

# Chapter 6

## Foundation Models for Text Generation



**Abstract** This chapter discusses Foundation Models for Text Generation. This includes systems for Document Retrieval, which accept a query and return an ordered list of text documents from a document collection, often evaluating the similarity of embeddings to retrieve relevant text passages. Question Answering systems are given a natural language question and must provide an answer, usually in natural language. Machine Translation models take a text in one language and translate it into another language. Text Summarization systems receive a long document and generate a short summary covering the most important contents of the document. Text Generation models use an autoregressive Language Model to generate a longer story, usually starting from an initial text input. Dialog systems have the task of conducting a dialog with a human partner, typically not limited to a specific topic.

**Keywords** Question answering · Machine translation · Text summarization · Text generation · Dialog systems · Document retrieval

In this chapter we describe Foundation Models, i.e. large Pre-trained Language Models for generating new text in different application areas.

- *Document Retrieval* systems accept a query and return an ordered list of text documents from a document collection, often evaluating the similarity of embeddings to retrieve relevant text passages (Sect. 6.1).
- *Question Answering* systems are given a natural language question and must provide an answer, usually in natural language (Sect. 6.2).
- *Machine Translation* takes a text in one language and generates a translation into another language (Sect. 6.3).
- *Text Summarization* receives a long document and has to write a short summary covering the most important contents of the document (Sect. 6.4).
- *Text Generation* uses an autoregressive Language Model to generate a longer story, usually starting from an initial text input (Sect. 6.5).

**Table 6.1** Language generation tasks illustrated by an example

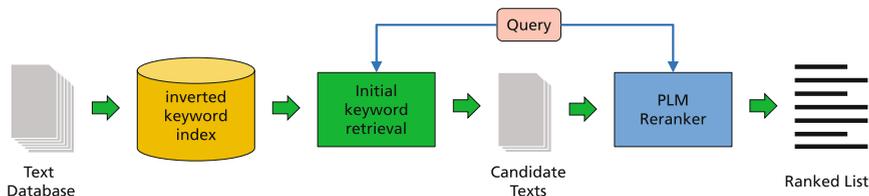
Task	Description	Example
Document retrieval	For a query return an ordered list of text documents	<i>Covid 19?</i> → <a href="http://doi.org/wikipedia/covid-19">http://doi.org/wikipedia/covid-19</a> , <a href="http://www.cdc.gov/">www.cdc.gov/</a> , ...
Generative question answering	Generate the answer to a question, often using some background knowledge	<i>What did Albert Einstein invent?</i>  → <i>Einstein developed the theory of relativity</i>
Translation	For a text in the source language generate a text in the target language with the same meaning	<i>Fritz isst gerne Schinken</i>  → <i>Fritz likes to eat ham</i>
Summarization	For a long text generate a concise summary	<i>It was the middle of winter, ...</i>  → <i>Snow White is awoken by the prince, whom she marries ...</i>
Text generation	Starting from an initial text, a consistent continuation text is created	<i>Beethoven was born in Bonn</i>  → <i>His father was a singer at the Duke's court ...</i>
Dialog answer generation	Generate a consistent response in a dialogue based on the sequence of previous utterances	<i>Could you recommend a video for tonight?</i>  → <i>There is "Memento" on Netflix</i>

- *Dialog systems* have the task of conducting a dialog with a human partner, typically not limited to a specific topic (Sect. 6.6).

Due to the large number of different approaches, we focus on representative models which exhibit a high performance at the time of writing. We review the current best techniques for each area, measured against appropriate benchmarks and taking into account the computational resources required. For standard models a link to the description in earlier chapters is provided. Examples for each application area are shown in Table 6.1.

## 6.1 Document Retrieval

*Information retrieval (IR)* uses computer systems to search databases for content. The resulting IR system is often called a *search engine*. Often, the user formulates a sentence or a *query* about to some topic, and the system is expected to return a sorted list of documents relevant to the query (*ad hoc retrieval*). Here we focus on



**Fig. 6.1** Retrieve-and-rerank architecture using PLMs. First, texts are retrieved from the document collection, usually with exact-match bag-of-words queries. These candidates are then reranked using PLM embeddings, e.g. from BERT. Image adapted from [123], reprinted with kind permission of authors

retrieving textual information from a stored collection of documents. In contrast to question answering approaches in Sect. 6.2, the system does not generate a direct answer to the query in natural language.

Former IR systems were *keyword-based*: all words contained in a document were stored in an *inverted index*. The retrieval algorithm searched the index to identify documents that contained the query words. Then, these documents were ranked according to the information content of each query word found in a document, e.g. measured by tf-idf or BM25 [186]. These two steps are shown in Fig. 6.1. A survey of earlier retrieval techniques is given by Abbasiyantaeb and Momtazi [2]. However, this approach had three major problems:

- Many objects, activities, or events may be expressed by different words called *synonyms*, e.g. “*drink*” and “*beverage*” or “*buy*” and “*purchase*”. The documents containing alternative words are not returned by keyword retrieval. *Paraphrases* like “*he has tons of stuff to throw away*” and “*he needs to get rid of a lot of junk*” are even harder to spot and were ignored. This is called the *vocabulary mismatch problem*.
- Many words have different meanings depending on the context (e.g. “*rock*”: music or stone). These words are called *homonyms*. Part of the retrieved documents containing such a word will be mismatches.
- The order of words is often crucial for the meaning of the sentences (e.g. “*dog kills person*” vs. “*person kills dog*”). This is usually ignored with keyword search.

As an alternative, contextual embeddings were used to represent queries and documents. By identifying matching documents through comparison of contextual semantic representations, word meaning differences between documents and queries can be reduced and texts with synonyms, homonyms, and paraphrases can be retrieved. These models have achieved SOTA results on various retrieval benchmarks [137] and have recently been introduced in commercial search engines. They are therefore one of the most commercially important applications of PLMs to date.

### 6.1.1 Dense Retrieval

*Dense retrieval* methods encode text as an embedding vector with a fixed length much smaller than the text length. Whether a document is relevant to a given query is determined by the similarity of embedding vectors, which is computed by cosine similarity or inner products. Unlike question answering (Sect. 6.2), these models do not generate a direct natural language response to a search query, but return complete documents or text passages. Recently, dense retrieval methods based on PLMs outperformed their keyword counterparts when fine-tuned on a small set of in-domain relevance-labeled documents. Lin et al. [124] provide a comprehensive overview of retrieval systems with PLMs. Different approaches for dense retrieval can be distinguished and are covered in the next sections:

- **Cross-Encoder:** Use the concatenated query and a document as input to BERT and determine the relevance of the document for the query (Sect. 6.1.3).
- **Retrieval with token embeddings:** The tokens of the query and the document are encoded by contextual embeddings. Then different metrics are used to compare these embeddings and to collect relevant documents (Sect. 6.1.4).
- **Retrieval with passage embeddings:** These techniques encode the query and passages of the document by an embedding. Subsequently, these embeddings are compared. This type of embedding respects word order and thus has the potential to return better matches (Sect. 6.1.5).

Only a very small selection of methods can be described, which should give an impression of the approaches currently used as shown in Table 6.2. In Sects. 6.2.2 and 6.2.3 retrieval techniques for question answering are discussed, which are even more powerful. A very comprehensive survey on PLMs for retrieval is provided by Lin et al. [124].

### 6.1.2 Measuring Text Retrieval Performance

There are a number of benchmark datasets used for training and comparing retrieval approaches. The *MS-MARCO* benchmark [16] is a large-scale collection created from about half a million anonymized questions sampled from Bing’s search query logs. For the passage ranking task it contains a corpus of 8.8M passages with an average length of 55 words extracted from 3.6M web documents. The goal is to retrieve passages that answer the question. The training set contains approximately 500k pairs of queries and relevant documents, and another 400M pairs of queries and non-relevant documents. There is a development set and a secret test set with about each 7k queries. However, there is a discussion that the gold annotation of the MS-MARCO benchmark is biased to some extent [10].

**Table 6.2** Document retrieval models with their performance. Benchmarks (Sect. 6.1.2): MARCO: MS-MARCO [16], NQuest: Natural Questions benchmark [109], Wiki65K: long Wikipedia documents [247]

Model	Description	Benchmark
monoBERT (Sect. 6.1.3)	Process each query-passage pair with BERT	MARCO 35.9% MRR@10
monoT5 (Sect. 6.1.3)	Process each query-passage pair with T5	MARCO 38% MRR@10
ColBERT (Sect. 6.1.4)	Reranks search results documents based on token embeddings	MARCO 36.7% MRR@10
Model 1 (Sect. 6.1.4)	Compute the probability that the query is a ‘translation’ of the document	MARCO 39.1% MRR@100
SMITH (Sect. 6.1.4)	Use a BERT-based hierarchical encoder	Wiki65K 95.9% acc.
SentenceBERT (Sect. 6.1.5)	BERT encoder for query and documents	Reduce recall time from 65 h to 5 s
DPR (Sect. 6.1.5)	Different BERT encoders for query and documents, fine-tuned to reduce retrieval loss. FAISS index for approximate nearest neighbor search	NQuest 79.4% top-20 acc.
RocketQA (Sect. 6.1.5)	RoBERTa encoders for query and documents. Later reranking	MARCO 41.9% MRR@10
coCondenser (Sect. 6.1.5)	RoBERTa encoders for query and documents using CLS token. Later reranking	MARCO 40.8% MRR@100

The *Natural Questions (NQ)* [109] contains questions with at least 8 words from real users to the Google search engine. It requires QA systems to read and comprehend an entire Wikipedia article, which may or may not contain the answer to the question. An example is the question “*Where is blood pumped after it leaves the right ventricle?*” The task is to retrieve a long answer, i.e. a paragraph from the page that answers the question, e.g. “*From the right ventricle, blood is pumped through the semilunar pulmonary valve . . .*”, or an indication that there is no answer. The task was designed to be close to an end-to-end question answering application. One to five answers are provided by human annotators. While the original Natural Questions benchmark was a reading comprehension task providing a number of evidence documents for each question, the *EfficientQA benchmark* [147] adapted this to open-domain QA by taking examples with up to five token answers and discarding the evidence documents.

Min et al. [146] note that over half of the queries in Natural Questions are ambiguous, with many sources of ambiguity such as event and entity references. They develop an *AmbigQA* with reformulated questions that yield a unique answer.

A simple evaluation measure is the *top-k accuracy*, the proportion of queries for which one of the  $k$  most likely answers returned is correct. More complex is the *mean reciprocal rank (MRR)*, the inverse of the rank of the first correct answer and 0, if no correct answer was returned. If, for instance, the third answer is correct, the

reciprocal rank is 1/3. The MRR for  $|Q|$  queries is

$$MRR = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{rank_i}. \tag{6.1}$$

$MRR@m$  indicates that always an ordered list of  $m$  documents is returned.

