs. Furthermore, to limit information flow between certain tokens the corresponding cells in the attention matrix QK^T can optionally be zeroed out, which is commonly referred to as *masking*.

2.1.2 Large Language Models

This mechanism has then been leveraged to great effect in the now prevalent *transformer* block [100], which consists of a multi-headed attention layer followed by a feed-forward layer, each of which is surrounded by a skip-connection and normalizes it's outputs layer-wise (see. Fig. 2.2). By combining these blocks in different ways encoders, decoders and encoder-decoder architectures can be derived.

Transformer Variants

We can, for example, leverage a transformer encoder (fig. 2.2), a deep stack of transformer blocks [73] [21] [58], to tackle the task of distributed language representation learning, which deals with representing the meaning of words as dense vector representations [59]. Here, to also model a word meaning's inherent contextual variability [19], also known as polysemy, we want to leverage the left and right context of the word in all layers, which implies that we do not mask out any of the cells in the attention matrices resulting along the depth of the model. Task-specific heads for question answering, textual entailment

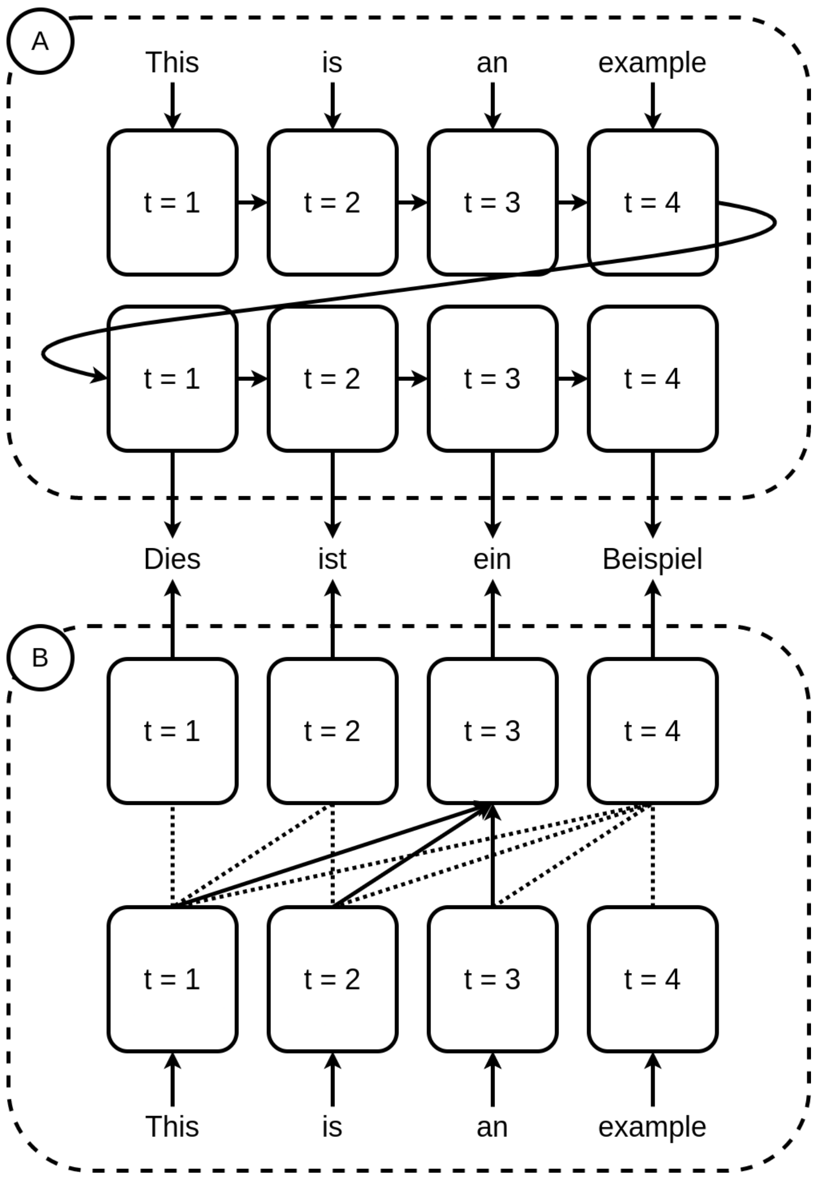

Figure 2.1: (A) An unrolled RNN solving a translation task. Note how the information from the earlier inputs is overloaded with information from the newer inputs. (B) An attention mechanism working on the same inputs.

or token- and sentence- level classification tasks can then be appended to the end of the architecture [73] [21] [58].

Supplementing the encoder is the *decoder* stack for causal language modelling, were we generate sentences auto-regressively, by always predict the next token given the previously predicted ones. Since during training we pass in the whole sentence teaching the model to always predict the next token in the input, also known as *teacher forcing*, we have to make sure to block information flow from later to earlier tokens, i.e. right-to-left. This can be achieved by using a *causal attention mask*, which zeroes out every cell above the attention matrix's diagonal as depicted in fig. 2.2. After multiple such causally masked transformer blocks, appending a linear layer that projects every token representation to the size of the vocabulary, and applying a softmax function, i.e. a classification layer or *language modelling head*, yields the auto-regressive decoder [78] [79] [7].

Combining these two components we come back to the earliest architectures using attention, the *encoder-decoder* used to solve the classical sequence-to-sequence task of machine translation [2] [100]. In this framework the encoder encodes a source sequence $(x_1, \ldots, x_n)$ into a latent representation $(z_1, \ldots, z_n)$, and the decoder uses the latents while auto-regressively generating the target sequence $(y_1, \ldots, y_m)$. In the basic setting, the encoder section of the architecture is structured just as the bidirectional stack of transformer blocks described above. The decoder is also designed as in the previous paragraph, with one exception, namely that between each causally masked self-attention layer and the succeeding feed-forward layer, an additional *cross-attention* layer is inserted which takes the encoders outputs as keys and values to the queries of the previous decoder layer [100] [51] [81] (see fig. 2.2).

Pretraining & Finetuning

Such deep models have great flexibility in modelling language, however with great representational capacity comes the risk of overfitting on small datasets. Thus, a common approach is to *pretrain* large models on large enough datasets, and then *finetune* the model on the particular downstream task [76].

Unfortunately for natural language processing human-annotation is costly, and it is often hard to come by large enough labeled datasets to serve as supervised pretraining signal. There is however an abundance of unlabeled text data available on the web to anyone willing to scrape and collect it in a corpus. A few examples include the BookCorpus [118], a large collection of free novel books written by unpublished authors or OpenWebText [28] which was used to train RoBerta [58], and is tself an open-source recreation of WebText which was used to train GPT-2 [79]. To get an idea of the size, WebText, for example, consists of 40GB of text from 8 million documents scraped from the internet, but an even larger example is the corpus used to train T5 [81], named C4, which consists of 750GB of text derived from the Common Crawl Corpus. Between such corpora the language can often vary significantly, e.g. domain-specific corpora often involve more technical terms while social media texts tends to be more noisy containing spelling errors, abbreviations, and emojis, etc..

To serve as pretraining datasets however a pretraining task, that doesn't rely on annotated examples, has to be derived first. Hence, pretraining for natural language processing typically is done with self-supervised learning tasks, where pseudo-labels, are derived from the data to serve as training signal [69] [78] [21] [51]. A good pretraining task should have high data utilization and be similar to the downstream task.

Two common pretraining tasks for pretraining decoders and encoders are *Causal Language Modelling* (CLM) and *Masked Language Modelling* (MLM) respectively. In Causal Language Modelling (L_{CLM} in Eq. 2.2a) the task is to predict the next token x_i given the preceding tokens $x_{j<i}$, which mirrors the auto-regressive decoding typically employed for text generation, and is therefore well suited for pretraining causal language models [78]. In Masked Language Modelling (L_{MLM} in Eq. 2.2b), out of the N tokens in the sequence, a

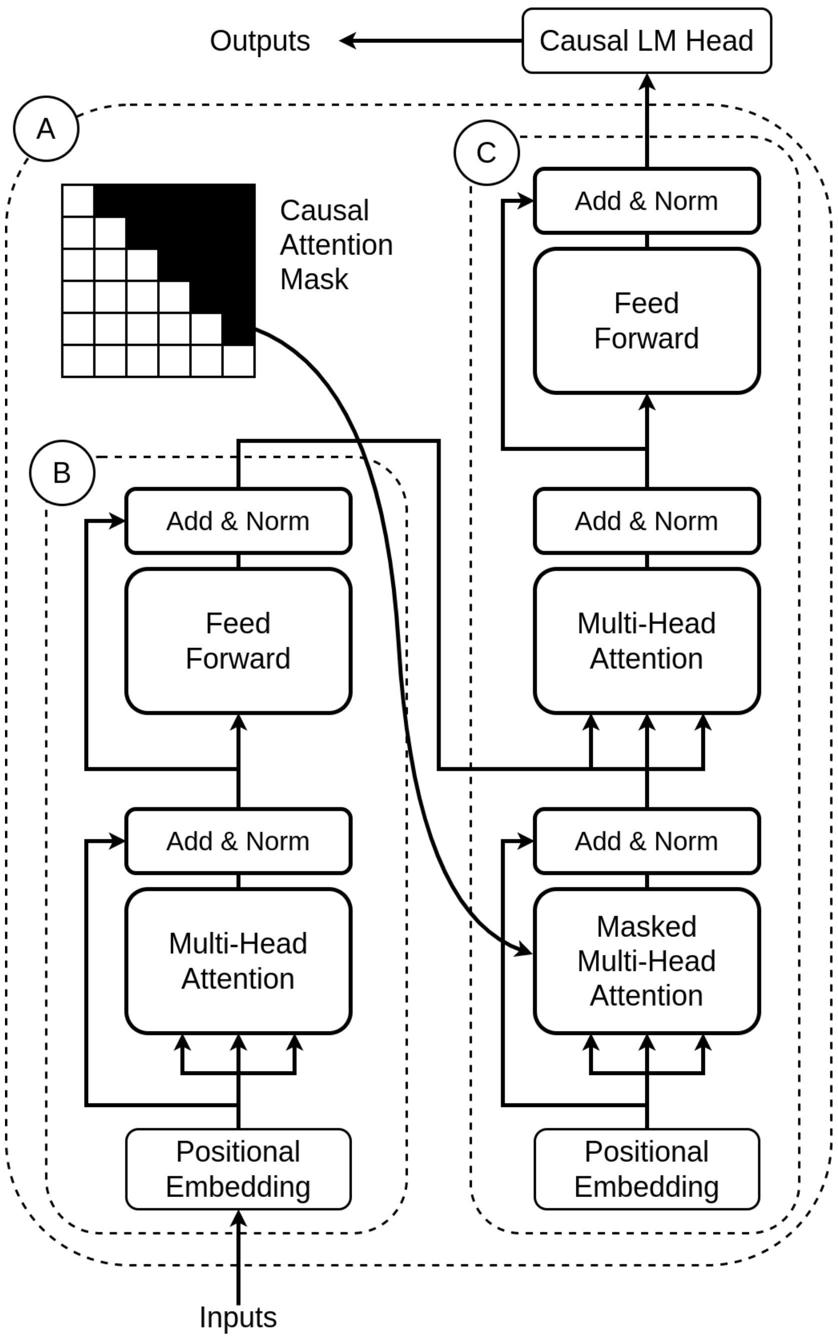

Figure 2.2: (A) The original encoder-decoder transformer. (B) ist the encoder, (C) the decoder. (Adapted from [100])

subset are masked out at random, and the model has to predict the missing tokens from the remaining ones. Masked Language Modelling typically serves as a pretraining task for encoders like BERT [21] or RoBERTa [58], and can be well adapted to token-level classification tasks, like pos-tagging for example, sequence-level classification, like sentiment analysis or sentence entailment, or extractive question answering, where a span of the input has to be marked as containing the answer to a given question.