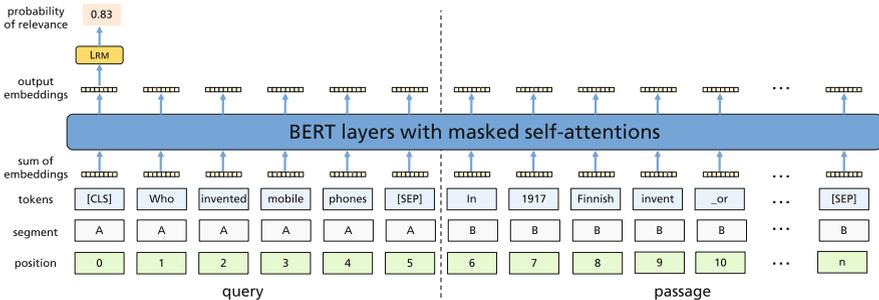
We may define  $Pr(i)$  as the precision reached by the first  $i$  elements of the list of size  $m$ , i.e. the fraction of relevant documents of the first  $i$ . Then we may define the *average precision* as

$$AP = \frac{1}{m} \sum_{i=1}^m Pr(i) * rel(i) \quad MAP = \frac{1}{|Q|} \sum_{j=1}^{|Q|} AP_j \tag{6.2}$$

where  $rel(i) = 1$  if the  $i$ -th document is relevant and 0 otherwise. The *mean average precision* (MAP) is the average of AP over  $|Q|$  different queries.

### 6.1.3 Cross-Encoders with BERT

**monoBERT** [155] performs reranking based on a fine-tuned BERT classifier based on the embedding of the  $[CLS]$  token. Query and document are combined to the input “ $[CLS] <query> [SEP] <document> [SEP]$ ”. This is processed by a BERT fine-tuned on MS-MARCO, where the embedding of  $[CLS]$  in the last layer is used by a logistic classifier to predict the probability that the current document is relevant for the query. This output score is used for ranking (Fig. 6.2). Note that by this technique paraphrases like “*symptoms of influenza include fever and nasal*



**Fig. 6.2** The monoBERT model uses a fine-tuned BERT model for ranking passages with respect to queries. The input contains the query concatenated with the passage. The  $[CLS]$  token embedding is trained to return the probability that the passage answers the query

congestion” and “a stuffy nose and elevated temperature are signs you may have the flu” may be identified.

On the MS-MARCO benchmark [153] monoBERT yields an MRR@10 value of 35.9% (i.e. the first relevant document at position 2.8 on average). As the keyword-based BM25-search before had an MRR@10-value of 16.5% (first relevant document at position 6.1 on average), this result was a dramatic increase in performance of search engines. Such a big jump in effectiveness caused by an individual model is rarely observed in either academia or industry, which led to immediate excitement in the community.

It is quite striking how monoBERT provides a simple yet effective solution to the problem of text ranking (at least for texts that are shorter than its maximal input length) [124]. In several studies monoBERT has been found to be better than BM25 in estimating relevance when term frequency is held constant. Using textual manipulation tests that alter existing documents, rearranging the order of words within a sentence or across sentences was found to have a large negative effect, while shuffling the order of sentences within a document has a modest negative effect. In contrast, rearranging only prepositions had little effect. Experimental results from input template variations show that monoBERT uses exact match, “soft” semantic matches, and information about the position of words. Exactly how these different components are combined—for different types of queries, across different corpora, and under different settings, etc.—remains an open question. Note that this search approach requires enormous computational resources, as for each passage a new evaluation has to be performed, while the effort for index search grows only logarithmically.

**monoT5** [154] used the T5 encoder-decoder model instead of BERT to rerank retrieved documents. The model receives the input “*Query: <query> Document: <document> Relevant:*”. monoT5 is fine-tuned to produce the tokens *true* or *false* if the document is relevant to the query or not. The predicted probability of *true* can be used as a relevance score. For T5 with 3B parameters the authors get an MRR@10-value of 38% for MS-MARCO passage retrieval. This shows that larger models increase performance of retrieval systems.

### 6.1.4 Using Token Embeddings for Retrieval

The all-to-all nature of the BERT attention patterns at each transformer encoder layer means that there is a quadratic complexity in terms of time and space with respect to the input length. In Sect. 3.2 we have introduced a number of approaches to cope with longer inputs. These all can be used to process longer documents. Among the many approaches we discuss ColBERT and Model 1 in more detail.

**ColBERT** [99] reranks the output of another (cheaper) retrieval model, typically a term-based model, or directly for end-to-end retrieval from a document collection. Queries and documents were prepended by different special tokens. ColBERT uses a single pre-trained BERT model to encode each query or document into a bag

of token embeddings. In a final layer the size of embeddings is reduced and they are normalized to Euclidean length 1.0. Hence, the inner product is equivalent to the cosine similarity. If  $(q_1, \dots, q_m)$  are the query tokens and  $d_{i,1}, \dots, d_{i,k}$  are the tokens of the  $i$ -th document, the similarity of  $q$  and  $d_i$  is computed as

$$s_{q,d_i} = \sum_{r=1}^m \max_j \eta(q_r)^\top \eta(d_{i,j}). \quad (6.3)$$

This is the sum of maximum cosine similarities (MaxSim) between each query term and the “best” matching term contained in the document  $d_i$ . For each query embedding the L2-nearest 10 embeddings are taken into account and  $k = 1000$  closest document vectors are retrieved.

For ranking a preliminary search result of, say 1000 documents, the maximum similarities (e.g. cosine similarity) between all query embeddings and all embeddings in the retrieved documents are computed. This approach is very efficient as it requires orders of magnitude fewer FLOPS than previous approaches. On the MS-MARCO benchmark [153] a reranking ColBERT achieves a MRR@10-value of 34.9% (first relevant document at position 2.9 on average), which is slightly below the cross-encoder monoBERT.

ColBERT can also be used for end-to-end retrieval. It employs the *FAISS* index [91] to store the document token embeddings for a  $k$ -nearest neighbor search in a preparatory step. Note that for each token in each document an embedding has to be stored, as the embedding depends on the context. The retrieval requires two stages: in the first stage, a number of approximate searches for each query token is performed. In the second refinement stage, these approximate matches are reranked according to the MaxSim criterion. On the MS-MARCO benchmark the end-to-end retrieval by ColBERT has a MRR@10-value of 36.7%, which is much better than the reranking performance and on par with the much more expensive BERT cross-encoder approach.

**Model 1** [28] mixes a number of techniques for their retrieval model based on token embeddings. First the authors estimate the probability  $p(\mathbf{q}|\mathbf{d})$  that the query  $\mathbf{q}$  has been generated as a “translation” of the document  $\mathbf{d}$ . Using Bayes rule the authors get

$$p(\mathbf{d}|\mathbf{q}) \propto p(\mathbf{q}|\mathbf{d})p(\mathbf{d}) \propto p(\mathbf{q}|\mathbf{d}) \quad (6.4)$$

assuming a uniform prior  $p(\mathbf{d})$  [21]. They consider the probability  $r(q_i|d_j)$  that a query token  $q_i$  is a translation of a document token  $d_j$ . Approximating  $r(q_i|d_j)$  by a neural network, they use embeddings of tokens  $q_i$  and  $d_j$  as inputs and are able to estimate  $p(\mathbf{d}|\mathbf{q})$ . The approach requires little computational effort. The authors combined the BERT dense retriever with a Lucene search index. Finally, they expand documents for Model 1 with Doc2query. *Doc2query* [156] aims at generating queries, for which the document is relevant. The approach trains a transformer to generate up to 100 query tokens from a document of up to 400

tokens. The model is trained using datasets consisting of pairs of query and relevant documents, e.g. MS-MARCO. On MS-MARCO they achieve 39.1% MRR@100. The context-free neural Model 1 is less effective than a BERT-based ranking model, but it can run efficiently on a CPU (without expensive index-time precomputation or query-time operations on large tensors).

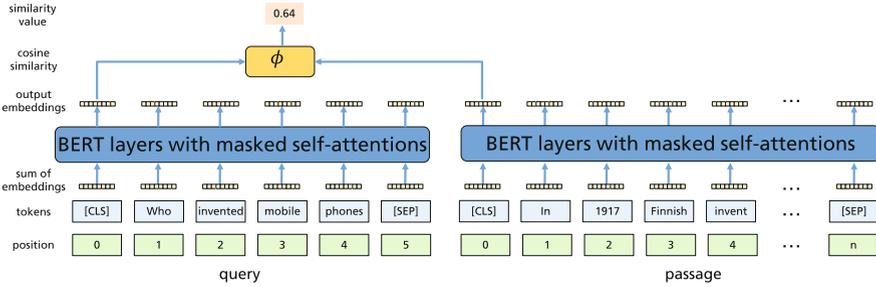
Currently, no retriever tries to process long documents. This has many important applications like news recommendation, related article recommendation and paper citation suggestion. Usually, long documents are partitioned into passages with the idea that the relevant contents is contained in a passage. Note that PLMs with longer inputs, e.g. BigBird, can improve performance (Sect. 3.2). However, it is clear that this has to be evaluated. The **SMITH** model [247] uses a BERT-based hierarchical encoder to capture the document structure information. The document is first partitioned into sentences and for each sentence token embeddings are computed. Each sentence starts with an *[CLS]* token, whose embedding represents the sentence. There is a higher sentence level BERT which just receives the sentence embeddings as input. The first artificial token of second level BERT is used as the embedding of the whole document.

The model is pre-trained by the masked language modeling task to get token embeddings. In addition, in the second level there is a masked sentence block prediction task where the model has to select the correct embedding from all sentence embeddings in a batch. The fine-tuning task maximizes the relevance score predicted from the document embedding by a logistic classifier for the relevance-annotated fine-tuning dataset. On the *Wiki65K* with long Wikipedia articles [87] the approach achieves an accuracy of 95.9% which is a significant improvement over prior approaches.

### 6.1.5 Dense Passage Embeddings and Nearest Neighbor Search

Representing text passages by embedding vectors has the potential to solve the problem of vocabulary mismatch by directly matching “meaning” in a representation space. These so-called *dense retrieval* techniques can perform ranking directly on vector representations generated by PLMs. In contrast to calculating pairwise differences of token embeddings, this approach offers a much more efficient retrieval procedure. This is performed by matching the embedding vector of a query with the embedding vectors of passages employing an index and approximate nearest neighbor search. Efficient, scalable solutions are available today in open-source libraries.

Given a query  $q$  and a set of documents  $D = \{d_1, \dots, d_n\}$  we want to define functions  $\eta_q(\cdot)$  and  $\eta_d(\cdot)$ , which convert the token sequences  $q$  and  $d$  into fixed-width vectors. The functions should have the property that the similarity between  $\eta_q(q)$  and  $\eta_d(d_i)$  is maximal if  $d_i$  is relevant for query  $q$ . We want to estimate



**Fig. 6.3** The SentenceBERT model uses two fine-tuned BERT models to transform queries and passages to embeddings of the  $[CLS]$  token. Subsequently, a cosine similarity module is used to compute a similarity value

$$p(\text{relevant} = 1 | d_i, q) := \phi(\eta_q(q), \eta_d(d_i)), \quad (6.5)$$

where  $\phi(\cdot)$  is a similarity comparison function, e.g. the scalar product [124, p. 133]. Note that  $\eta_d(d_i)$  may be precomputed and organized in an index. By using different encoders  $\eta_q(\cdot)$  and  $\eta_d(\cdot)$  for queries and documents, we can take into account the different roles and wordings of queries and documents.

**SentenceBERT** [183] is the prototype of a bi-encoder design for generating semantically meaningful sentence embeddings to be used in large-scale textual similarity comparisons (Fig. 6.3). The query  $q$  and the documents  $d_i$  are processed by the same PLM (BERT or RoBERTa). Similarity was compared by the *cosine similarity*

$$\phi(\eta_q(q), \eta_d(d_i)) = \frac{\eta_q(q)^\top \eta_d(d_i)}{\|\eta_q(q)\| * \|\eta_d(d_i)\|}. \quad (6.6)$$

To generate sentence embeddings the authors investigated three alternatives. (1) Use the embedding of the  $[CLS]$  token. (2) Averaging (mean-pooling) of all output embeddings. (3) Component-wise maximum (max-pooling) of all output embeddings. Without fine-tuning the results were worse than for non-contextual embeddings. Fine-tuning boosted performance and yields a new SOTA. It turned out that average pooling was the most effective design, slightly better than max pooling or using the  $[CLS]$  token. Most important the computation time for finding the best match in 10,000 documents was reduced from 65 h to 5 s.

**DPR** [94] used separate encoders  $\eta_q(q)$  and  $\eta_d(d_i)$  for the query  $q$  and the text passages  $d_i$  of about 100 words. Both encoders took the  $[CLS]$  embedding from  $\text{BERT}_{\text{BASE}}$  as its output representation. As comparison function the inner product  $\eta_q(q)^\top \eta_d(d_i)$  was used. For each query  $q_i$  the training set contained one correct passage  $d_i^+$  and a number of negative passages  $d_{i,1}^-, \dots, d_{i,m}^-$ . The loss function

encoded the goal to get a large  $\phi$ -value (i.e. similarity) for  $q_i$  and  $d_i^+$  and small similarities for  $q_i$  and  $d_{i,j}^-$

$$L(w) = -\log \frac{\exp[\eta_q(q)^\top \eta_d(d_i^+)]}{\exp[\eta_q(q)^\top \eta_d(d_i)] + \sum_{j=1}^m \exp[\eta_q(q)^\top \eta_d(d_{i,j}^-)]} \quad (6.7)$$

The negative examples were a mixture of passages retrieved with keyword search that did not contain the answer and thus were difficult negatives. In addition, passages from other examples in the same training batch were used. Instead of performing an exhaustive computation of similarities for all documents between  $\eta_q(q)$  and the  $\eta_d(d_i)$ , we can employ an approximate nearest neighbor search. *FAISS* [91] is an open-source method based on hierarchical navigable small world graphs. For the Natural Questions benchmark they achieved a top-20 accuracy of 79.4%, which is much better than the previous top-20 accuracy of 59.1% for the keyword-based BM25 search. The replication study [136] could confirm these results, but found that a hybrid approach of DPR and BM25 could increase the performance to 82.6%.

**ANCE** [238] uses a single RoBERTa model to encode query and document. During training, hard negative examples are selected by approximate nearest neighbor search on an index over the representations generated by the trained encoder. In this way, they can select “difficult” negative examples. The index is periodically updated. On Natural Questions ANCE achieved 82.1% top-20 accuracy. The performance was also compared with the monoBERT cross-encoder, which reranks first-stage BM25 results with monoBERT by comparing all documents to the query. It turned out that on MS-MARCO the application of monoBERT to BM25 had a  $MRR@10$  of 34.7% while ANCE has 33%. The cross-encoder obviously is more effective than ANCE. The authors also applied ANCE to 8 billion documents using embeddings of size 64 and approximate nearest neighbor search. They reported a gain of 16% compared to the prior commercial implementation.

**RocketQA** [184] performs a first retrieval step and subsequently a re-ranking procedure. Both approaches are jointly optimized using a listwise training approach, where a list of positive and negative examples is used for training both modules. In addition, they perform a data augmentation to construct diverse training instances by incorporating both random sampling and denoised sampling. They report a  $MRR@10$  on MS-MARCO of 38.8% for passage retrieval. When the 50 top results are reranked later, they can increase  $MRR@10$  to 41.9%.

**coCondenser** [63] is one of the highest entries of the MS-MARCO leaderboard [140]. The model is forced to learn to aggregate information into the “CLS” embedding, which will then participate in the LM prediction. Then an additional “contrastive loss” is used: “CLS” embeddings of passages from the same document close together should be similar, while those for passages in different documents should have a larger distance. This yields highly expressive embeddings for passages. When the model is fine-tuned on MS-MARCO, it returns an  $MRR@100$  of 40.8% on the MS-MARCO leaderboard [140].

## Available Implementations

- DPR code is available at <https://github.com/facebookresearch/DPR>.
- The code for the FAISS nearest neighbor search is available at <https://github.com/facebookresearch/faiss>.
- ANCE code and data trained nearest neighbor search is available at <https://github.com/microsoft/ANCE>.
- RocketQA code and data is available at <https://github.com/PaddlePaddle/RocketQA>.
- FlexNeuART [27] implements the Model 1 retrieval system [28].
- coCondenser code at <https://github.com/luyug/Condenser>.

### 6.1.6 Summary

Retrieval is a crucial step in web search, in which a small set of query-relevant candidate passages are identified from a corpus of billions of texts. Discovering more semantically related candidates in the retrieval phase holds great promise for presenting more high-quality results to the end user. Dense retrieval approaches represent a paradigm shift in search engine technology. They make it possible to recognize the meaning of words and paraphrases and thus find much better passages matching a query. Search results can also be used for question-answer models (Sect. 6.2) and dialog systems (Sect. 6.6). They are already being used in production search engine by Bing [35, 238, 266], Google [152, 197], and Facebook [82].

Dense retrieval methods discussed above are fine-tuned in a supervised setting using human relevance labels as input, e.g. from MS-MARCO. Best results are obtained by two different PLMs to encode the query and the documents. Both PLMs are trained to improve the probability of a correct reference document in contrast to some negative documents. As two different PLMs require more effort, most systems use a single model to encode the question and the documents. Experiments show that the combination of dense retrieval and keyword retrieval seems to have advantages. In Sects. 6.2.2 and 6.2.3 retrieval techniques for question answering are discussed, which are even more powerful.

A problem is the transferability of a search system to a new domain. BERT was found to have strong cross-domain relevance classification capabilities when used in a similar way as monoBERT [124, p. 72]. If a BERT model is fine-tuned using relevance judgments from one domain (e.g., tweets) it can be successfully applied to a different domain (e.g., newswire articles). On the other hand, Thakur et al. [221] created a benchmark called *BEIR* with 18 retrieval tasks from very different domains like bio-medicine and tweets. The authors trained a large number of dense retrieval techniques on MS-MARCO and evaluated them on the other tasks. They found that they were on average less effective than BM25, which due to its simplicity just works in most cases.

The memory requirements for an index for embeddings cannot be ignored. While a keyword Lucene index for the MS-MARCO passage corpus with 8.8M passages needs 661 MB, a FAISS index for vectors of size 768 requires 42 GB and an index for ColBERT takes 156 GB [124, p. 159]. To apply these techniques to web-scale, approaches with a smaller memory footprint are needed.

## 6.2 Question Answering

*Question Answering* (QA) is an application of NLP that receives a natural language query and automatically generates a precise answer in natural language. It is a long-standing AI task dating back to the 1960s [69]. Compared with search engines discussed in Sect. 6.1, the QA system presents the final answer to a question directly instead of returning a list of relevant snippets or hyperlinks. Thus, it is more user-friendly and efficient. Often, the system has access to a database or a *knowledge base* (KB) of documents, such as Wikipedia, where it can search for relevant information.

A *Closed Domain QA* system handles questions for a specific domain, e.g. medicine, and has background knowledge about that domain or is trained with a large training set covering that domain. *Open Domain QA* systems (ODQA) deal with questions on almost any topic and usually rely on general KBs or Internet search [37]. *Multimodal QA* systems address questions in different media, e.g., text and images. A survey of ODQA is given by Zhu et al. [265]. Table 6.3 compiles leading QA Models with their performance.

A simple form of question answering is *Reading Comprehension*, where the system has to identify an answer to a question in a given text. Often a BERT-like system marks the answer span in the text by span prediction (Sect. 2.1.3). This task can mainly be considered as solved. For the *SQuAD 2.0 benchmark* [179] ALBERT yields more than 93% F1-value and the fine-tuned *ST-MoE-32B* mixture-of-experts model (Sect. 3.5.2) with 269B parameters [270] achieves 96.3% F1-value, while the human F1-value is 89.5% [178]. However, Sen et al. [199] indicate that systems trained on one dataset may not generalize well to other benchmarks.

### 6.2.1 Question Answering Based on Training Data Knowledge

Language models often are trained on comprehensive text collections and are able to memorize a large amount of information. A frequently used benchmark is *Natural Questions* (NQ) [109], which has been sampled from the Google search logs (Sect. 6.1.2). For the given question, the system has to find a short answer span in the given support documents. An example is the question “*When are hops added to the brewing process?*”, which should yield the answer “*The boiling process*”.

The *TriviaQA* benchmark [92, 226] contains a set of trivia questions with answers that were originally scraped from the Web. Different from Natural Questions, the

**Table 6.3** Question answering models with their performance. The lower part contains retrieval models. Benchmarks: NQ: natural Questions benchmark of Google queries [109], TriviaQA: TriviaQA benchmark [92, 226], HotpotQA: multihop benchmark [249], EM: exact match

Model	Details	Benchmark
BigBird (Sect. 6.2.1)	Autoencoder with long input, supervised training with QA pairs	NQ with ref-docs 57.9% EM WikiHop 82.3% acc.
PoolingFormer (Sect. 6.2.1)	Autoencoder with two-level attention schema, supervised training with QA pairs	NQ with ref-docs 61.6% EM
RealFormer (Sect. 6.2.1)	Autoencoder with bypass attention, supervised training with QA pairs, multihop QA	WikiHop 84.4% acc.
GPT-3 (Sect. 6.2.1)	Large autoencoder 175B, only pre-training	NQ few-shot 29.9% TriviaQA few-shot 71.2%
Gopher (Sect. 6.2.1)	Large autoencoder 280B, only pre-training	NQ few-shot 28.2%
PaLM (Sect. 6.2.1)	Large autoencoder 540B, only pre-training	NQ few-shot 36.0% TriviaQA few-shot 81.4%
DPR (Sect. 3.4.5)	Retriever-reader with two BERT models and FAISS index	NQ exact match acc 41.5% TriviaQA 57.9%
FiD (Sect. 3.4.5)	Retriever-reader with T5 models and FAISS index	NQ exact match acc 51.4% TriviaQA 67.6%
REALM (Sect. 3.4.5)	Retriever-reader with dot product of BERT embeddings, slow	NQ exact match acc 40.4%
FB HYBRID (Sect. 3.4.5)	DPR retriever combined with other retriever, FiD reader	NQ exact match acc 53.9%, corresponds to 67.4% correct
MS UNITED (Sect. 3.4.5)	BERT-based retriever, T5+ELECTRA as readers, final re-ranking	NQ exact match acc 54.0%, corresponds to 65.8% correct
AISO (Sect. 3.4.5)	Retriever-reader with repeated retrieval rounds, multihop QA	HotpotQA 72.0% F1
RETRO (Sect. 6.2.3)	Language model with frozen BERT retriever, language model periodically includes retrieved token chunks	NQ exact match acc 45.5%
WEBGPT (Sect. 6.2.3)	GPT-3 combined with Bing search engine, which can be periodically invoked	TriviaQA 69.5%

questions here are written with known answers in mind. *TruthfulQA* [125] is a special QA benchmark with short questions that are constructed adversarially, so that some people’s answers might be wrong due to false beliefs and misconceptions. The answers are evaluated according to informativeness and truthfulness.