$$L_{CLM} = -\frac{1}{N}\sum_{i=1}^{N} \log \mathbb{P}(x_i \mid x_{j<i}), \qquad \text{Causal Language Modelling} \tag{2.2a}$$

$$L_{MLM} = -\frac{1}{N}\sum_{i\,:\,x_i \text{ masked}}^{N} \log \mathbb{P}(x_i \mid \{x_j : x_j \neg\text{masked}\}), \quad \text{Masked Language Modelling} \tag{2.2b}$$

Furthermore, there also exist pretraining tasks for encoder-decoder models, such as *Sequence-to-Sequence Denoising* [51], where the input text is corrupted by a noising function and has to be reconstructed by the model. Possible noising functions are for example random masking and deleting or permuting of tokens or spans. These tasks, typically serve as good preparation for downstream tasks like summarization, translation or open-ended question answering [51] [81].

To finetune the resulting models for a downstream task, we then initialize the architecture with the pretrained weights and train the model with a task-specific loss for a few epochs. This will bring all model weights closer to the ideal value for the task [75], including the embedding layer, but predominantly affect higher layers [116] [66]. In this way the initialization with the pretrained model weights, provides adequate regularization for the smaller dataset [24]. Besides warm-starting the architecture as described, these pretrained models can also be used in a zero- or few-shot setting, by designing task specific prompts [59].

2.2 Knowledge Graphs & Graph Neural Nets

In this section we want to discuss how to use knowledge graphs to represent knowledge in a structured fashion, and how to process such networked data with graph neural networks. We also try to motivate such relational inductive biases from a standpoint of generalization.

2.2.1 Representing Knowledge Graphically

Even if our experience of complex systems as compositions of entities and their interactions may not exist in the noumenal realm, [45] it is at least a strong (and arguably useful) human bias [3]. Such a structure also makes it easier to continually learn new knowledge since new concepts can easily be fitted into an existing network of knowledge and efficiently retrieved by traversing the existing categories in such a semantic network [1].

The first step in inducing such a relational structure into a machine learning model, however, is to represent the input in a relational fashion. This means structuring the background knowledge in the form of a graph. In the case, where the target domain is natural language, such graphs are commonly referred to as *Knowledge Graphs* (KG). These are structured representation of facts consisting of entities, representing real-world or abstract objects, and relationships between them, which tend to belong to a predefined and therefore closed set of relationship types A [23]. A typical formalization goes as follows: A knowledge graph is an attributed graph $G = (V, E, A)$ with nodes V representing entities and edges $E \subset V \times A \times V$ representing the relationships between two connected concepts.

For instance (Moskva, instance of, Shipwreck) and (Shipwreck, subclass of, disaster remains), are examples of such triplets. Here “Moskva” refers to a real-world object, “Shipwreck” refers to an abstract concepts, just as "disaster remains" does, and "instance..." and "subclass of" represent their interactions. With this example we can also see, how knowledge graph triplets can be connected to move between different layers of abstraction, in this case the resulting path is “Moskva” $\longrightarrow$ “Shipwreck” $\longrightarrow$ “Disaster Remains”, which connects the concept “Moskva” on the lowest level of abstraction to concept of “Disaster Remains” on the highest.

Interestingly enough, there is new research emerging on *Temporal Knowledge Graphs*, which extend the traditional edge triplets, by introducing an additional temporal dimension. For example Know-Evolve and [99] HyTE [20] proposed frameworks for dynamic representation learning of temporal KG entities. We highlight, such research here since we believe that it aligns nicely with the mentioned analogy to evolving memory in humans.

The process of constructing a knowledge graph depends a lot on the envisioned application, and the source of information, e.g. human collaboration, unstructured text or structured formats, like relational databases, web pages, JSON or XML files [37]. In the case where the graph is to be derived from unstructured text, it is common to construct a so called automatic *Information Extraction (IE)* pipeline. These pipelines also leverage machine learning components more and more, and there is even work on extracting KG triplets from transformer attention matrices [105]. Nevertheless, the most common approach is to use a pipeline consisting of the three main components of *Named Entity Recognition* (NER), *Entity Linking* (EL) and *Relation Extraction* (RE). More specifically, the NER system tags entities in the text, and depending on whether a partial KG already exists or not, the extracted entities then have to be matched to the already existing ones by the EL stage. Finally the RE system can identify short spans of text that describe the relationship between the identified entities [37].

2.2.2 Relational Reasoning with Graph Neural Nets

Coming back to combinatorial generalization, not only do we rely on a relational representation when reasoning, we also tend to compose these familiar concepts to solve novel tasks [3] and generalize to new domains by drawing analogies between our familiar structures and unfamiliar environments [36]. Therefore it has been implied that we need computational models that are just as inherently relational [96]. One possible manifestation of which is the so called *Graph Neural Network* (GNN) [3].

Where the complex relationships and inter-dependencies in graphs represent challenges to other deep learning architectures, that presuppose the data can be meaningfully represented on one or two-dimensional grids, e.g. images and text, Graph Neural Networks are a class of deep learning models equipped to deal with such relational data, by the use of a *Message Passing* mechanism [6].

Message passing here refers to a general framework for graph representation learning. It generally describes a series of operation to repeatedly update information on nodes in a graph. In the context of deep learning this means repeatedly updating the vector representations $h_t^n, n \in |V|$ associated with each node n at time t to the next representation h_{t+1}^n at time $t+1$. The update rule for a single node n with neighborhood $\mathcal{N}(n)$ is

$$h_{t+1}^n = q(h_t^n, \bigotimes_{(n,a,m)\in\mathcal{N}(n)} f_t(h_t^n, a, h_t^m)) \tag{2.3}$$

More specifically (2.3) describes the following steps:

1. Compute a *message* $f_t(h_t^n, k, h_t^{m_j})$ for each neighbor m of the central node. The message can depend on the neighbors vector representation h_t^m, the central node's representation h_t^n, and possibly the edge attribute a connecting them (if edge features exist).

2. Aggregate the messages of all neighbors, with a permutation invariant function $\bigotimes$, e.g. sum, mean, min or max. The invariance is important since we don't want the aggregate to change depending on the order in which we enumerate the messages, since the underlying graph would be the same nonetheless.

3. Compute the new representation h_{t+1}^n from the aggregated messages, and the old representation h_t^n using a (preferably) non-linear activation function q. Then repeat the three steps for all nodes $n \in V$ and all time steps $t \in 1, ..., T$.

Performing message passing for multiple rounds increases the size of the context, that went into the final node representation by one step towards its most distant neighbor each round. This means that after a certain number of rounds the set of nodes that contribute to the the final representation of the center node becomes V [68].

2.3 Information Retrieval

The first stage in a retriever-reader system is the *Information Retrieval* (IR) engine, aimed at identifying a subset of documents from a larger collection to satisfy an information need. This engine can be based on sparse/lexical or dense retrieval mechanisms. While sparse representations, build document vectors containing statistics about every word in its components, dense representations compress the semantic information of the document into a dense, i.e. low-dimension (learned) embedding vector. A good introduction to classical information retrieval can be found in [63].

2.3.1 Sparse Retrieval

There exist four models of the sparse approach, the *boolean, vector space, probabilistic and language model* [117]. In the *boolean model* every document is represented by a binary array of dummy-variables, with entry i storing whether term i occurs in the document. This means that the resulting document embedding has one dimension for every word in the vocabulary. With these arrays it becomes possible to query whether certain logical combinations of terms appear within the document. The document "Bass describes a low-frequency sound" contains the terms “bass” AND NOT “instrument” for example. Such queries can be easily answered with the aforementioned binary vectors by bitwise logical operations between them.

Nevertheless, we usually want retrieved documents to be ranked, which the boolean approach unfortunately does not provide out of the box. We hence turn to *vector space models* where the document vectors are modified so as to contain the TF-IDF weights of the terms. This statistic stands for “term-frequency times inverse document frequency” and intends to reflect how important a word is to a document relative to the other documents in the collection [82]. The documents can then be ranked by computing the cosine similarity (2.4) between the TF-IDF vector of the query document with the document vector.

$$sim(\mathbf{a}, \mathbf{b}) = \cos \angle(\mathbf{a}, \mathbf{b}) = \frac{\mathbf{ab}}{||\mathbf{a}||\,||\mathbf{b}||} \tag{2.4}$$

A different approach from ranking by cosine similarity is to rank documents by their probability of being relevant to the query, as it is done in the probabilistic class of sparse retrieval engines. These generally involve the same statistics, as the vector space models, like term-frequency, document length and inverse document frequency, but cast the problem in terms of probability theory. One of the most successful retrieval systems empirically, BM25 [44], is such a model.

Lastly, the *language modeling* approach to IR builds a probabilistic language model from each document, and ranks the documents based on the probability of the model generating the query [63].

2.3.2 Dense Retrieval

Sparse retrieval however has several issues. First there is the problem of the *lexical gap*, i.e. not knowing about certain words. Moreover, all words have the same distance from each other, since every term has its own dimension, for example synonyms like: "example", "sample" and "exemplar" have representations just as far apart as antonyms. The third weakness of sparse retrieval models is that they treat documents as bags of words, i.e. word order is not preserved. For example, the documents "Oceania started the war against Eurasia" and "Eurasia started the war against Oceania" both have the same sparse representation.

Thus, the idea of *dense retrieval* is to find an embedding $em : \mathcal{D} \to \mathbb{R}^n$, with $\mathcal{D}$ representing the corpus and $n << |\mathcal{V}|$, such that semantically similar words are closer together in the embedding space $em(\mathcal{V})$. To retrieve the top-k documents to a query, one can efficiently compute the k nearest neighbors to $em(query)$, using a vector database like Pinecone[1] for example. With this approach dealing with multiple modalities like text, images, and videos alongside each other becomes possible as well.

A first approach to dense retrieval that doesn't require learning an embedding function, is to simply average the word embeddings from all words contained in the document using Word2Vec [69] or GloVe [72]. This does not however preserve word-order, so another (naive) solution is to use the contextualized word embeddings of a pretrained encoder like BERT, i.e. to average the encoder outputs that result from feeding in the document as input. This happens to perform even worse however, so what is typically employed is a so-called *Bi-Encoder* [83, 46]. These models are trained by encoding pairs of queries and documents separately and comparing the resulting encodings using some similarity based or contrastive loss (e.g. cosine-similarity, triplet loss, etc.) [83, 50, 77].