## Fine-Tuned Question Answering Models

The **BigBird** (Sect. 3.2) self-attention was used as an autoencoder and trained with the MLM objective using an input sequence of 4096 tokens [253]. During fine-tuning on Natural Questions the model had to find a short answer span in one of the given evidence documents. The model achieved 57.9% F1-value on this task. The **PoolingFormer** [256] is an alternative model for long input sequences with a two-level attention schema. Its first level uses a smaller sliding window pattern to aggregate information from neighbors. Its second level employs a larger window to increase receptive fields with pooling attention. An ensemble of fine-tuned PoolingFormers achieves 61.6% F1-value on the Natural Questions benchmark. The model is similar to the **SMITH** model [247], which uses a BERT-based hierarchical encoder to capture the document structure information (Sect. 6.1.4).

An alternative is **Macaw** [218], a freely available QA-system with 11B parameters. It is built on T5 and has strong zero-shot QA-capabilities. On a set of 300 challenge questions the authors claim that Macaw outperforms GPT-3 by 10%, although it has only a small fraction of its parameters. In addition to providing an answers for a question, Macaw can also take an answer and produce a question; or generate multiple-choice options for an answer and a question. The authors also provide a detailed analysis of errors.

It is much more difficult to combine different pieces of evidence to find an answer. A benchmark to test this ability is *WikiHop* [232], where information from different documents has to be merged. An example is the question “*Hanging gardens of Mumbai, country?*” and the documents “*The Hanging Gardens, in Mumbai, also known as Pheroze Shah Mehta Gardens, are terraced gardens ...*” and “*Mumbai is the capital city of the Indian state of Maharashtra. It is the most populous city in India ...*”. For each query up to 140 background paragraphs are provided to the model. On this benchmark BigBird-ETC (Sect. 3.2.1) achieved an accuracy of 82.3%. Currently, the best model for this task is the **RealFormer** with an accuracy of 84.4% [171], which is slightly below the human performance of 85%. The RealFormer is an autoencoder with a modified architecture, which provides a bypass with the raw attention scores of all attention heads from the previous layer in the subsequent layers [76].

## Question Answering with Few-Shot Language Models

Recent Foundation Models are trained with an enormous collection of documents and can generate answers to questions without additional knowledge input. An example is the autoregressive language model **GPT-3** with 175B parameters, which was pre-trained on a text collection of books, Wikipedia and web pages of about 500 billion tokens (Sect. 3.1.2). Because of its high model capacity it can absorb a lot of ‘knowledge’ in its parameters. When a Foundation Model is not allowed to use external information, this is called *Closed-book QA*.

As discussed in Sect. 3.6.3, GPT-3 can be instructed by a few examples (few-shot) to solve a task. Figure 6.4 provides a few-shot prompt example. For Natural Questions, GPT-3 achieves an exact match accuracy of 14.6% in the zero-shot setting, 23.0% in the one-shot setting, and 29.9% in the few-shot setting [29, p. 14]. This was achieved without fine-tuning on Natural Questions. The larger **Gopher** model with 280B parameters (Sect. 3.1.2) performs slightly worse with 28.2% in the few-shot setting [175, p. 80].

The even larger **PaLM** model with 540B parameters (Sect. 3.1.2) was trained on a high-quality dataset with 780B tokens. It uses a new prompt technique to pose logical questions, where examples are presented to the system together with *thought chains* partitioning a reasoning task into smaller problems (Sect. 3.6.4). In this way it gets the recipe to combine facts from different sources to arrive at the final answer.

PaLM was evaluated on a large number of other benchmarks, which in part are QA-tasks. On Natural Questions it arrived at an accuracy of 21.2% with 0-shots and at 36.0% with few-shot prompts [43, p. 47]. On Trivia QA (questions concerning the Wikipedia), BoolQ (question answering with yes/no answers), and PIQA (question answering with reasoning) PaLM also achieved a new SOTA. The results are shown in Table 3.4. PaLM was benchmarked with a large number of tests, among them the more than 150 BIG-bench tasks (Sect. 4.1.4). Many of them are QA-related tasks: 21 contextual QA tasks, 24 context-free QA tasks, 36 reading comprehension tasks, and a large number of tasks on specific knowledge and common sense [1, 22]. Additional outcomes for QA-benchmarks of PaLM are given in [43, p. 12], where PaLM always achieves SOTA.

**Prompt:**  
 I am a highly intelligent question answering bot. If you ask me a question that is rooted in truth, I will give you the answer. If you ask me a question that is nonsense, trickery, or has no clear answer, I will respond with "Unknown".  
 Q: What is human life expectancy in the United States?  
 A: Human life expectancy in the United States is 78 years.  
 Q: Who was president of the United States in 1955?  
 A: Dwight D. Eisenhower was president of the United States in 1955.  
 Q: Which party did he belong to?  
 A: He belonged to the Republican Party.  
 Q: What is the square root of banana?  
 A: Unknown  
 ...  
 Q: Where is the Valley of Kings?  
 A:

---

**Answer:**  
 The Valley of Kings is located in Luxor, Egypt.

**Fig. 6.4** A possible few-shot prompt for GPT-3 to get an answer based on existing knowledge acquired during pre-training [160]

## 6.2.2 Question Answering Based on Retrieval

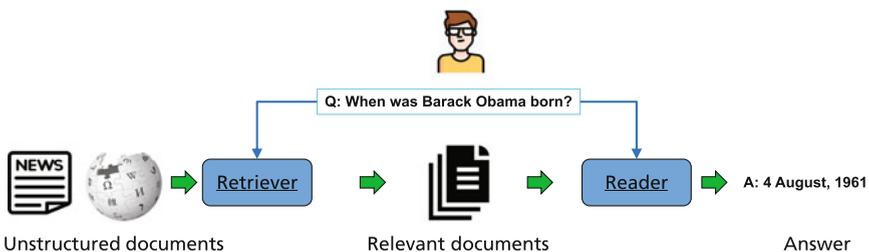
Retrieval ODQA systems usually work in two stages: for a question a *retriever* module finds a number of documents from a text collection, for which might contain the answer. Subsequently, a *reader* considers the question and the retrieved documents and generates a natural language answer (Fig. 6.5). Since the model relies on external information, it is referred to as *Open-book QA*.

Retrievers have been introduced in Sect. 3.4.5 and were discussed in the context of document retrieval in Sect. 6.1. The retriever may employ a traditional search engine using tf-idf weighting or BM25. Alternatively the retriever may be a *dense retriever* based on document and question embeddings. It is trained to retrieve passages by computing embedding similarities e.g. by DPR [94] (Sect. 3.4.5). A tutorial on ODQA is provided by Chen [36].

The *reader* is usually an autoregressive language model that receives both the query and the retrieved documents as inputs. It is fine-tuned to generate a response to the query based on the retrieved information and its internal knowledge.

Question answering with external knowledge bases has the advantage that curated KBs usually are checked for correctness. They may have, however, limited coverage of entities and relations may not be up-to-date. There are a number of approaches to combine PLMs with KBs using techniques like entity mapping (Sect. 3.4.1). Recent papers propose a hybrid approach using KBs and retrieval [239]. Knowledge-Guided Text Retrieval [145] starts with retrieving text passages for a query. It creates a passage graph, where vertices are passages of text and edges represent relationships that are derived either from an external knowledge base or co-occurrence in the same article. On Natural Questions [109] they achieve an accuracy of 34.5%.

**HYBRIDER** [41] uses information from a retriever as well as from a structured KB and tables. The authors collected Wikipedia pages and constructed a benchmark dataset HybridQA containing question-answer pairs requiring multi-hop reasoning using text, tables and hyperlinks (Fig. 6.6). The model first links questions to



**Fig. 6.5** Question answering often combines dense retrieval with an answer selection module. The retriever performs a dense retrieval by comparing the embedding of the query with the embeddings of passages. The reader ranks the retrieved documents and generates an answer by an autoregressive Pre-trained Language Model [36]. Credits for image parts in Table A.2



**Fig. 6.6** For hybrid question answering Wikipedia pages are retrieved by HYBRIDER [41] (top left). Some pages contain tables (left). Here the column titles may be interpreted as well as hyperlinks to entities (underlined). The lower part shows two human-annotated question-answer pairs. Image reprinted with kind permission of the authors [41, p. 2]

tables cells as well as Wikipedia passages and hyperlinks. In a reasoning phase the linked information is ranked and consolidated to derive the probabilities of different answers. The experiments with the dataset show that the utilization of tables or retrieval alone achieves an exact match accuracy of about 20% while the joint model yields more than 40%. However, the hybrid model’s score is still far below human performance.

One of the first retrieval-reader systems was **DPR** (Dense Passage Retriever) [94]. It employs a BERT model to encode passages by embeddings and retrieves them by approximate *k*-nearest neighbor search with the FAISS index (Sect. 6.1.5). In this way it can gather passages with similar meaning but different wording. The DPR reader is another BERT model which is fine-tuned to predict a probability for each retrieved passage that this passage contains the correct answer. In addition, it selects a span of tokens by span prediction, which probably provides the answer. The approach can be easily applied to KBs with billions of passages [94, 213]. On the *Natural Questions* [109] it yields a test set accuracy of 41.5%.

**FiD** [84] is described in Sect. 3.4.5. The retriever is based on DPR and compares query and passages embeddings. Raffel et al. [177] have shown that generative models like T5 can produce the answer for QA-tasks. FiD processes the query and the retrieved passages by a *reader* based on a T5 model to generate an answer. Since the first step is to process the passages one by one, the system is very efficient. FiD achieves an exact match accuracy of 51.4% on the *Natural Questions* test set compared to 41.5% for DPR.

**REALM** [75] and **RAG** [114] are retrieval augmented generative models for open domain question answering. However, they process all retrieved passages simultaneously in an autoregressive language model and were unable to take into account a large number of passages leading to lower accuracies on *Natural Questions* of 40.4 for REALM and 44.5 for RAG. Sachan et al. [194] propose an

end-to-end differentiable training method for retrieval-augmented ODQA. Latent variables indicate which of the relevant documents should be included. The values of the latent variables are iteratively estimated by an EM-algorithm. On Natural Questions they achieve an exact match accuracy of 52.5%.

**MTR** [138] starts from the observation that neural retrievers perform well on their fine-tuning domain, but will typically achieve low out-of-domain performance. The authors propose a multitask retriever similar to DPR which is jointly fine-tuned on eight diverse retrieval tasks. They use a shared passage encoder—so that a single index of encoded passages can be used—as well as a query encoder that is shared across all tasks. In five of the eight models they achieve a higher performance than special models tuned to the corresponding domain.

**AISO** [268] is a retriever-reader architecture for solving multi-hop QA tasks, where multiple documents are required to answer a question. Repeated retrieval rounds are performed in which associated terms are taken as new search queries to find additional evidence. The approach is adaptive and at each step selects one of three types of retrieval operations (e.g., BM25, DPR, and hyperlink) or one answer operation. On the *HotpotQA benchmark* [249], the question-answering system must find the answer to a query in the scope of the entire Wikipedia. The AISO model achieved a new SOTA with a joint F1-value of 72.0%.