Since the encoder however doesn't see queries and documents together during training, the resulting embedding spaces can have a bad local and/or global structure. Therefore *Cross-Encoders* that receive the query and documents together, and can therefore model token-level interaction between the two, typically have a better accuracy, albeit being more slow to train and infer [83, 98] due to the quadratic cost in the attention layer becoming prohibitive when encoding query and document together, which further motivates the development of efficient transformers.

Combinations of cross-encoders and bi-encoders have also been proposed, where the cross-encoder is used to mine better examples for the bi-encoder [97], and there also exist a few unsupervised methods, although they still lack behind the supervised approaches [25].

[1]`https://www.pinecone.io/`

3 Related Work

We now want to briefly discuss related works in the area of grounding language models on external knowledge. For a more in-depth literature review we refer the reader to [112, 18, 117, 62]. To tackle this task, many of the architectures proposed follow a retriever-reader framework, where an external knowledge base is connected to a reading comprehension component, which is then connected to a generative language model. Generally speaking, we observe that there are two main choices for the external knowledge source here: Unstructured document collections and knowledge graphs. However, grounding on additional modalities, like tables and images is also a problem of interest in the knowledge grounding domain [11, 114]. Moreover, we can observe that how the external knowledge is represented influences the design of the reader component significantly, which is why we divide the existing works along this line.

As an aside, there also exist approaches for internal i.e. parametric grounding of language models, thereby deviating from the retriever-reader paradigm. For example [31] internalize the knowledge from Atomic [87] and ConceptNet [92] by finetuning GPT-2 on textual representations of knowledge triplets therein. However, to solve the stale-memorization problem eluded to in the introduction, the models knowledge needs to be easily updateable, which we argued is difficult once injected into the models' parameters [39]. The same goes for any approach that only scales up language models further [7, 80, 91, 13], without addressing the problem of continual knowledge learning. On this topic, we want to note that some promising methods to resolve that issue are emerging [8, 39].

3.1 Document-Grounded NLG

Starting with document-grounded NLG, where the knowledge base consists of a collection of documents, like passages from Wikipedia [22], search engine results [70, 48] or online news [34] among others. This setting allows for retrievers based on semantic similarity, like sparse [22] or dense document models [52], active web search [70, 48], by generating natural language queries or reinforcement learning in a browser-like environment or leveraging relationships between retrieved documents for re-ranking with graph neural nets. [111]

The reader component is then often implemented via a sequence-to-sequence model. However many models, especially in the domain of question-answering [117], only extract spans of text from the knowledge base instead of generating new natural and fluent responses. This makes it harder to synthesize knowledge from multiple sources however, so we will focus on works deploying generative readers.

Typical applications for document-grounded natural language generation are question-answering [117], dialogue modelling [62] and summarization [9, 106, 54]. An example in the domain of question answering is S-Net [94], which uses a bidirectional gated recurent unit as decoder that receives the question and the outputs of an extractive reader, i.e. the relevant spans of text, likely to contain the answer. More recent question-answering models though, have started to adopt pretrained sequence-to-sequence models, like BART and T5, as readers, as in RAG [52] and the Fusion-in-Decoder (FID) [38].

In dialogue modelling we have for examples MTask [27], that uses an encoder based on a Memory Network [107], which uses an associative memory for retrieving encoded facts that might be relevant for the conversation. Other examples include [22, 65, 47, 115], that all encode a set of passages and then decode into natural language using recurrent neural networks or transformers, conditioned on the dialogue history as well. A final noteworthy example is BlenderBot 2.0[1] that combines active web search [48] with a dense retrieval over the dialogue history [108].

3.2 Graph-Grounded NLG

We now turn to the second approach to knowledge grounded NLG, the knowledge-graph-augmented generation models. Many architectures in this vein will choose either ConceptNet [92] and/or Atomic [87] as their external knowledge graph. Some works also use knowledge graphs custom built for their target domain [58, 106, 12]. Once acquired, to retrieve from a knowledge graph, the procedure is to first identify entities in the query and match them to the entities in the graph in a step referred to as entity linking. Then, a sub-graph is spanned composed of the k-hop neighbors of the identified concepts. Increasing k increases the amount of information retrieved, but too big of a k typically leads to irrelevant information being retrieved alongside the relevant concepts, which will lead to noisy inputs to the model and high computational cost. Therefore a common solution is to have a reasonably high k of about 3-5 but to then deploy an intermediate re-ranking stage before feeding the sub-graph to the model [113, 109, 43]. These ranking schemes typically involve assigning scores to the retrieved nodes using an attention-mechanism and possibly deactivating those, that fall beneath a predefined threshold.

The retrieved and possibly weighted sub-graphs will then typically be encoded into a latent representation. This can either be done by using pretrained entity embeddings like ConceptNet Numberbatch [92], by training custom word embeddings using a network representation learning framework like TransE [4] as was done in [57], or by a learned embedding layer followed by a graph neural network [42, 12, 106, 53]. These representations will then have to be injected into a decoder. For this, there are three possible choices as documented by [17]. Either the concepts are concatenated to the input of the decoder [115, 58, 42], injected in the middle of the architecture, typically by transforming the decoder hidden representations by some attention-weighted combination of the encoder outputs [113, 12, 59], or they are incorporated at the output of the decoder [53, 113, 42], which typically involves a copy or pointing mechanism [30, 89].

[1]https://parl.ai/projects/blenderbot2/

4 Methodology

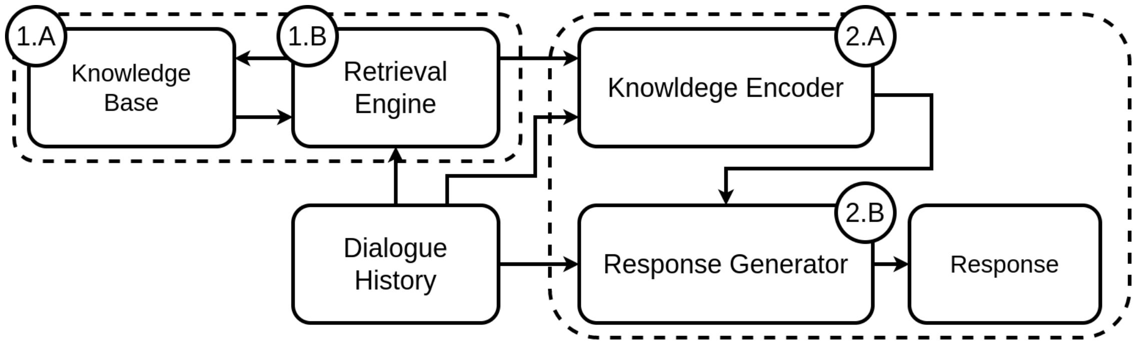

Figure 4.1: The *Retriever-Reader Blueprint*: A knowledge base is queried with the dialogue history via the retrieval engine, then a knowledge encoder prepares the inputs for the response generator.

We now introduce the *Retriever-Reader Blueprint*, a general framework for injecting external knowledge into neural dialogue models. As the name suggest, the system has two components: the *retriever* (1) and the *reader* (2). The retriever is responsible for supplying relevant knowledge, and the reader is required to synthesize the retrieved knowledge into a coherent response.

A general retriever itself consists of two parts: the *knowledge base* (1.A) and the *retrieval engine* (1.B). The knowledge base $\Gamma = \{s_1, \dots, s_n\}$ is made up of many individual sources. These could be, for example, texts, images, tables or other modalities, and the collection could have some additional structure, like links between documents as in scientific publications and their citation graphs or tweets and their quote-retweets. The retrieval engine $\text{Ret} : \mathcal{V}^* \to \Gamma^k$ takes the dialogue history $d = (w_1, \dots, w_m) \in \mathcal{V}^*$ and retrieves the k most relevant sources $s = (s_1, \dots, s_k) \in \Gamma^k$ from the knowledge base (Eq. 4.3).

The reader consists of two subsystems as well: the *knowledge encoder* (2.A) and the *response generator* (2.B). The encoder $\text{Enc} : \mathcal{V}^* \times \Gamma^k \to \mathbb{R}^d$ takes the relevant sources s, and possibly the dialog history d and encodes both into a latent representation $\mathbf{z} = (z_1, \dots, z_e) \in \mathbb{R}^{e \times d}$ usable by the response generator (Eq. 4.2). How this component is implemented will heavily depend on the retriever component, e.g. a retriever that provides knowledge in a networked format typically implies a graph neural network for the encoder, whereas a sequence model is usually better suited to deal with documents. Finally, given $\mathbf{z}$ and the additional context d from the dialogue, the response generator $\text{Gen} : \mathcal{V}^* \times \mathbb{R}^{e \times d} \to \mathcal{V}^*$, that is typically realized by a generative language model, gives the response r in natural language (Eq. 4.1)

$$r = \text{Gen}(d, \mathbf{z}) \quad (4.1)$$
$$\mathbf{z} = \text{Enc}(d, s) \quad (4.2)$$
$$s = \text{Ret}(d, \Gamma) \quad (4.3)$$

4.1 Retrieval Engine

Both our architectures adapt the *Retriever-Reader Blueprint* to the Wizard of Wikipedia task explained in Chapter 5. Therefore, our retriever's external knowledge base is a collection of 1365 crowd-sourced Wikipedia articles. These articles cover a diverse range of topics such as commuting, Gouda cheese, music festivals, podcasts, bowling and Arnold Schwarzenegger [22].

Connected to this collection is a sparse retrieval engine based on [10], that is able to retrieve the first paragraphs of the seven articles most relevant to the last two turns of dialogue. It consists of an inverted index lookup with a TF-IDF based vector model, and takes into account bi-gram features. While this part of the model is in principle learnable, we have used the passages collected with the original dataset[1], in order to be able to better compare our models to the chosen baselines.

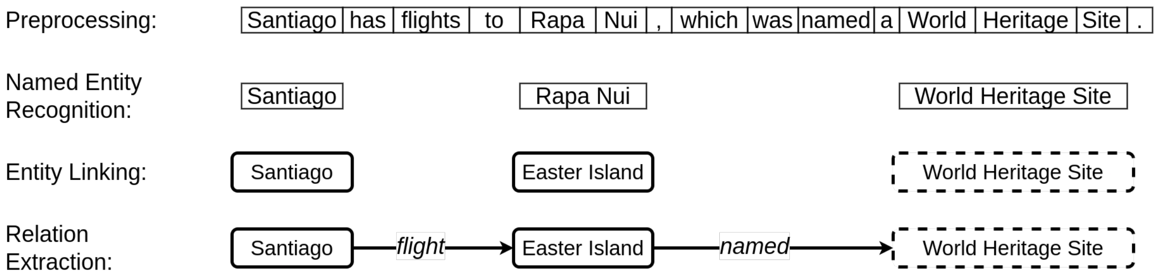

Figure 4.2: The information extraction pipeline used in this work. Dotted lines represent concepts not found in the target ontology. Example and visualization adapted from [37].