The **FB Hybrid** system was presented at the EfficientQA competition [147], where real user questions for the Google search engine from the Natural Questions dataset [109] were tackled. While the original NQ was a reading comprehension task providing a number of evidence documents for each question, the EfficientQA benchmark [147] adapted this to open-domain QA by taking examples with up to five token answers and discarding the evidence documents. The system uses a retriever-reader architecture [158]. The retriever is a mixture of DPR and another retrieval system, which covers lists and tables as well as KB-relations and retrieves 100 passages. The reader is a T5-large Seq2seq model, which is given 100 passages from the retriever and generates an answer. The background corpus contained 18.8M passages from Wikipedia. On Natural Questions the model achieves an exact match accuracy of 53.9%. According to an evaluation by human raters the model was able to answer 67.4% of the questions correctly, which is about as good as the performance of human experts using a search engine. The **MS UnitedQA** model had a similar architecture [139]. It uses a BERT-based retriever and a reader combined from a T5-model and ELECTRA processes the returned documents to generate different answers. A final re-ranking model selects the answer. MS UnitedQA yields an exact match accuracy of 54.0% and 65.8% correctness on Natural Questions. If the systems were restricted to a memory footprint of 6 GB the performance was only marginally reduced.

### 6.2.3 Long-Form Question Answering Using Retrieval

#### A Language Model with Integrated Retrieval

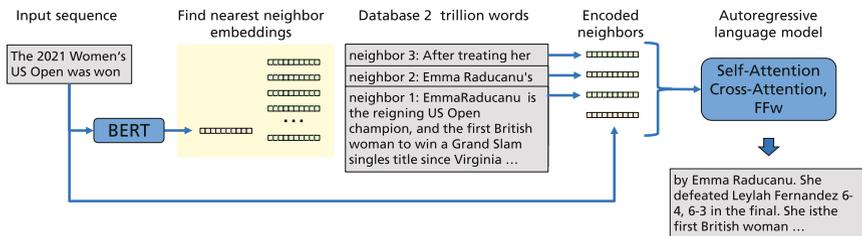
**Retro** [25] is an autoregressive language model with 7B parameters using retrieved information to predict the next token. As retriever a frozen BERT model is employed (Fig. 6.7). Each training sequence is split into chunks, which are augmented with their  $k$ -nearest neighbors retrieved from the database of 2 trillion tokens. The returned information is processed in a language model to improve the prediction of the next token leading to large performance gains. The reader consists of a differentiable autoregressive encoder and a chunked cross-attention module to predict tokens.

An input sequence  $\mathbf{v} = (v_1, \dots, v_n)$  of  $n=2048$  tokens is split into chunks  $\mathbf{c}_t = (v_{(t-1)*m+1}, \dots, v_{t*m})$  of length  $m=64$ . Each chunk  $\mathbf{c}_t$  is expanded with a set  $\text{RET}(\mathbf{c}_t)$  of retrieved  $k$  nearest neighbor chunks from the database. The probability of a token  $v_{t*m+i}$  in the next chunk  $\mathbf{c}_{t+1}$  then can be recursively computed as

$$p(v_{t*m+i} | v_{t*m+(i-1)}, \dots, v_{t*m+1}, \mathbf{c}_t, \text{RET}(\mathbf{c}_t), \dots, \mathbf{c}_1, \text{RET}(\mathbf{c}_1)). \quad (6.8)$$

The probability of the  $i$ -th token of the  $(t+1)$ -th chunk  $\mathbf{c}_{t+1}$  depends only on the previous tokens and on the data  $\text{RET}(\mathbf{c}_j)$  retrieved from the database for the previous chunks. This integrates the retrieval process in the language model.

The retriever for a chunk  $\mathbf{c}_t$  uses the average  $\text{BERT}(\mathbf{c}_t)$  of all BERT embeddings of the tokens in  $\mathbf{c}_t$  as key. It retrieves the  $k$  nearest neighbors from the database with respect to the  $L_2$  distance  $\|\text{BERT}(\mathbf{c}_t) - \text{BERT}(\tilde{\mathbf{c}}_s)\|_2^2$ . The model receives the corresponding chunks  $\tilde{\mathbf{c}}_{s,j}$  and additionally their continuation chunk  $\tilde{\mathbf{c}}_{s+1,j}$  for  $j = 1, \dots, k$ , which collectively form the elements of  $\text{RET}(\mathbf{c}_t)$ . By filtering it is avoided that the chunk to be predicted is included in  $\text{RET}(\mathbf{c}_t)$ , as this would invalidate the conditional probability definition. The retrieval is performed in  $O(\log T)$  time using the *SCaNN* library [73], which collects a set of chunks from a database of 2 trillion tokens in 10ms. Note that the document corpus of Retro is about 1000 times larger than the databases of FiD and other retrieval models.



**Fig. 6.7** The Retro language model retrieves information for the input sequence. The model uses the input sequence and the retrieved neighbor chunks from the database as input and generates an appropriate output [176]

Inside the reader the retrieved tokens in  $\text{RET}(c_t)$  are fed into an autoencoder, which computes a set  $E$  of encoded neighbors. Then, so-called RETRO blocks

$$\text{RETRO}(H, E) := \text{FCL}(\text{CATL}(\text{ATTL}(H), E)), \quad (6.9)$$

and standard self-attention blocks  $\text{LM}(H) := \text{FCL}(\text{ATTL}(H))$  are interleaved and operate on the intermediate embeddings  $H \in \mathbb{R}^{n \times d}$ . Here  $\text{FCL}(\cdot)$  is a fully connected layer,  $\text{ATTL}(\cdot)$  a self-attention layer, and  $\text{CATL}(\cdot, E)$  a cross-attention layer which includes the information in  $E$ . The input and output dimension of these modules is  $\mathbb{R}^{n \times d}$ .

The resulting language model is able to predict the next token with a high reliability. The *Pile data* [62] is a 825GB open-source text collection set that consists of 22 diverse, high-quality datasets. It was screened for toxic language and bias, e.g. with respect to gender, religion, and race. Its authors recommend measuring the quality of token prediction in *bits per byte* (*bpb*), which in contrast to perplexity is independent of the tokenizer [62, p. 6]. The authors compare Retro with GPT-3<sub>175B</sub> [29], Jurassic-1<sub>178B</sub> [121], and Gopher<sub>280B</sub> [176]. It turns out that Jurassic-1 has the lowest (and best) bpb-value on 5 Pile datasets, Gopher on 2 datasets and Retro on 9 datasets, although it is far smaller than the other models [25]. GPT-3 was inferior to all three models. A possible problem for these results is the overlap of the retrieval corpus with the test data.

For the *LAMBADA benchmark* [165] a model has to predict the last word of a paragraph. The authors measure the following accuracies: Retro without retrieval 70%, Retro with retrieval 73%, Gopher 74.5%, and GPT-3 76.2%. Note that Retro has only 4% of the parameters of GPT-3. For question answering the Natural Question benchmark is relevant. Here, Retro achieved an exact match accuracy of 45.5%.

The *LaMDA* [222] dialog system (Sect. 6.6.3) is an expanded version of Retro with 137B parameters. It demonstrates that facticity can be improved by retrieval models. In addition, it is able to reduce toxic language by a system of filters that block unwanted speech. Although this model could also easily be used for question answering, no corresponding benchmark results are known.

## Controlling a Search Engine by a Pre-trained Language Model

**WebGPT** [149] extends GPT-3 to control the *Bing search engine* and performs a web search for a specific query. The language model must issue commands such as “*Search ...*”, “*Find in page: ...*” or “*Quote: ...*”, as shown in Fig. 6.8. In this way, the model collects passages from web pages which contain information relevant for the question. The utilization of Bing has the advantage that it has powerful search capabilities, and covers a large number of up-to-date documents.

Browsing continues until the model issues a command to end browsing, the maximum total length of references has been reached, or the maximum number

Command	Effect
Search <query>	Send <query> to the Bing API and display a search results page
Clicked on link <link ID>	Follow the link with the given ID to a new page
Find in page: <text>	Find the next occurrence of <text> and scroll to it
Quote: <text>	If <text> is found in the current page, add it as a reference
Scrolled down <1, 2, 3>	Scroll down a number of times
Scrolled up <1, 2, 3>	Scroll up a number of times
Top	Scroll to the top of the page
Back	Go to the previous page
End: Answer	End browsing and move to answering phase
End: <Nonsense, Controversial>	End browsing and skip answering phase

**Fig. 6.8** Possible actions of the WebGPT language model. If another text is generated, this is an invalid action and ignored [149]

of actions has been reached. If a relevant reference has been retrieved, the model will generate a long-form answer to the question.

The GPT-3 model is first fine-tuned to mimic human demonstrations, enabling it to use the text-based browser to answer questions. Then, the usefulness and accuracy of the model’s answers is improved by fine-tuning a reward model to predict human preferences, and optimizing it by rejection sampling. Specifically the model is fine-tuned to answer questions from *ELI5* [56], a dataset of open-ended questions obtained from the subreddit ‘Explain Like I’m Five’. An example is given in Fig. 6.9. The proposed WebGPT answers should be coherent, relevant, and supported by trustworthy documents. No details are reported on the input prompts of GPT-3 containing the current state of search, and how the GPT-3 model combines the returned documents into an answer. Note, however, that there is significant overlap between training and validation in *ELI5*, as at least 81% of *ELI5* validation questions occur in the training set [106] in circumscribed form.

The final answers were selected from 64 trials of the 175B WebGPT model by ranking. These answers were preferred by human raters to the reference responses from the *ELI5* dataset 69% of the time. Moreover, they were preferred to the human demonstrator responses in 56% of the cases.

For WebGPT, responses to TruthfulQA [125] were correct about 75% of time, whereas GPT-3 scored 64% with helpful prompts. While GPT-3’s answers were truthful and informative in about 20% of the time, the best version of WebGPT increased this to about 56%. Since people answered 94% of the questions correctly, the models still have a significant performance difference. On TriviaQA WEBGPT achieved a score of 69.5%, which is far less than the value of PaLM with 81.4%.

An innovative feature is the support of text passages by references. This corresponds to the approach of scientific papers to underpin claims by references and was already suggested by Metzler et al. [143]. The references explain the answer and support the factual accuracy of the statements. The citations are selected by Bing in response to the query. They should therefore be close to the final reader-generated response and provide an easy way to assess the correctness of the response.

However, the authors point out that the references are not always representative for the available evidence, although the model cites references that correspond to the generated text. In addition, it is difficult for the model to verify the trustworthiness

**Question:**  
How do contact lenses stay on your iris? What stops them from moving all over your eye?

---

**Answer:**  
Contact lenses stay in place by adhering to the tear film of the eye, and through the pressure from the eyelids<sup>[1]</sup>. The way that contact lenses stay in place in the eye is by sticking to the layer of tear fluid on the surface<sup>[2]</sup>. This tear fluid floats on your eye at all times and the pressure of your eyelids help to make the contact lenses “*stick to it*” with lubrication<sup>[2]</sup>. When you blink, you are providing lubrication and flushing away the impurities that might become stuck on the lens as well<sup>[2]</sup>. Your contact lenses can fall out for a variety of reasons, including improper fit and rubbing your eyes too vigorously<sup>[3]</sup>.

---

[1]. What are your Contact Lenses? Optometrics.org  
 [2]. Are Non-Nonprescription Contacts Safer to Wear? Drs. Campbell, Cunningham, Taylor, and Haun  
 [3]. Why Do My Contacts Keep Popping Out Of My Eyes? www.self.com

**Fig. 6.9** Long-form answer to a question generated by WebGPT. The best of 64 answers was automatically selected. The citations were automatically retrieved from the Bing search engine and added to the answer [80]

of references. Here, Web-of-Trust systems and search engine technology could be employed, which favor trust-checked frequently linked web pages.