Furthermore, while the text-based encoder can directly work on the retrieved passages, for the graph-based variant we additionally have to transform the text into a knowledge graph. To this end we built the information extraction pipeline depicted in fig. 4.2. The first stage is to resolve co-references in the text. This means that we identify all expressions, typically pronouns, that refer to the same entity. To accomplish this step we leverage the nerualcoref project[2] which is based on [14]. Next, we extract all entities that exist in our target ontolgy, in this case WikiData[3], the free and collaborative knowledge base, derived from Wikipedia. In this ontology each Wikipedia article is represented by a uniquely identifiable item, with uniquely identifiable properties connecting them, often

[1]`http://parl.ai/downloads/wizard_of_wikipedia/wizard_of_wikipedia.tgz`
[2]`https://github.com/huggingface/neuralcoref`
[3]`https://www.wikidata.org/wiki/Wikidata:Main_Page`

derived from the hyperlinks between articles. Linking entities identified in text to WikiData items is also sometimes referred to as Wikification, and we have used the Wikifier API [5] for this step. The final stage of the information extraction procedure is to infer the relationships between the concepts mentioned in the passage. To this end we use the OpenNRE project [32].

4.2 Text Encoder

For the text-grounded model, we deploy a pretrained BERT model as our encoder [21], this means the architecture is a stack of $L_{encoder}$ consecutive transformer encoder blocks, each one using gelu activations and a hidden size of $d_{encoder}$. The final latent representation $\mathbf{z}$ is derived as follows:

$$\mathbf{z} = W_{proj}\mathbf{h}^{L_{encoder}} \in \mathbb{R}^{k \times d} \tag{4.4}$$

$$\mathbf{h}^{l} = \text{BERT-Block}(\mathbf{h}^{l-1}) \qquad \forall l \in \{1, \dots, L_{encoder}\}, \tag{4.5}$$

$$\mathbf{h}^{0} = sW_e + W_p \tag{4.6}$$

After the final layer $\mathbf{h}^{L_{encoder}}$ we project the encoder outputs to the hidden dimension of the generator with a linear projection W_{proj}. W_e and W_p refer to embedding and postion encoding matrices.

BERT's vocabulary was obtained by a WordPiece tokenization scheme and has a total size of 30,000 tokens [21]. The model was pretrained with masked language modelling and next sentence prediction on BooksCorpus [118] and English Wikipedia, which contain around 800M and 2,500M words respectively.

More specifically, we initialized our instance of the base model with the checkpoint "bert-base-cased" from the Huggingface Hub[4], which implies $L_{encoder} = 12$, $d_{encoder} = 718$, #Heads = 16 and #Parameters = $110M$. To obtain the checkpoint the authors had to train on 16 TPU chips for 4 days in total, with a batch size of 128,000 tokens/batch.

4.3 Graph Encoder

For the graph-based variant, our encoder takes in a knowledge graph. In order to derive vector-valued node and edge features for the GNN, we first tokenize the extracted spans of text corresponding to the entities and relationships in the KG using the decoder's tokenizer. We then share an embedding layer $W_e^{generator}$ (without the positional encoding) with the decoder. This possibly leads to multiple tokens per node or edge, but since we require one feature vector for each node and edge, we average all embedding vectors corresponding to one node or edge. This has the added benefit of aligning the representations of the encoder with the decoder, thereby simplifies the task of the encoder (see section 5.3.1).

[4]`https://huggingface.co/bert-base-cased`

Our graph-encoder is based on a $L_{encoder}$-layer graph attention (GAT) network [101]. The GAT operation is a message passing variant that enables each node to attend to it's neighbors with differing intensity. Since we also have edge features, the embeddings of the extracted relationships, to consider, the node update function becomes:

$$\mathbf{z} = W_{proj}\mathbf{h}_m^L \in \mathbb{R}^{|V|\times d} \tag{4.7}$$

$$\mathbf{h}_m^l = a_{m,m} W\mathbf{h}_m^{l-1} + \sum_{n\in\mathcal{N}(m)} \alpha_{m,n} W\mathbf{h}_m^{l-1}, \qquad \forall m \in V, \forall l \in \{1,\dots,L_{encoder}\}, \tag{4.8}$$

$$\alpha_{m,n} = \frac{\exp(\text{LeakyReLU}(\mathbf{a}^T[W\mathbf{h}_m^{t-1}||W\mathbf{h}_n^{l-1}||W_e\mathbf{e}_{m,n}])}{\sum_{k\in\mathcal{N}(m)\cup m}\exp(\text{LeakyReLU}(\mathbf{a}^T[W\mathbf{h}_m^{l-1}||W\mathbf{h}_k^{l-1}||W_e\mathbf{e}_{m,k}])} \tag{4.9}$$

$$h^0 = sW_e^{generator} \tag{4.10}$$

Here W is a weight matrix, $\mathbf{a}$ is a weight vector and $\mathbf{e}_{m,n}$ represents the edge feature on the edge connecting nodes m and n. $\mathbf{h}_t^m$ signifies the vector representation of node m at layer t, finally $\alpha_{m,n}$ is the attention coefficient between node m and it's neighbor n.

We deploy two layers of this mechanism ($L_{encoder} = 2$) and use a hidden dimension of $d_{encoder} = 718$ throughout the network. Afterwards we project the node features to the hidden dimension of the decoder.

4.4 Response Generator

The response generator, is the same for both encoder variants, namely a modified version of a pretrained DialoGPT [113]. Arcitecture-wise DialoGPT consists of a stack of $L_{generator}$ consecutive transformer decoder blocks, where each attention-layer is equipped with a causal attention mask. After the $L_{generator}$ layers follows a causal language modelling head. Furthermore, to connect the decoder to the encoder, we insert randomly initialized cross-attention layers between the masked self-attention and feed-forward layers in each block. Furthermore warm-starting encoder-decoder architectures in this way was shown to be effective in [85].

$$o = \text{softmax}(\mathbf{h}^{L_{generator}} W_e) \tag{4.11}$$

$$\mathbf{h}^l = \text{ModifiedDecoderBlock}(\mathbf{h}^{l-1}, \mathbf{z}) \qquad \forall l \in \{1,\dots,L_{generator}\} \tag{4.12}$$

$$\mathbf{h}^0 = dW_e + W_p \tag{4.13}$$

The decoder uses byte-pair tokenization for the vocabulary. The model was originally pretrained on a dataset of approximately 147M dialogues extracted from comment chains scraped from Reddit. The training task was designed as causal language modelling on the concatenation of all dialog turns, and the authors used a modified initialization scheme that accounts for depth [113].

We use the checkpoint "microsoft/DialoGPT-medium" from the Huggingface Hub, corresponding to a stack of 23 transformer decoder blocks, i.e. $L_{generator} = 23$, a hidden dimension d of 1024 and a total of 345M parameters, while the cross-attention layers add

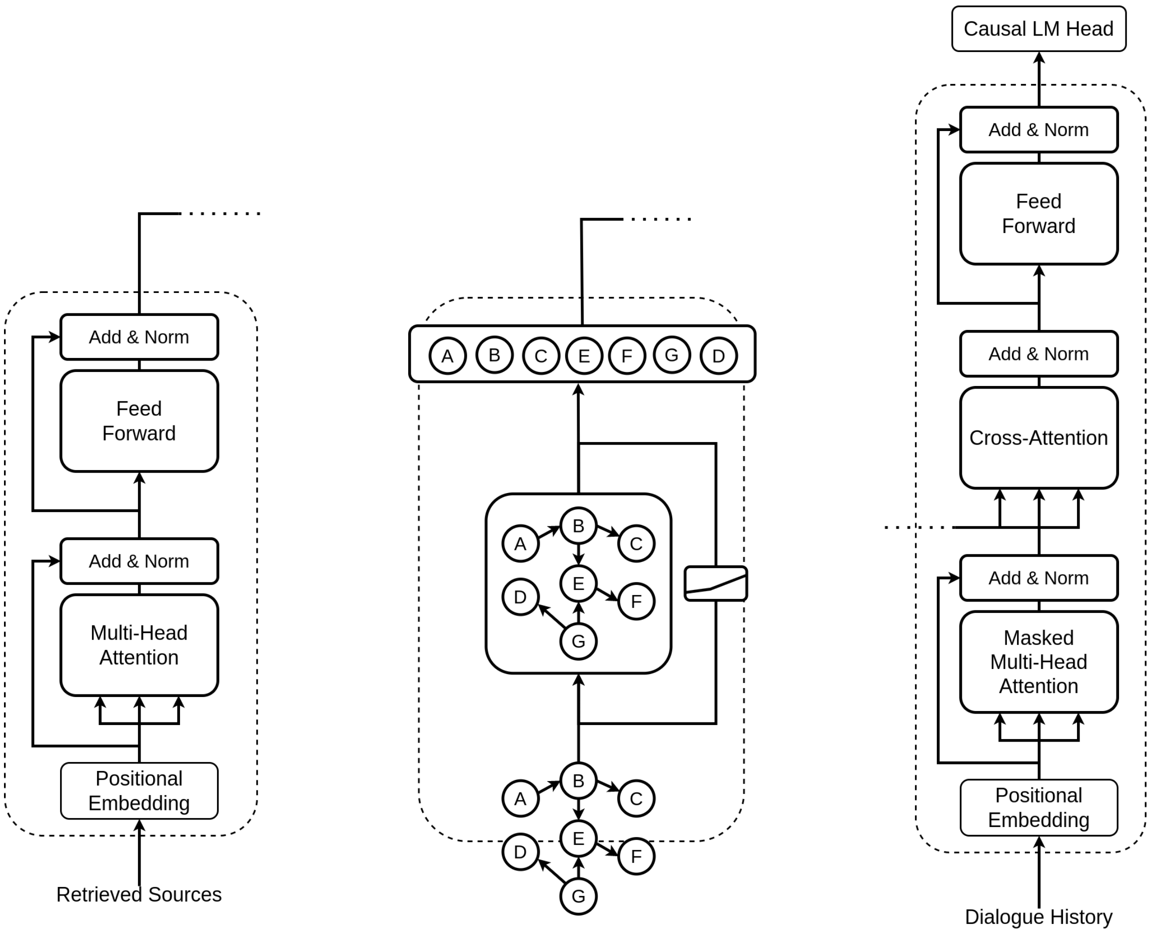

Figure 4.3: The three neural components used in our model. (Left) the text-based encoder. (Middle) the Graph Attention Network. (Right) DialoGPT with the added cross-attention layers.

an additional 110M. For our finetuning task the decoder receives the last utterance of the dialogue.

5 Experiments

5.1 Dataset and Metrics

5.1.1 Dataset

We compare the two architectures on the Wizard of Wikipedia dataset, a large collection of 22,311 crowd-sourced dialogues, each of which is directly grounded on knowledge retrieved from Wikipedia. In each dialog, two participants engage in chit-chat about a topic that is chosen in advance. One of the two, plays the role of a knowledgeable expert, referred to as the wizard, while the other talks to the wizard freely [22].