### Available Implementations

- BigBird code and models are available at <https://huggingface.co/google/bigbird-roberta-base>
- DPR code and models <https://github.com/facebookresearch/DPR>
- FiD code and models <https://github.com/facebookresearch/FiD>
- RealFormer code <https://github.com/jaketae/realformer>
- REALM code <https://github.com/google-research/language/blob/master/language/realms/README.md>
- RETRO implementation, Deepmind’s Retrieval based Attention net, in PyTorch. This will deviate from the paper slightly, using rotary embeddings for relative positional encoding, as well as FAISS library instead of SCAANN <https://github.com/lucidrains/RETRO-pytorch>.

### 6.2.4 *Summary*

A number of Foundation Models have been presented, which were able to improve Question Answering performance. Examples are the autoregressive language models GPT-3 (175B), Gopher (175B), and PaLM (540B) with huge parameter sets, which are trained on a large document collections and can acquire extensive knowledge. Using few-shot prompts they are able to answer questions with high accuracy without employing external knowledge.

Recently, the retriever-reader architecture has been increasingly used for QA systems. It has the potential to tap into a larger knowledge base or the Internet that can easily be kept up-to-date. The retriever can employ keyword search or dense retrieval. Dense retrieval mitigates the term-mismatch problem, where relevant paraphrases are ignored. Usually, embeddings for each document or phrase are pre-computed and the embedding index is constructed beforehand. Current systems can access document collections of up to trillions of tokens using advanced nearest-neighbor search engines like FAISS and ScaNN to compare embeddings.

The reader usually receives the query and the returned passages in text form and generates the answer. It is fine-tuned to select the correct answer and to provide answers which are expressive and truthful. The Retro model is an autoregressive language model with only 7B parameters, which uses passages retrieved by a frozen BERT model as additional current state information to generate the next tokens. It is capable of improving accuracy to high levels for many QA tasks, but can also be used for other applications such as story generation.

WebGPT combines GPT-3 and the Bing search engine to retrieve documents and create appropriate answers. It is able to enhance the generated text by references to documents, which justify and explain the answer. The LaMDA dialog model is an expanded version of Retro with 137B parameters with specific tuning to increase usability and factual accuracy. In addition, it is able to reduce toxic language by a system of filters that block unwanted speech. These techniques can also be applied to question answering.

Still difficult is the generation of answers where the correct response needs information from multiple documents. In this case several rounds of querying are necessary. Special models like RealFormer, HYBRIDER, or AISO can improve the performance for benchmarks like WikiHop.

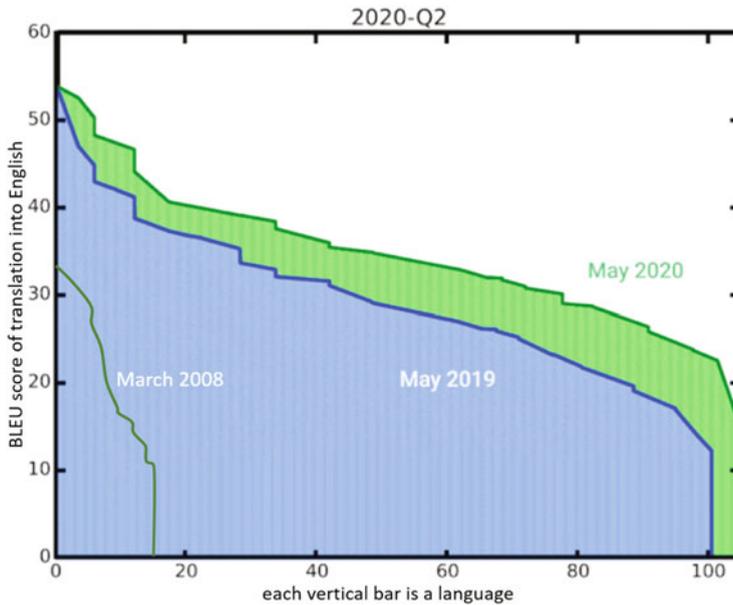


**Fig. 6.10** This map shows some of the world’s 7100 languages, with each dot representing a language and the color indicating the top language family for each language. Only a small fraction of the world’s languages are currently represented in Foundation Models. Image reprinted with kind permission of the authors [24, p. 23]

### 6.3 Neural Machine Translation

Language is the cornerstone of most human communication and interaction. Moreover, many persons think in terms of language, and use it to express and communicate feelings, goals, and ideas. We communicate knowledge by language and use it to establish social and emotional relationships. There are more than 7100 languages in the world [19], some of which are shown in Fig. 6.10. The ability to understand each other across language barriers is essential for communicating ideas between people.

After an initial success with Recurrent Neural Networks [15, 215] the development of the Transformer encoder-decoder (Sect. 2.3) has driven progress in Neural Machine Translation (NMT). By cross-attention a “correlation” between each token of the source text and the translated text can be established, producing better translations than before. The availability of large training sets and better model architectures has steadily increased the performance of Pre-trained Language Models for NMT (Fig. 6.11). Standard models for multilingual processing are described in Sect. 3.3. A survey is provided by Yang et al. [248].



**Fig. 6.11** BLEU scores for Google translation of 100+ different languages to English for different years. Image credits in Table A.2

### 6.3.1 Translation for a Single Language Pair

The training data of NMT consist of text pairs of the source language and its translations to the target language. Traditionally evaluation is done by comparing one or more reference translations to the proposed translation, as described in the survey [195]. There are a number of automatic metrics like BLEU, METEOR or BERT-score (Sect. 2.3.3). It turned out that there is a noticeable difference between human judgment and automatic evaluation. Therefore, most high-end comparisons today use human translators to assess the quality of translation methods.

At the WMT2021 Machine Translation conference, numerous teams solved benchmarks tests for translating English news texts from/to German, Japanese, Russian, Chinese, and a number of low-resource languages [5]. Instead of using comparison statistics like BLEU, the translations of each system was evaluated by a number of human evaluators without showing them a reference translation. They were asked to rate a given translation according to how adequately it expressed the meaning of the corresponding source language input on an analog scale, which corresponds to an underlying absolute rating scale of 0–100. As some raters could be stricter, the systems are ranked by a z-score, where the score is mean-centered and normalized per rater. Systems are grouped together according to which system significantly outperforms all others measured by the Wilcoxon rank-sum test. A large effort was spent to assess the validity of human evaluation.

In total 173 submissions were received. In addition, five anonymized online systems were included. Further human-produced reference translations were denoted by “HUMAN” in all tables. Results show that almost all good systems are based on transformer encoder-decoders. Words are mostly encoded by the SentencePiece [107] tokenizer (Sect. 1.2). A widely used technique is *back-translation* [200]. Here a monolingual text is translated to the other language and then back-translated. By minimizing the difference to the original text, both models may be improved. Up to 500M sentences per language were available and could be used for back-translation, which led to a significant improvement in quality. In addition, ensembles are able to increase the performance in most cases.

The result of the best system for each language pair is shown in Table 6.4. Usually, there is a cluster of 2–5 models at the top, whose performance differences are not significant. The Facebook-AI model (FB) had the best results for half of the language pairs. In addition, the BLEU scores for the best systems automatically computed from  $n$ -grams are shown. As can be seen, the values are in general better for the translation “to English” than “from English” especially for morphology rich languages like Czech and German. Compared to the human reference translation, the best system was significantly better for three language pairs. This has already been discussed critically by Toral [223], who decries the limited amount of context between sentences and the limited translation proficiency of the evaluators.

Improved performance was reached by increasing the number of parameters. The Facebook model [224], for instance, used a standard model of 4.7B parameters and a sparsely gated mixture-of-experts system with up to 128 experts. In each Sparsely Gated MoE layer, each token is routed to the top-2 expert feedforward blocks based on the score of a learned gating function. In addition, the models were fine-tuned with domain-specific data from the news domain. The  $n$ -best hypotheses were generated with a beam search. These were ranked with a weighted average of the probabilities  $p(\text{tgt}|\text{src})$ ,  $p(\text{src}|\text{tgt})$ , and  $p(\text{tgt})$ , where  $\text{src}$  is the source and  $\text{tgt}$  is the target sentence.

It is well-known that the translation of single sentences suffers from ambiguities (e.g. pronouns or homonyms), which can be resolved by considering the document context. In WMT2021 this is taken into account by assessing the quality of translation within the document context [5]. As current encoder-decoder Foundation Models are able to consider larger contexts, this could improve translation performance [141]. Instead of finding the most probable translation of a sentence, we need to generate the best translation for a given complete source document. While comparing sentence-level translation often does not indicate a difference between human and machine translation, the comparison of document-level translation often yields a statistically significant preference for human translations [110].

Instead of using a Seq2seq model with extra long input sequence, **HAT** [187] proposes a hierarchical attention transformer. The authors split the input text into sentences and start each sentence  $i$  with a specific  $[BOS_i]$  token. These tokens summarize the sentence content and are connected to the other sentences by the usual self-attention and cross-attention. While the usual encoder-decoder transformer has a BLEU of 32.5 for the document translation from English to German on WMT2019, HAT is able to yield a SOTA BLEU of 34.5.

**Table 6.4** Leading systems of the WMT2021 News Translation Task. The systems are ordered by normalized z-score [5, pp. 15–19]. If either the best system or a human reference translation is significantly better, the value is printed in bold. Systems: FB: Facebook-AI, BL: Borderline, HW: HW-TSC, NV: Nvidia-NeMo, NI: NiuTrans, OB: Online-B, OW: Online-W, HN: HappyNew Year

Score	Czech	German	Hausa	Icelandic	Japanese	Russian	Chinese
<b>To English</b>							
Best model z-score	FB <b>0.111</b>	BL <b>0.126</b>	FB 0.248	FB 0.293	HW 0.141	NV 0.137	NI 0.042
Human z-score	-0.085	-0.081				0.089	0.019
Best model BLEU	43.1	53.0	18.8	40.6	27.8	56.3	33.4
<b>From English</b>							
Best model z-score	FB 0.263	OB <b>0.266</b>	FB 0.264	FB 0.594	FB 0.430	OW 0.277	HN 0.284
Human z-score	<b>0.397</b>	0.030	0.362	<b>0.872</b>	0.314	0.317	0.325
Best model BLEU	33.6	31.3	20.4	30.6	46.9	45.0	49.2

### 6.3.2 Multilingual Translation

Usually, languages with scarce training data have a much lower translation accuracy, as holds for Hausa in Table 6.4. A recent success was the extension of NMT by multilinguality, which was already discussed in Sect. 3.3. This led to a marked improvement of translations for languages with few resources. For a survey see [48].

**M2M** of Facebook AI [57] improves translation between many languages by utilizing a massive corpus of 7.5B sentences covering 100 languages and thousands of translation directions with supervised data, created through large-scale mining. The model is a transformer encoder-decoder with 15B parameters. The authors add a special token in the encoder indicating the source language and a special token in the decoder indicating the target language. The transformer has 12 encoder and 12 decoder layers and an embedding size of 1024. As there is a joint token vocabulary for all languages, the input and output embeddings are shared. To improve performance the authors added language-specific layers to the decoder for five languages. Using specific parallelization techniques they were able to train the model with only hundreds of GPUs.