For every turn the wizard selects one sentence from a set of paragraphs retrieved by the retrieval system described in section 4.1. So overall the conversation flow is as follows:

1. The two participants receive a topic chosen from a predefined pool. Our model ignores this information.
2. The IR engine retrieves the first paragraph of the seven most relevant Wikipedia articles relating to the topic and the previous two turns of dialogue.
3. The wizard chooses a single relevant sentence from the retrieved passages, and synthesizes a response.

These steps are repeated until the conversation ends. Given this setup we have constructed the model inputs so that every sample consists of the last utterance, which is passed as a prompt to the generator. The sentence that was selected by the wizard, or the derived knowledge graph thereof, is passed to the encoder. An exemplary conversation is depicted in Fig. 5.1

5.1.2 Metrics

Following the original paper introducing the dataset, we evaluate our model using perplexity and the unigram F1 scores between the predicted and generated responses [22]. In addition to these, we use bag-of-words embedding-based metrics as suggested by [115]. This is because when only comparing word-overlap between predicted and target sentences, the resulting metrics correlate only very weakly with human judgements [56]. Embedding-based metrics instead combine the word-embeddings of prediction and target in some fashion to derive an embedding of the sentence, and these aggregates are then compared.

The additional embedding-based metrics we employ are a *Greedy Matching* and the *Embedding Average* and *Vector Extrema*. The first of which is computed by going through

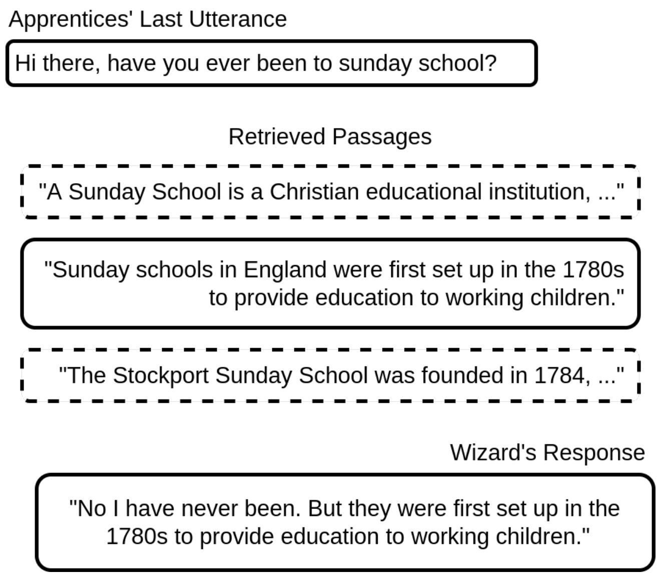
Apprentices' Last Utterance
Hi there, have you ever been to sunday school?
Retrieved Passages
"A Sunday School is a Christian educational institution, ..."
"Sunday schools in England were first set up in the 1780s to provide education to working children."
"The Stockport Sunday School was founded in 1784, ..."
Wizard's Response
"No I have never been. But they were first set up in the 1780s to provide education to working children."

Figure 5.1: An exemplary conversation from the Wizard Of Wikipedia validation set. The retrieved passage chosen by the wizard is in solid lines.

the target sentences' word embeddings and greedily matching them to the predictions' word embeddings [86]. This approach favours prediction-target pairs where there is overlap between key words [56]. The *embedding average* computes the sentence embeddings by averaging all word vectors in the sentences. The results are compared using cosine-similarity. Finally the *vector extrema* construct the sentence embedding dimension by dimension, such that, for each component of the sentence vector the maximum value over all word embeddings in the sentence is chosen. The alignment is again compared by cosine-similarity. This may help to reduce the influence of common words [56].

5.2 Experimental Setup

To implement the two models, we depend on the PyTorch[1], Huggingface Transformers[2] and PyTorch Geometric[3] Libraries. We also implement some metrics with the help of Torchmetrics[4] and GenSim[5]. Furthermore we put our data under version control using DVC[6], and manage all model configurations with Hydra[7]. For experiment tracking we use Weights & Biases[8]. We train each model on an NVIDIA Titan RTX, on a batch size of 16 sequences per batch and use AdamW [60], with a learning rate of 5×10^{-5}, an Adam epsilon of 10^{-8} and no weight decay or warmup. During evaluation we decode a whole

[1]https://pytorch.org/
[2]https://huggingface.co/docs/transformers/index
[3]https://pytorch-geometric.readthedocs.io/en/latest/
[4]https://torchmetrics.readthedocs.io/en/latest/
[5]https://radimrehurek.com/gensim/
[6]https://dvc.org/
[7]https://hydra.cc/
[8]https://wandb.ai/home

batch of prompts (padded on the left) and encoder inputs using a simple greedy decoding, with a hard cutoff after 1000 tokens.

We build all knowledge graphs for the graph-based model upfront as shown in Algorithm 1.

Algorithm 1: Knowledge Graph Construction

```
Initialize:
    turns ← {turn | turn ∈ Dialog, Dialog ∈ Dataset}
Do in parallel:
    processTurns ← subset(turns, processId)
    result ← ∅
    for (apprentice, wizard, checkedSentence) ∈ processTurns do
        checkedSentence ← resolveCoreferences(checkedSentence)
        entities ← wikify(checkedSentence)
        relations ← ∅
        for (source, target) ∈ Permutations(entities) do
            relations ← relations ∪ extractRelations(source, target)
        result ← result ∪ (apprentice, wizard, entities, relations)
return results
```

5.3 Results

We will now discuss the results obtained on the wizard of wikipedia task. First, we answer whether the document or graph-based approach turned out to be superior, for our application. Then, we will compare our models to a set of baselines.

5.3.1 Comparing Graph-to-Seq and Seq-to-Seq

Tab. 5.1 shows that the two model variants perform comparably. Especially, we do not see an out-performance arising from the relational representation of the background knowledge. Furthermore, we compare our two models to a DialoGPT-only variant, that has no access to the background knowledge, which already compares favorably to the two models. Next, we go into possible reasons for this and we compare the two variants on aspects relating to the ease of training.

Knowledge Graph Construction

Regarding ease of implementation, the first thing we might note is that the construction of knowledge graphs is accompanied by considerable engineering effort, that is not required when working on a document collection by itself. For the small passages we transformed into knowledge graphs, the transformation of the training split took 2 full days, even with 50 CPUs working in parallel. If the background passages became larger, our approach would almost certainly become infeasible, due to the factorial scaling in Algorithm 1 arising from the loop over all permutations of identified entities. We argue that therefore, if

Model	PPL	F1	Avg.	Greedy	Extrema
DialoGPT	16.0	**17.9**	–	–	–
BERT2DialoGPT	**14.3**	16.0	**97.7**	67.9	**80.8**
GAT2DialoGPT	17.4	16.7	**97.7**	68.0	80.7

Table 5.1: Evaluation Results on the Wizard of Wikipedia seen-topics split. GAT2DialoGPT and BERT2DialoGPT, refer to the graph- and text-based model variants respectively. DialoGPT refers to the generator only-variant that has no access to the background knowledge. The scores of which are taken from [115].

the background knowledge is not inherently graphical, e.g. citation graphs, the relative ease of compiling document collections favors the sequence-to-sequence approach. Especially since the transformer-based architectures already perform well on unstructured text.

Effects of the Relational Representation of Knowledge

As we noted, we don't observe any benefit from representing the knowledge inherent in the text in a graphical fashion. We assume that this is since bidirectional attention can already be interpreted as receiving a fully connected graph between all tokens in a given input text. The model can then decide itself which relationships are important for the task at hand, instead of humans having to decide on that upfront. The hypothesis is that the capacity for relational reasoning in transformers is already strong enough. This is also underpinned by the fact that many of the best performing entity recognition and especially relation extraction models, are based on transformers [88, 116], and that one can even extract knowledge graphs from the attention matrices of bidirectional transformer models [105].

Semantic Misalignment

Finally, as we can see the model without access to the background knowledge (DialoGPT) performs just as well as our implementations that read the retrieved passages. We hypothesis that this is since the encoder and decoder hidden representations are not semantically aligned, a problem which has been observed in other works as well [53]. A possible intuition is that, aligning these representations requires the model to go through a high-loss region, whereas simply ignoring the encoder outputs is easier to learn. This hypothesis is also supported by Fig. 5.2, which shows that the highest attention scores tend to lie on the [CLS] and [SEP] tokens of the encoder inputs. A second supporting fact is that, when replacing the encoder and decoder with a pretrained sequence-to-sequence model, BART in this case, the F1 Score rises significantly and the attention tends to cluster around the matrix diagonal (see Fig. 5.2). This is another reason to prefer the text-based knowledge representation, since doing so allows the use of pretrained sequence-to-sequence models which do not suffer from the misalignment problem.

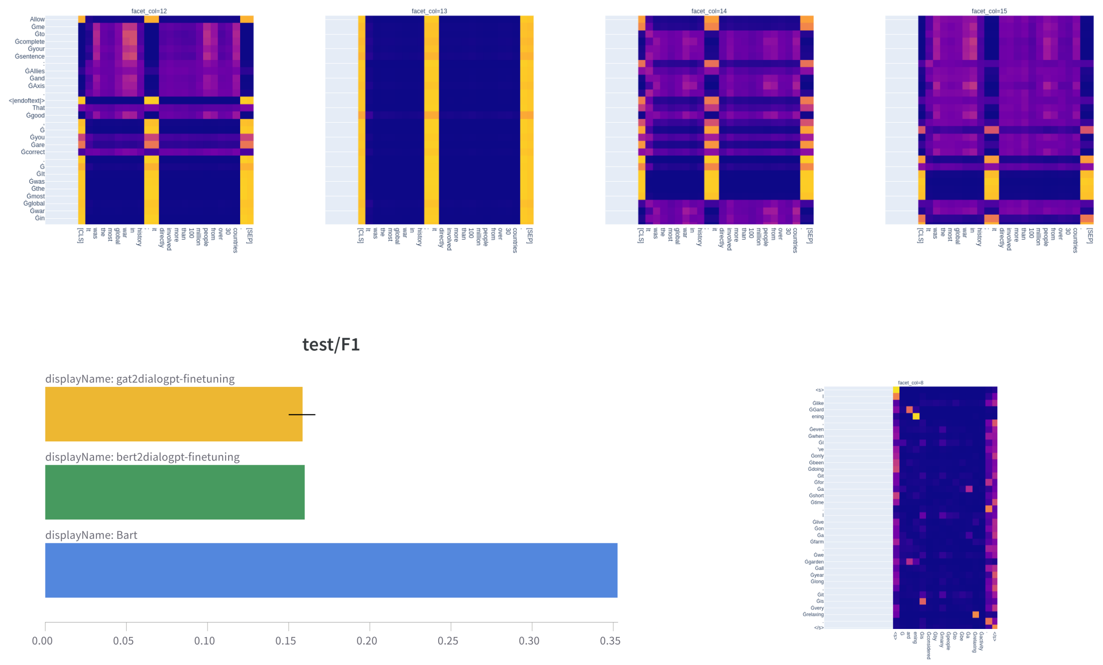

Figure 5.2: Visualizations of the cross attention matrices of BERT2DialoGPT (top), taken from the eleventh layer, heads twelve to fifteen. (Bottom left) F1 score reached by the BART-variant next to the scores of our two variants. (Bottom right) an attention matrix from the BART-variant, taken from the first layer and the ninth attention head.