Except for four language directions (En→Chinese, Chinese→En, En→Fi, En→Estonian) the model improved translation results on the WMT benchmarks for 1.9 BLEU points on average. Especially marked is the improvement for regional languages with an average increase of 7.6 BLEU. For resource-rich language pairs Liu et al. [130] propose to use very deep transformers with up to 60 encoder layers and 12 decoder layers. They develop a simple yet effective initialization technique that stabilizes training and achieve SOTA on WMT2014 En-Fr of 46.4 BLEU.

Although multilingual translation has many advantages, it usually performs worse than specially trained bilingual models for high-resource language pairs. Recently Facebook [225] presented a single multilingual model, which outperformed the best specially trained bilingual models across 10 out of 14 language pairs of the WMT2021 news benchmark. Facebook built two multilingual systems: any-to-English and English-to-any. They employed data mining techniques to identify translations in large web crawl data and leverage available monolingual data with hundreds of millions of sentences from all eight languages to maximize performance of MT systems. They filtered the available monolingual data to reduce the amount of noise, and then back-translated them with an ensemble of the strongest multilingual models available. The number of parameters was increased from 15B to 53B to enhance the model capacity.

The BLEU scores are shown in Table 6.5. In comparison to the best bilingual models of WMT2021, the multilingual model achieves a better BLEU in 9 of 14 cases indicating that the additional training data from other languages supports translation. Only for Chinese→English there was a larger drop of 1.3 BLEU points. The authors also performed a human evaluation for the language pairs English→Russian and English→German. It turned out that there was no perceived difference between the quality of bilingual and multilingual translations.

**Table 6.5** BLEU scores of the Facebook multilingual model and the best language pair model submitted to the WMT2021 news task. The numbers reported are BLEU scores on the final WMT2021 test set [225]. The difference between the models is printed in bold, if the multilingual model is better

Model	Czech	German	Hausa	Icelandic	Japanese	Russian	Chinese
<b>From English</b>							
FB-Mult	36.1	31.3	20.1	33.3	46.8	46.0	49.9
WMT2021 best	33.6	31.3	20.4	30.6	46.9	45.0	49.2
Difference	<b>2.5</b>	0.0	-0.3	<b>2.7</b>	-0.1	<b>1.0</b>	<b>0.7</b>
<b>To English</b>							
FB-Mult	43.5	53.3	21.0	41.7	27.7	57.1	32.1
WMT2021 best	43.1	53.0	18.8	40.6	27.8	56.3	33.4
Difference	<b>0.4</b>	<b>0.3</b>	<b>2.1</b>	<b>1.1</b>	-0.1	<b>0.8</b>	-1.3

**Table 6.6** Influence of different modeling improvements on the BLEU scores on the development set of WMT2021 for Facebook AI’s WMT2021 submission [225]. The version of the last row was submitted

Improvement strategy	Czech	German	Hausa	Icelandic	Japanese	Russian	Chinese
Bilingual	33.1	38.7	14.7	25.8	25.4	25.8	40.0
+ Back-translation	33.1	39.6	23.1	29.4	26.1	25.7	42.4
+ Fine-tuning	35.7	39.5	23.3	29.4	27.7	26.0	43.0
+ Multilingual	36.4	40.8	24.6	31.2	29.7	26.8	43.6
+ Ensemble	36.8	41.1	25.0	32.5	29.7	26.9	43.6
+ Reranking	37.2	41.1	25.5	32.8	29.7	27.4	43.6
+ Postprocessing	39.8	42.6	25.5	34.5	29.8	28.8	48.2

Table 6.6 shows the effect of employed improvement strategies for the different languages of the multilingual model. Back-translation has a large effect for languages with little training data like Hausa and Icelandic. The authors note, however that back-translation produces *translationese* by generating artificial uncommon phrases in a language. These effects may be mitigated by fine-tuning on the specific domain, e.g. news texts. This yields about 3 BLEU points for translation into English and 0.7 BLEU points for translation out of English. Switching to the multilingual model generates an improvement for all models. While the effect of model ensembles is minor, re-ranking the BEAM translations with conditional target-source probabilities yields about 0.4 BLEU points. Postprocessing (for example applying standard punctuation rules) can have a large effect, e.g. 5 BLEU points for Chinese.

The **PaLM** autoregressive language model with 540B parameters [43] has about 22% non-English training texts among its 780B training tokens (Sect. 3.1.2). Similar to other large LMs, PaLM is not trained explicitly on parallel text, although some such data is likely to exist naturally in the training corpus. In Table 6.7 the results of PaLM 540B few-shot translation is compared with prior few-shot and fine-tuned SOTA [43, p. 27]. The best BLEU value per language pair is underlined and the

**Table 6.7** Comparison of PaLM few-shot translation performance against prior fine-tuned translation performance by specialized models and prior few-shot performance. On the left you find the translation from English and into English for the traditional WMT language pairs. On the right there is the translation to and from English to Kazakh (kk) and a translation between German and French [43, p. 27]

From	en	en	en	fr	de	ro	en	de	kk	fr
To	fr	de	ro	en	en	en	kk	fr	en	de
Prior fine-tuned SOTA	<u>45.6</u>	<u>41.2</u>	<u>33.4</u>	<u>45.4</u>	41.2	39.1	<u>15.5</u>	<u>31.5</u>	<u>30.5</u>	<u>24.9</u>
Prior few-shot SOTA	33.9	26.8	20.5	38.8	40.6	37.3	–	–	–	–
PaLM 540B few-shot	<b>44.0</b>	<b>37.4</b>	<b>28.7</b>	<b>42.8</b>	<b>47.5</b>	<b>43.8</b>	5.1	25.7	20.8	17.4

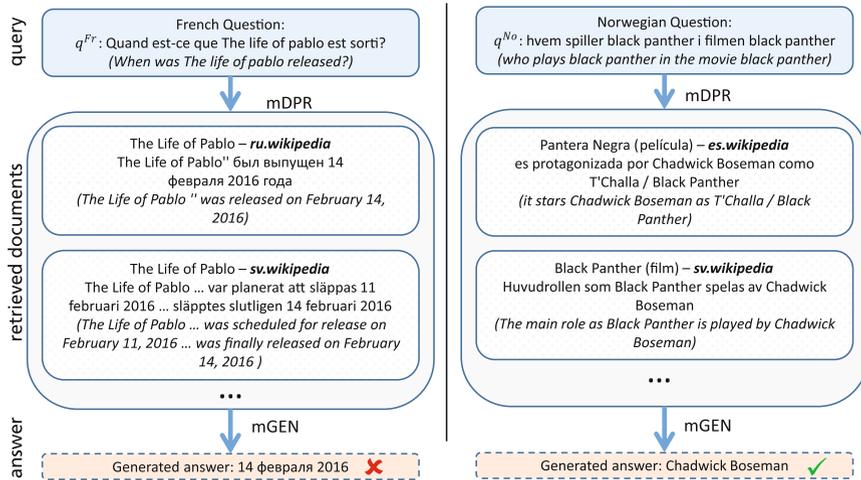
best few-shot BLEU is printed in bold. The table shows that PaLM on the traditional WMT translation pairs always achieves the best few-shot BLEU, often improving by a wide margin. For the low-resource language Kazakh (kk) the fine-tuned translation models have a better BLEU than PaLM. However, for de→en and ro→en PaLM is able to outperform the supervised models. In addition, the 0-shot PaLM translation of fr→en achieves a BLEU value of 25.2, which is better than the fine-tuned SOTA of 24.9. Overall, PaLM performs well close to the fine-tuned models without having been trained for this task.

### 6.3.3 Multilingual Question Answering

In recent years open domain question answering (ODQA) has taken a rapid development (Sect. 6.2). Therefore, it is extremely rewarding to extend these techniques to multilingual question answering. In this way, information encoded with the world’s different languages can be tapped and the digital divide can be narrowed by bringing answers to people who speak rarer languages. There is a tutorial on multilingual ODQA by Ruder [192, 193].

A simple way to perform multilingual ODQA is to translate the question to English, use an English ODQA system to generate an answer, and translate the answer back to the target language. Because of ambiguities in translation, this procedure may generate errors in some cases [132]. In addition, information specific to the target language and conceptualizations of the target culture may not be available in English [258].

The *TyDiQA-GoldP benchmark* [44] is a question answering dataset covering 11 typologically different languages with 204K question-answer pairs. The following languages are included: English, Arabic, Bengali, Finnish, Indonesian, Japanese, Kiswahili, Korean, Russian, Telugu, Thai. As the languages represented in this benchmarks have a very diverse structure, a model which performs well on this data can be expected to have a good QA-accuracy on other languages. *MKQA* [133] is an evaluation dataset created by translating 10k Natural Questions [109] to 25 target languages.



**Fig. 6.12** Cross-lingual retrieval by mDPR and answer generation with mGEN for the CORA system [13, p. 9]. The answers to the questions are correct, however, on the left side the answer should have been given in French

As an alternative, one can train cross-lingual retriever and reader models combining the information from multiple languages to generate an answer in the target language (Fig. 6.12). CORA [13] answers questions across many languages, even for ones without language-specific annotated data or knowledge sources. It includes a dense passage retriever collecting documents with different languages for a question. A pre-trained multilingual language model *mDPR* using mBERT (Sect. 3.3.1) is fine-tuned to encode passages and questions separately. By performing a maximum inner product search the top  $k$  documents are retrieved similar to DPR (Sect. 3.4.5). It could be shown that mBERT improves the search quality in non-English mono-lingual retrieval [203]. The reader *mGEN* is a multilingual autoregressive sequence model (e.g. mT5, Sect. 3.3.2) generating the answer in the target language by compiling the information in the retrieved passages. No specific translation models are used. The initial training data is a combination of multilingual QA datasets. Each training instance from these datasets comprises a question, a positive passage, and an answer. However, these datasets suffer from limitations on language diversity. Therefore, the authors iteratively generate more representative training data for low-resource languages by exploiting links between Wikipedia articles in different languages.

It turns out that CORA substantially outperforms the previous SOTA on multilingual open QA benchmarks across 26 languages, 9 of which are unseen during training. Here CORA can improve the average F1-value from 17.1 to 21.8. Retrieval with mDPR performs well in Indo-European languages with Latin script, even when the language is unseen. There is a major drop for languages with non-Latin script

**Table 6.8** Comparison against SOTA on TyDiQA question answering benchmark with 11 typologically different languages. The values are for the validation set with respect to the exact match accuracy [43, p. 32]. Best values for each language printed in bold

Model	Ar	Bn	En	Fi	Id	Ko	Ru	Sw	Te	Avg
mT5 XXL	76.9	80.5	75.5	76.3	81.8	75.7	76.8	84.4	83.9	79.1
ByT5 XXL	<b>80.0</b>	<b>85.0</b>	<b>77.7</b>	78.8	<b>85.7</b>	<b>78.3</b>	<b>78.2</b>	84.0	<b>85.5</b>	<b>81.4</b>
PaLM 540B fine-tuned	75.0	83.2	75.5	<b>78.9</b>	84.1	75.7	77.1	<b>85.2</b>	84.9	80.0
PaLM 540B few-shot	56.4	54.0	65.5	66.4	69.2	63.8	46.8	75.6	46.9	60.5

(e.g., Japanese, Russian, Chinese). Here, perhaps, the model is unable to use relevant passages from other languages to answer questions.