5.3.2 Comparing to the State-of-the-Art

Next, we compare our models to state-of-the-art methods for knowledge grounded dialogue. As baselines we have chosen several works from the literature for which, performance scores on the wizard of wikipedia dataset were published in [115]. First, we have the Transformer Memory Network (TMN) by [22], which was introduced along with the dataset. This model first encodes the dialogue history and each retrieved passage independently with a transformer encoder. A standard dot-product attention between the encoded dialogue history and passages, then ranks the passages. The passage that received the highest attention is then concatenated to the encoding of the dialogue, and passed to a transformer decoder to generate the final sentence. Next, we compare to the Incremental Transformer with Deliberation Decoder (ITDD) by [55]. This model also encodes the retrieved passages independently with a transformer encoder, but instead of selecting one candidate, incorporates them all into one latent context one at a time. This is done by cross-attending between the current context and the candidate representation. This context is then used to condition a decoder alongside the utterance, and the result is in turn used to condition a second decoder alongside the candidate encodings. Our third baseline is the Sequential Knowledge Transformer (SKT) introduced in [47]. In contrast to the previous methods, this approach encodes all previous turns and candidates using a GRU network. In addition they use a probabilistic sequential latent variable model to select from all previous candidates before conditioning a transformer decoder with an added copy mechanism. Our final baseline is KnowledGPT [115], the only baseline that also uses a pretrained decoder transformer. The dialogue context and background knowledge is encoded with BERT$_{base}$, then a knowledge selector builds up a text prompt for GPT-2.

Model	PPL	F1	Avg.	Greedy	Extrema
TMN	66.5	15.9	84.4	42.7	65.8
ITDD	17.8	16.2	84.1	42.5	65.4
SKT	52.0	19.3	84.6	44.0	66.5
KnowledGPT	19.2	**22.0**	87.2	46.3	68.2
BERT2DialoGPT	**14.3**	16.0	**97.7**	67.9	**80.8**
GAT2DialoGPT	17.4	16.7	**97.7**	68.0	80.7

Table 5.2: Results of the selected baselines next to ours. These are the scores obtained on the test split that also has topics already seen during training. The scores for the baselines are taken from [115]. Best scores are in bold letters.

An important thing to note is that, in contrast to all the baseline models, we do not select from all the candidate passages retrieved by the retrieval engine, only the one sentence selected by the mechanical turker, since a knowledge selection step might have been a further confounding variable in our comparison. This has advantages and disadvantages however, on the one hand our model doesn't have to select from the candidates making the task easier, on the other hand almost all of the baselines here incorporate more knowledge into the generation than just the checked sentence.

6 Discussion

6.1 Conclusion

To summarize, in this work we compared the performances of different approaches to grounding large language model on externally retrieved text, and dialogues from the wizard of wikipedia task. More precisely, we evaluated warm-started transformer sequence-to-sequeuence models working on the retrieved texts directly against GNNs working on knowledge graphs derived from these texts.

The results suggest that, architecture-wise while it may be required to work with GNNs if the background knowledge is best represented in graphical form, i.e. citation graphs, chemistry data and generally networked data, it is hard to justify the use of knowledge graphs when transformer-based language models already perform well on text data, especially since it has been suggested that such models have tacit knowledge that is good enough to extract knowledge graphs from them [105]. Adding to that, it is often easier to compile a document collection to serve as background knowledge for a chosen domain than it is to construct an ontology. We also observe, that it is non-trivial to bridge the semantic gap between foreign, as in separately pretrained, encoders and decoders with finetuning alone. It is therefore advisable to, if possible, use jointly pretrained encoders and decoders.

We further acknowledge that our comparison might be less meaningful, since we weren't able to adequately bridge the semantic gap between the encoders and decoders. A possible approach for this would be to derive a multi-phasic training scheme in which parts of the encoder and decoder parameters are frozen at certain points in time. Another approach taken by [53] would be to add auxiliary losses that penalize encoder outputs that deviate to far from valid decoder representations. To this end a more thorough optimization of our architectures would be required. We also note that the way in which the knowledge graphs were constructed can significantly impact the models overall performance, but as we stated earlier, this can also be seen as an argument against using knowledge graphs in the first place.

6.2 Future Work

We now end this discussion, by highlighting some promising future research directions in the area of knowledge grounded dialogue and open question answering.

Conversational Search

As we already discussed in the background section, neural networks can learn rich semantic representations of items in a knowledge base to help in the retrieval process, but as we have also seen deep learning can do more. For example understanding a dialog context and synthesizing multiple documents into one coherent text. Conversational search is a new paradigm that aims to improve search in situations where the information need of a user is difficult to express in a single query.

Say, for example that you want to learn about the latest progress in neural conversational search. How would you go about finding this information? If you are lucky, you might find a related survey [41], otherwise the process will likely start with formulating multiple alternative queries to a search engine, a lot of open tabs in your browser, and you having to figure out how these pieces of knowledge fit together. The aim of neural conversational search is to integrate this entire process, by formulating alternative queries [15], responding to questions or feedback about the recommendations so far [84], and automatically synthesizing a coherent narrative from multiple sources [67]. Combining this with recent advances in multi-modal architectures, the need arises for databases storing representations for different modalities in a unified way, so that models can base their knowledge on multiple modalities, promising approaches in this direction include for example vector databases such as Pinecone[1].

[1] `https://www.pinecone.io/`

Bibliography

[1] Alan D. Baddeley, Michael W. Eysenck, and Mike Anderson. *Memory*. 2. Psychology Press, 2014.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. *Neural Machine Translation by Jointly Learning to Align and Translate.* 2014. DOI: `10.48550/ARXIV.1409.0473`. URL: `https://arxiv.org/abs/1409.0473`.

[3] Peter W. Battaglia et al. "Relational inductive biases, deep learning, and graph networks". In: *CoRR* abs/1806.01261 (2018). arXiv: `1806.01261`. URL: `http://arxiv.org/abs/1806.01261`.

[4] Antoine Bordes et al. "Translating Embeddings for Modeling Multi-Relational Data". In: *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2.* NIPS'13. Lake Tahoe, Nevada: Curran Associates Inc., 2013, pp. 2787–2795.

[5] Janez Brank, Gregor Leban, and Marko Grobelnik. "Annotating documents with relevant Wikipedia concepts". In: 2017.

[6] Michael M. Bronstein et al. "Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges". In: *CoRR* abs/2104.13478 (2021). arXiv: `2104.13478`. URL: `https://arxiv.org/abs/2104.13478`.

[7] Tom B. Brown et al. "Language Models are Few-Shot Learners". In: *CoRR* abs/2005.14165 (2020). arXiv: `2005.14165`. URL: `https://arxiv.org/abs/2005.14165`.

[8] Nicola De Cao, Wilker Aziz, and Ivan Titov. "Editing Factual Knowledge in Language Models". In: *CoRR* abs/2104.08164 (2021). arXiv: `2104.08164`. URL: `https://arxiv.org/abs/2104.08164`.

[9] Ziqiang Cao et al. "Retrieve, Rerank and Rewrite: Soft Template Based Neural Summarization". In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers).* Melbourne, Australia: Association for Computational Linguistics, July 2018, pp. 152–161. DOI: `10.18653/v1/P18-1015`. URL: `https://aclanthology.org/P18-1015`.

[10] Danqi Chen et al. "Reading Wikipedia to Answer Open-Domain Questions". In: *CoRR* abs/1704.00051 (2017). arXiv: `1704.00051`. URL: `http://arxiv.org/abs/1704.00051`.

[11] Wenhu Chen et al. "HybridQA: A Dataset of Multi-Hop Question Answering over Tabular and Textual Data". In: *CoRR* abs/2004.07347 (2020). arXiv: `2004.07347`. URL: `https://arxiv.org/abs/2004.07347`.

[12] Liying Cheng et al. “Knowledge Graph Empowered Entity Description Generation”. In: *CoRR* abs/2004.14813 (2020). arXiv: 2004.14813. URL: https://arxiv.org/abs/2004.14813.

[13] Aakanksha Chowdhery et al. *PaLM: Scaling Language Modeling with Pathways.* 2022. DOI: 10.48550/ARXIV.2204.02311. URL: https://arxiv.org/abs/2204.02311.

[14] Kevin Clark and Christopher D. Manning. “Deep Reinforcement Learning for Mention-Ranking Coreference Models”. In: *CoRR* abs/1609.08667 (2016). arXiv: 1609.08667. URL: http://arxiv.org/abs/1609.08667.

[15] Vincent Claveau. “Query expansion with artificially generated texts”. In: *CoRR* abs/2012.08787 (2020). arXiv: 2012.08787. URL: https://arxiv.org/abs/2012.08787.

[16] Jasmine Collins, Jascha Sohl-Dickstein, and David Sussillo. *Capacity and Trainability in Recurrent Neural Networks.* 2016. DOI: 10.48550/ARXIV.1611.09913. URL: https://arxiv.org/abs/1611.09913.

[17] Pedro Colon-Hernandez et al. “Combining pre-trained language models and structured knowledge”. In: *CoRR* abs/2101.12294 (2021). arXiv: 2101.12294. URL: https://arxiv.org/abs/2101.12294.

[18] Pedro Colon-Hernandez et al. “Combining pre-trained language models and structured knowledge”. In: *CoRR* abs/2101.12294 (2021). arXiv: 2101.12294. URL: https://arxiv.org/abs/2101.12294.

[19] Alan Cruse. *Meaning in Language: An Introduction to Semantics and Pragmatics.* Oxford University Press UK, 2010.

[20] Shib Sankar Dasgupta, Swayambhu Nath Ray, and Partha Talukdar. “HyTE: Hyperplane-based Temporally aware Knowledge Graph Embedding”. In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, Oct. 2018, pp. 2001–2011. DOI: 10.18653/v1/D18-1225. URL: https://aclanthology.org/D18-1225.

[21] Jacob Devlin et al. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: *CoRR* abs/1810.04805 (2018). arXiv: 1810.04805. URL: http://arxiv.org/abs/1810.04805.

[22] Emily Dinan et al. “Wizard of Wikipedia: Knowledge-Powered Conversational agents”. In: *CoRR* abs/1811.01241 (2018). arXiv: 1811.01241. URL: http://arxiv.org/abs/1811.01241.

[23] Lisa Ehrlinger and Wolfram Wöß. “Towards a Definition of Knowledge Graphs.” In: *SEMANTiCS (Posters, Demos, SuCCESS)*. 2016.

[24] Dumitru Erhan et al. “Why Does Unsupervised Pre-training Help Deep Learning?” In: *Journal of Machine Learning Research* 11.19 (2010), pp. 625–660. URL: http://jmlr.org/papers/v11/erhan10a.html.

[25] Tianyu Gao, Xingcheng Yao, and Danqi Chen. “SimCSE: Simple Contrastive Learning of Sentence Embeddings”. In: *CoRR* abs/2104.08821 (2021). arXiv: `2104.08821`. URL: `https://arxiv.org/abs/2104.08821`.

[26] Xiang Gao et al. “Jointly Optimizing Diversity and Relevance in Neural Response Generation”. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers).* Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 1229–1238. DOI: `10.18653/v1/N19-1125`. URL: `https://aclanthology.org/N19-1125`.

[27] Marjan Ghazvininejad et al. “A Knowledge-Grounded Neural Conversation Model”. In: *CoRR* abs/1702.01932 (2017). arXiv: `1702.01932`. URL: `http://arxiv.org/abs/1702.01932`.

[28] Aaron Gokaslan and Vanya Cohen. *OpenWebText Corpus*. `http://Skylion007.github.io/OpenWebTextCorpus`. 2019.

[29] Ian J. Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. `http://www.deeplearningbook.org`. Cambridge, MA, USA: MIT Press, 2016.

[30] Jiatao Gu et al. “Incorporating Copying Mechanism in Sequence-to-Sequence Learning”. In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers).* Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 1631–1640. DOI: `10.18653/v1/P16-1154`. URL: `https://aclanthology.org/P16-1154`.

[31] Jian Guan et al. “A Knowledge-Enhanced Pretraining Model for Commonsense Story Generation”. In: *CoRR* abs/2001.05139 (2020). arXiv: `2001.05139`. URL: `https://arxiv.org/abs/2001.05139`.

[32] Xu Han et al. “OpenNRE: An Open and Extensible Toolkit for Neural Relation Extraction”. In: *Proceedings of EMNLP-IJCNLP: System Demonstrations.* 2019, pp. 169–174. DOI: `10.18653/v1/D19-3029`. URL: `https://www.aclweb.org/anthology/D19-3029`.

[33] Tianxing He and James R. Glass. “Negative Training for Neural Dialogue Response Generation”. In: *CoRR* abs/1903.02134 (2019). arXiv: `1903.02134`. URL: `http://arxiv.org/abs/1903.02134`.

[34] Karl Moritz Hermann et al. “Teaching Machines to Read and Comprehend”. In: *CoRR* abs/1506.03340 (2015). arXiv: `1506.03340`. URL: `http://arxiv.org/abs/1506.03340`.

[35] Sepp Hochreiter and Jürgen Schmidhuber. “Long Short-Term Memory”. In: *Neural Comput.* 9.8 (Nov. 1997), pp. 1735–1780. ISSN: 0899-7667. DOI: `10.1162/neco.1997.9.8.1735`. URL: `https://doi.org/10.1162/neco.1997.9.8.1735`.

[36] Douglas Hoffstadter and Emmanuel Sander. *Surfaces and essences: Analogy as the fuel and fire of thinking.* 2013.

[37] Aidan Hogan et al. “Knowledge Graphs”. In: *CoRR* abs/2003.02320 (2020). arXiv: `2003.02320`. URL: `https://arxiv.org/abs/2003.02320`.

[38] Gautier Izacard and Edouard Grave. "Leveraging Passage Retrieval with Generative Models for Open Domain Question Answering". In: *CoRR* abs/2007.01282 (2020). arXiv: 2007.01282. URL: https://arxiv.org/abs/2007.01282.

[39] Joel Jang et al. "Towards Continual Knowledge Learning of Language Models". In: *CoRR* abs/2110.03215 (2021). arXiv: 2110.03215. URL: https://arxiv.org/abs/2110.03215.

[40] Joel Jang et al. "Towards Continual Knowledge Learning of Language Models". In: *International Conference on Learning Representations*. 2022. URL: https://openreview.net/forum?id=vfsRB5MImo9.

[41] Dietmar Jannach et al. "A Survey on Conversational Recommender Systems". In: *CoRR* abs/2004.00646 (2020). arXiv: 2004.00646. URL: https://arxiv.org/abs/2004.00646.

[42] Haozhe Ji et al. "Generating Commonsense Explanation by Extracting Bridge Concepts from Reasoning Paths". In: *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*. Suzhou, China: Association for Computational Linguistics, Dec. 2020, pp. 248–257. URL: https://aclanthology.org/2020.aacl-main.28.

[43] Ziwei Ji et al. "Survey of Hallucination in Natural Language Generation". In: *CoRR* abs/2202.03629 (2022). arXiv: 2202.03629. URL: https://arxiv.org/abs/2202.03629.

[44] Karen Spärck Jones. "A statistical interpretation of term specificity and its application in retrieval". In: *Journal of Documentation* 28 (1972), pp. 11–21.

[45] Immanuel Kant. *Critique of Pure Reason*. The Cambridge Edition of the Works of Immanuel Kant. Translated by Paul Guyer and Allen W. Wood. New York, NY: Cambridge University Press, 1998.

[46] Vladimir Karpukhin et al. "Dense Passage Retrieval for Open-Domain Question Answering". In: *CoRR* abs/2004.04906 (2020). arXiv: 2004.04906. URL: https://arxiv.org/abs/2004.04906.

[47] Byeongchang Kim, Jaewoo Ahn, and Gunhee Kim. "Sequential Latent Knowledge Selection for Knowledge-Grounded Dialogue". In: *CoRR* abs/2002.07510 (2020). arXiv: 2002.07510. URL: https://arxiv.org/abs/2002.07510.

[48] Mojtaba Komeili, Kurt Shuster, and Jason Weston. "Internet-Augmented Dialogue Generation". In: *CoRR* abs/2107.07566 (2021). arXiv: 2107.07566. URL: https://arxiv.org/abs/2107.07566.

[49] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". In: *nature* 521 (2015).

[50] Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. "Latent Retrieval for Weakly Supervised Open Domain Question Answering". In: *CoRR* abs/1906.00300 (2019). arXiv: 1906.00300. URL: http://arxiv.org/abs/1906.00300.

[51] Mike Lewis et al. “BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension”. In: *CoRR* abs/1910.13461 (2019). arXiv: `1910.13461`. URL: `http://arxiv.org/abs/1910.13461`.

[52] Patrick S. H. Lewis et al. “Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks”. In: *CoRR* abs/2005.11401 (2020). arXiv: `2005.11401`. URL: `https://arxiv.org/abs/2005.11401`.

[53] Junyi Li et al. “Few-shot Knowledge Graph-to-Text Generation with Pretrained Language Models”. In: *CoRR* abs/2106.01623 (2021). arXiv: `2106.01623`. URL: `https://arxiv.org/abs/2106.01623`.

[54] Wei Li et al. “Leveraging Graph to Improve Abstractive Multi-Document Summarization”. In: *CoRR* abs/2005.10043 (2020). arXiv: `2005.10043`. URL: `https://arxiv.org/abs/2005.10043`.

[55] Zekang Li et al. “Incremental Transformer with Deliberation Decoder for Document Grounded Conversations”. In: *CoRR* abs/1907.08854 (2019). arXiv: `1907.08854`. URL: `http://arxiv.org/abs/1907.08854`.

[56] Chia-Wei Liu et al. “How NOT To Evaluate Your Dialogue System: An Empirical Study of Unsupervised Evaluation Metrics for Dialogue Response Generation”. In: *CoRR* abs/1603.08023 (2016). arXiv: `1603.08023`. URL: `http://arxiv.org/abs/1603.08023`.

[57] Ye Liu et al. “KG-BART: Knowledge Graph-Augmented BART for Generative Commonsense Reasoning”. In: *CoRR* abs/2009.12677 (2020). arXiv: `2009.12677`. URL: `https://arxiv.org/abs/2009.12677`.

[58] Yinhan Liu et al. “RoBERTa: A Robustly Optimized BERT Pretraining Approach”. In: *CoRR* abs/1907.11692 (2019). arXiv: `1907.11692`. URL: `http://arxiv.org/abs/1907.11692`.

[59] Zhiyuan Liu, Yankai Lin, and Maosong Sun. “Representation Learning for Natural Language Processing”. In: *CoRR* abs/2102.03732 (2021). arXiv: `2102.03732`. URL: `https://arxiv.org/abs/2102.03732`.

[60] Ilya Loshchilov and Frank Hutter. “Fixing Weight Decay Regularization in Adam”. In: *CoRR* abs/1711.05101 (2017). arXiv: `1711.05101`. URL: `http://arxiv.org/abs/1711.05101`.

[61] Wenjie Luo et al. “Understanding the Effective Receptive Field in Deep Convolutional Neural Networks”. In: *CoRR* abs/1701.04128 (2017). arXiv: `1701.04128`. URL: `http://arxiv.org/abs/1701.04128`.

[62] Longxuan Ma et al. “A Survey of Document Grounded Dialogue Systems (DGDS)”. In: *ArXiv* abs/2004.13818 (2020).

[63] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval.* Cambridge, UK: Cambridge University Press, 2008. ISBN: 978-0-521-86571-5. URL: `http://nlp.stanford.edu/IR-book/information-retrieval-book.html`.

[64] Michael Mccloskey and Neil J. Cohen. "Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem". In: *The Psychology of Learning and Motivation* 24 (1989), pp. 104–169.

[65] Chuan Meng et al. "RefNet: A Reference-aware Network for Background Based Conversation". In: *CoRR* abs/1908.06449 (2019). arXiv: 1908.06449. URL: http://arxiv.org/abs/1908.06449.

[66] Amil Merchant et al. "What Happens To BERT Embeddings During Fine-tuning?" In: *CoRR* abs/2004.14448 (2020). arXiv: 2004.14448. URL: https://arxiv.org/abs/2004.14448.

[67] Donald Metzler et al. "Rethinking Search: Making Experts out of Dilettantes". In: *CoRR* abs/2105.02274 (2021). arXiv: 2105.02274. URL: https://arxiv.org/abs/2105.02274.

[68] Alessio Micheli. "Neural Network for Graphs: A Contextual Constructive Approach". In: *IEEE Transactions on Neural Networks* 20.3 (2009), pp. 498–511. DOI: 10.1109/TNN.2008.2010350.

[69] Tomas Mikolov et al. *Efficient Estimation of Word Representations in Vector Space.* 2013. DOI: 10.48550/ARXIV.1301.3781. URL: https://arxiv.org/abs/1301.3781.

[70] Reiichiro Nakano et al. "WebGPT: Browser-assisted question-answering with human feedback". In: *CoRR* abs/2112.09332 (2021). arXiv: 2112.09332. URL: https://arxiv.org/abs/2112.09332.

[71] David A. Patterson et al. "Carbon Emissions and Large Neural Network Training". In: *CoRR* abs/2104.10350 (2021). arXiv: 2104.10350. URL: https://arxiv.org/abs/2104.10350.

[72] Jeffrey Pennington, Richard Socher, and Christopher Manning. "GloVe: Global Vectors for Word Representation". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP).* Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1532–1543. DOI: 10.3115/v1/D14-1162. URL: https://aclanthology.org/D14-1162.

[73] Matthew E. Peters et al. "Deep contextualized word representations". In: *CoRR* abs/1802.05365 (2018). arXiv: 1802.05365. URL: http://arxiv.org/abs/1802.05365.

[74] Fabio Petroni et al. "KILT: a Benchmark for Knowledge Intensive Language Tasks". In: *CoRR* abs/2009.02252 (2020). arXiv: 2009.02252. URL: https://arxiv.org/abs/2009.02252.

[75] Jason Phang, Thibault Févry, and Samuel R. Bowman. "Sentence Encoders on STILTs: Supplementary Training on Intermediate Labeled-data Tasks". In: *CoRR* abs/1811.01088 (2018). arXiv: 1811.01088. URL: http://arxiv.org/abs/1811.01088.

[76] Xipeng Qiu et al. "Pre-trained Models for Natural Language Processing: A Survey". In: *CoRR* abs/2003.08271 (2020). arXiv: 2003.08271. URL: https://arxiv.org/abs/2003.08271.

[77] Yingqi Qu et al. "RocketQA: An Optimized Training Approach to Dense Passage Retrieval for Open-Domain Question Answering". In: *CoRR* abs/2010.08191 (2020). arXiv: 2010.08191. URL: https://arxiv.org/abs/2010.08191.

[78] Alec Radford et al. "Improving language understanding by generative pre-training". In: (2018).

[79] Alec Radford et al. "Language Models are Unsupervised Multitask Learners". In: (2018). URL: https://d4mucfpksywv.cloudfront.net/better-language-models/language-models.pdf.

[80] Jack W. Rae et al. "Scaling Language Models: Methods, Analysis & Insights from Training Gopher". In: *CoRR* abs/2112.11446 (2021). arXiv: 2112.11446. URL: https://arxiv.org/abs/2112.11446.

[81] Colin Raffel et al. "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer". In: *CoRR* abs/1910.10683 (2019). arXiv: 1910.10683. URL: http://arxiv.org/abs/1910.10683.

[82] Anand Rajaraman and Jeffrey David Ullman. "Data Mining". In: *Mining of Massive Datasets*. Cambridge University Press, 2011, pp. 1–17. DOI: 10.1017/CBO9781139058452.002.

[83] Nils Reimers and Iryna Gurevych. "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks". In: *CoRR* abs/1908.10084 (2019). arXiv: 1908.10084. URL: http://arxiv.org/abs/1908.10084.

[84] Pengjie Ren et al. "Conversations with Search Engines". In: *CoRR* abs/2004.14162 (2020). arXiv: 2004.14162. URL: https://arxiv.org/abs/2004.14162.

[85] Sascha Rothe, Shashi Narayan, and Aliaksei Severyn. "Leveraging Pre-trained Checkpoints for Sequence Generation Tasks". In: *CoRR* abs/1907.12461 (2019). arXiv: 1907.12461. URL: http://arxiv.org/abs/1907.12461.

[86] Vasile Rus and Mihai Lintean. "A Comparison of Greedy and Optimal Assessment of Natural Language Student Input Using Word-to-Word Similarity Metrics". In: *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*. Montréal, Canada: Association for Computational Linguistics, June 2012, pp. 157–162. URL: https://aclanthology.org/W12-2018.

[87] Maarten Sap et al. "ATOMIC: An Atlas of Machine Commonsense for If-Then Reasoning". In: *CoRR* abs/1811.00146 (2018). arXiv: 1811.00146. URL: http://arxiv.org/abs/1811.00146.

[88] Stefan Schweter and Alan Akbik. "FLERT: Document-Level Features for Named Entity Recognition". In: *CoRR* abs/2011.06993 (2020). arXiv: 2011.06993. URL: https://arxiv.org/abs/2011.06993.

[89] Abigail See, Peter J. Liu, and Christopher D. Manning. "Get To The Point: Summarization with Pointer-Generator Networks". In: *CoRR* abs/1704.04368 (2017). arXiv: 1704.04368. URL: http://arxiv.org/abs/1704.04368.

[90] Kurt Shuster et al. “Retrieval Augmentation Reduces Hallucination in Conversation”. In: *CoRR* abs/2104.07567 (2021). arXiv: 2104.07567. URL: https://arxiv.org/abs/2104.07567.

[91] Shaden Smith et al. “Using DeepSpeed and Megatron to Train Megatron-Turing NLG 530B, A Large-Scale Generative Language Model”. In: *CoRR* abs/2201.11990 (2022). arXiv: 2201.11990. URL: https://arxiv.org/abs/2201.11990.

[92] Robyn Speer, Joshua Chin, and Catherine Havasi. “ConceptNet 5.5: An Open Multilingual Graph of General Knowledge”. In: *CoRR* abs/1612.03975 (2016). arXiv: 1612.03975. URL: http://arxiv.org/abs/1612.03975.

[93] Ron Sun. *Artificial Intelligence: Connectionist and Symbolic Approaches.* 1999.

[94] Chuanqi Tan et al. “S-Net: From Answer Extraction to Answer Generation for Machine Reading Comprehension”. In: *CoRR* abs/1706.04815 (2017). arXiv: 1706.04815. URL: http://arxiv.org/abs/1706.04815.

[95] Yi Tay et al. “Long Range Arena : A Benchmark for Efficient Transformers”. In: *International Conference on Learning Representations*. 2021. URL: https://openreview.net/forum?id=qVyeW-grC2k.

[96] Joshua B. Tenenbaum et al. “How to Grow a Mind: Statistics, Structure, and Abstraction”. In: *Science* 331.6022 (2011), pp. 1279–1285. ISSN: 0036-8075. DOI: 10.1126/science.1192788. eprint: http://science.sciencemag.org/content/331/6022/1279.full.pdf. URL: http://science.sciencemag.org/content/331/6022/1279.

[97] Nandan Thakur et al. “Augmented SBERT: Data Augmentation Method for Improving Bi-Encoders for Pairwise Sentence Scoring Tasks”. In: *CoRR* abs/2010.08240 (2020). arXiv: 2010.08240. URL: https://arxiv.org/abs/2010.08240.

[98] Nandan Thakur et al. “BEIR: A Heterogenous Benchmark for Zero-shot Evaluation of Information Retrieval Models”. In: *CoRR* abs/2104.08663 (2021). arXiv: 2104.08663. URL: https://arxiv.org/abs/2104.08663.

[99] Rakshit Trivedi et al. “Know-Evolve: Deep Reasoning in Temporal Knowledge Graphs”. In: *CoRR* abs/1705.05742 (2017). arXiv: 1705.05742. URL: http://arxiv.org/abs/1705.05742.

[100] Ashish Vaswani et al. “Attention Is All You Need”. In: *CoRR* abs/1706.03762 (2017). arXiv: 1706.03762. URL: http://arxiv.org/abs/1706.03762.

[101] Petar Veličković et al. *Graph Attention Networks.* 2017. DOI: 10.48550/ARXIV.1710.10903. URL: https://arxiv.org/abs/1710.10903.

[102] Jesse Vig et al. “Causal Mediation Analysis for Interpreting Neural NLP: The Case of Gender Bias”. In: *CoRR* abs/2004.12265 (2020). arXiv: 2004.12265. URL: https://arxiv.org/abs/2004.12265.

[103] A. Waibel et al. “Phoneme recognition using time-delay neural networks”. In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 37.3 (1989), pp. 328–339. DOI: 10.1109/29.21701.

[104] Ben Wang and Aran Komatsuzaki. *GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model*. https://github.com/kingoflolz/mesh-transformer-jax. May 2021.

[105] Chenguang Wang, Xiao Liu, and Dawn Song. "Language Models are Open Knowledge Graphs". In: *CoRR* abs/2010.11967 (2020). arXiv: 2010.11967. URL: https://arxiv.org/abs/2010.11967.

[106] Kai Wang, Xiaojun Quan, and Rui Wang. "BiSET: Bi-directional Selective Encoding with Template for Abstractive Summarization". In: *CoRR* abs/1906.05012 (2019). arXiv: 1906.05012. URL: http://arxiv.org/abs/1906.05012.

[107] Jason Weston, Sumit Chopra, and Antoine Bordes. *Memory Networks*. 2014. DOI: 10.48550/ARXIV.1410.3916. URL: https://arxiv.org/abs/1410.3916.

[108] Jing Xu, Arthur Szlam, and Jason Weston. "Beyond Goldfish Memory: Long-Term Open-Domain Conversation". In: *CoRR* abs/2107.07567 (2021). arXiv: 2107.07567. URL: https://arxiv.org/abs/2107.07567.

[109] Michihiro Yasunaga et al. "QA-GNN: Reasoning with Language Models and Knowledge Graphs for Question Answering". In: *CoRR* abs/2104.06378 (2021). arXiv: 2104.06378. URL: https://arxiv.org/abs/2104.06378.

[110] Sanghyun Yi et al. "Towards Coherent and Engaging Spoken Dialog Response Generation Using Automatic Conversation Evaluators". In: *CoRR* abs/1904.13015 (2019). arXiv: 1904.13015. URL: http://arxiv.org/abs/1904.13015.

[111] Donghan Yu et al. "KG-FiD: Infusing Knowledge Graph in Fusion-in-Decoder for Open-Domain Question Answering". In: *CoRR* abs/2110.04330 (2021). arXiv: 2110.04330. URL: https://arxiv.org/abs/2110.04330.

[112] Wenhao Yu et al. "A Survey of Knowledge-Enhanced Text Generation". In: *ACM Computing Survey (CSUR)* (2022).

[113] Houyu Zhang et al. "Conversation Generation with Concept Flow". In: *CoRR* abs/1911.02707 (2019). arXiv: 1911.02707. URL: http://arxiv.org/abs/1911.02707.

[114] Zhu Zhang et al. "Learning to Rehearse in Long Sequence Memorization". In: *CoRR* abs/2106.01096 (2021). arXiv: 2106.01096. URL: https://arxiv.org/abs/2106.01096.

[115] Xueliang Zhao et al. "Knowledge-Grounded Dialogue Generation with Pre-trained Language Models". In: *CoRR* abs/2010.08824 (2020). arXiv: 2010.08824. URL: https://arxiv.org/abs/2010.08824.

[116] Yichu Zhou and Vivek Srikumar. "A Closer Look at How Fine-tuning Changes BERT". In: *CoRR* abs/2106.14282 (2021). arXiv: 2106.14282. URL: https://arxiv.org/abs/2106.14282.

[117] Fengbin Zhu et al. "Retrieving and Reading: A Comprehensive Survey on Open-domain Question Answering". In: *CoRR* abs/2101.00774 (2021). arXiv: 2101.00774. URL: https://arxiv.org/abs/2101.00774.

[118] Yukun Zhu et al. “Aligning Books and Movies: Towards Story-like Visual Explanations by Watching Movies and Reading Books”. In: *CoRR* abs/1506.06724 (2015). arXiv: `1506.06724`. URL: `http://arxiv.org/abs/1506.06724`.