**mT5** (Sect. 3.3.2) is a multilingual version of the T5 Seq2seq transformer with up to 13B parameters [246]. It was pre-trained using a training dataset of web pages covering 101 languages with about 48B tokens and a common vocabulary of 250k tokens. After fine-tuning on the TyDiQA benchmark, it arrives at an exact match score of 79.1%. **ByT5** [245] is a variation of the mT5 multilingual encoder-decoder with 12.9B parameters. It operates on utf-8 bytes with a vocabulary of 256 possible byte values instead of tokens. The model is pre-trained to replace corrupted spans of 20 bytes on average. The largest model uses 36 encoder and 12 decoder layers. When the model is fine-tuned on gold data in all target languages, it achieves an exact match score of 81.4% on the TyDiQA benchmark.

The **PaLM** Foundation Model [43] has about 22% non-English training texts in its 780B training tokens (Sect. 3.1.2). Therefore, it can be applied to multilingual tasks such as translation and question answering. With few-shot prompts it gets an exact match score on TyDiQA of 60.5%. When the model is fine-tuned on TyDiQA, the score grows to 80.0%, which is slightly below of the performance of ByT5 XXL. The detailed results in Table 6.8 show the performance for different languages. Here PaLM has a better score for two languages than ByT5. The authors remark, that ByT5 was trained with 50% more non-English text compared to PaLM, which may explain the difference.

### Available Implementations

- Hugging Face provides Marian, BART and T5 (up to 11B parameters) as well as multilingual mBART and mT5 implementations and trained models <https://huggingface.co/transformers/>.
- The M2M-100 [55] is available with open-source data collection scripts, model code and parameters of trained models. In addition, the Fairseq system <https://github.com/pytorch/fairseq> can freely be used.
- The CORA [13] implementation of multilingual QA, generated training data and trained models are available at <https://github.com/AkariAsai/CORA>.

### 6.3.4 Summary

In recent years, machine translation has taken a dramatic development. The use of encoder-decoder PLMs could overcome the limitations of RNN architectures and increase the performance to near-human levels. Besides the utilization of encoder-decoder Transformers, the availability of high-quality training examples by web crawlers using Foundation Models and specific assessment procedures is a reason for progress [33]. A further improvement resulted from sentence back-translation, which particularly increases results for low-resource languages, and from training a single multilingual model for translation between all languages. Training multilingual translation models with up to 600B parameters—using appropriate parallelization strategies—leads to significant performance increase for 100 languages, as measured by BLEU [113]. Recently multilingual models even were able to outperform high-resource bilingual translation models. This is also demonstrated by the PaLM Foundation Model, which achieved higher performance in few-shot translation than the prior fine-tuned models for some language pairs. Therefore, multilingual models are likely to become standard in the future. However, current multilingual models using unsupervised multilingual training may not deeply model the subtleties of languages and language varieties to their full extent. This has to be checked in future applications.

The developments opened up the opportunity for multilingual question answering systems, e.g. CORA, where queries can be posed in a large number of languages. The answers are compiled from information available in multiple languages. In this way, cultural characteristics and concepts that are not available in all languages can be taken into account. There are also close links to cross-lingual semantic parsing, where a natural language utterance is translated to a logical form for execution in some knowledge base to return an answer [202]. Again the PaLM Foundation Model provided few-shot answers to multilingual questions, which are competitive in accuracy to fine-tuned models for the same benchmarks. A fine-tuned version of PaLM is even able to outperform prior fine-tuned SOTA for two languages.

However, machine translation is not yet solved. There is still the problem of domain mismatch between train and test data. In some cases, it fails to accurately capture the meaning of a sentence. Systems can generate biased text, e.g. if gender is handled differently in different languages. But attention allows the decoder to look directly at faraway text and provides a soft alignment between words for free. Recently, performance could be increased by translating entire documents, as sentences often are not sufficient to disambiguate all words. To extend current multilingual models to thousands of languages, new techniques are required [19]. One approach is to use monolingual datasets to improve translation, since the amount of available monolingual text is orders of magnitude greater than the amount of translated text. This in addition requires highly reliable language detectors which also work for low-resource languages.

## 6.4 Text Summarization

With the rapid increase of textual information in companies and on the Internet, it is increasingly difficult for people to keep track of a topic. Automatic *summarization* of documents, which compiles the essential statements from a text, can help to grasp the most relevant information in the documents. A *summary* is a short version produced from a single document or multiple documents conveying the main points of the original texts. The purpose of automatic text summarization is to create a *summarizer* method to produce this summary efficiently and precisely. Recent in-depth surveys are provided by Ma et al. [135], Guan et al. [71], Syed et al. [216], and El-Kassas et al. [95].

Earlier machine learning approaches produced *extractive summaries* selecting a few sentences from the document. This approach typically selected grammatically correct sentence parts, but the language style of the combined parts and the coverage were usually not sufficient. Modern summarizers pose summarization as a translation problem, which translates the original document to a short version covering the main points. Since 2017 the encoder-decoder transformer (Sect. 2.3) provided an effective technique to generate *abstractive summaries* containing the main points of the document. Abstractive summarization is a bit more complex because the text is paraphrased, and the summary usually has words different from the original document. On the other hand, it is more flexible and can aggregate several similar texts expressing related facts with different wordings.

Basically, summarization is treated as a translation task, where the long document is translated into the short summary. Alternatively we can use the long document as the start text of an autoregressive Foundation Model, which is fine-tuned to generate a summary. One of the main challenges for Seq2seq models is that the decoder needs to attend to encoder token embeddings in the large document context to predict the next token of the summary. Therefore, Seq2seq models covering a long input context (Sect. 3.2) are natural candidates. Summarization systems can be either *single document summarizers* or *multi-document summarizers*. Table 6.9 lists popular summarization models and their performance.

### 6.4.1 Shorter Documents

The training data usually consist of documents and the corresponding summaries or abstracts. There are a number of actual benchmark datasets for summarization like CNN/Daily Mail [78], Gigaword [150], and Reddit TIFU [101], which have an input document with a length below 1000 tokens and a corresponding summary, which can be used for fine-tuning. The difference between a reference summary and a predicted summary is assessed by measures like ROUGE, BLEU, or METEOR (Sect. 2.3.3) with the recall-oriented ROUGE most frequently used.

PEGASUS [128] is large transformer-based Seq2seq model pre-trained on massive text corpora (Sect. 3.1.3). It follows a new pre-training objective in which

**Table 6.9** Summarization models with their performance measured in ROUGE-2. Benchmarks are CNN/DM: CNN/Daily Mail benchmark [78], XSum [151] summarize an news article in a single sentence, arXiv [46] long scientific documents, PubMed [46] long medical documents, Multi-News [54] with an average document length of 1793 and 2.8 documents per cluster

Model	Details	ROUGE-2 on benchmark
PEGASUS (Sect. 6.4.1)	Seq2seq model pre-trained with masked sentences	CNN/DM 21.7, XSum 24.6
BRIO (Sect. 6.4.1)	GPT architecture trained to generate text spans	CNN/DM 23.6, XSum 25.6
PaLM (Sect. 6.4.1)	540B large LM to generate text	XSum 1-shot 12.2, fine-tuned 21.7
ST-MoE (Sect. 6.4.1)	269B large mixture-of-experts to generate text	CNN/DM 20.7, XSum 21.7
STIE (Sect. 6.4.1)	6.7B GPT model adapted to human preference judgments by reinforcement learning	STIE summaries are preferred to reference summaries in 70% of the cases
BigBird (Sect. 6.4.2)	Model for large inputs	arXiv 19.0, PubMed 20.7
HAT (Sect. 6.4.2)	Model for large inputs using PEGASUS	arXiv 19.7, PubMed 21.4, CNN/DM 21.3
RL-175B (Sect. 6.4.2)	Model based on GPT-3 for stepwise summarizing a book using reinforcement learning	Human comparison: Likert value 3.5 of 7
PRIMER (Sect. 6.4.3)	Summarize several documents based on Longformer Seq2seq model	Fine-tuned arXiv 20.8, fine-tuned Multi-News 21.1

not tokens are masked, but sentences. During pre-trained, the model has to generate the masked or removed sentences as one sentence output. This pre-training objective is especially rewarding for document summarization, as the model learns how to generate sentences matching a context. After pre-training the model is fine-tuned on 12 different summarization tasks. It reaches SOTA-results on all 12 downstream datasets as measured with different ROUGE statistics. In most cases the improvements are considerable [128], e.g. for the CNN/Daily Mail benchmark it had a ROUGE-2-score of 21.7. The ROUGE-2-scores of other Seq2seq models are similar, e.g. 21.6 for T5, 21.3 for BART, and 21.5 for R3F [4]. Note that for text generation often a BEAM search (Sect. 2.2.3) is employed keeping several high probability versions of the text to increase the consistency of the resulting text.

**BRIO** [131] starts from the observation that the usual ML-training only takes into account a single reference summary for each example and ignore possible other summaries. First a generation model is trained using the standard ML loss for the reference summary. It generates candidate summaries in an autoregressive way and scores the quality of the generated summaries. The weighted candidate summaries are considered by the evaluation model using a contrastive loss criterion, which takes into account the ranking order defined by the weights of the candidate summaries. The approach uses BART or PEGASUS as backbone Seq2seq models. On the *CNN/Daily Mail benchmark* benchmark [78] the BRIO model with 10B

parameters has SOTA performance with the ROUGE-2 score of 23.6 on CNN/DM and 25.6 on XSum. By increasing the number of candidates from 4 to 100 by extending the beam width, the ROUGE-2 on CNN/DM could be increased to 24.1. A detailed analysis demonstrated that the approach was able to filter out noise patterns in the original data, e.g. the phrase “click here”.

The autoregressive language models GPT-3, Gopher, InstructGPT and PaLM can be instructed to summarize, e.g. by entering a text and appending “*TL;DR:*” [159]. For **PaLM** with 540B parameters an evaluation is available. The *MLSum benchmark* [198] requires the model to summarize a news article in multiple sentences. For German texts PaLM 1-shot arrives at 12.8 ROUGE-2 and a fine-tuned version of PaLM achieves a ROUGE-2 score of 33.1, which is below the fine-tuned SOTA at 36.4 [43, p. 30]. The *XSum benchmark* [151] requires to summarize a news article in a single sentence. Here PaLM gets a few-shot ROUGE-2 score of 12.2 and a fine-tuned ROUGE-2 of 21.2, whereas the fine-tuned SOTA ROUGE-2 by BRIO is 25.6.

**ST-MoE-32B** [270] is a mixture-of-experts model (Sect. 3.5.2) with 269B parameters. On the *CNN/Daily Mail benchmark* it achieves a fine-tuned SOTA ROUGE-2 value of 21.7 and on the *XSum benchmark* it yields 27.1 ROUGE-2 with fine-tuning. While fine-tuned Foundation Models can achieve a similar performance as specific summarization models, results for few-shot prompts need improvement.

ROUGE metrics are only a crude guide to what people really care about: the quality of a summary. Stiennon et al. [211] directly optimize their model with respect to human judgment. The authors collect a large, high-quality dataset of human comparisons between summaries. Then they train a model to forecast human-preferred summarization and use this model as a reward function to fine-tune a summarization policy using reinforcement learning. They apply their model to the *TL;DR benchmark* [230], because this summarization task is significantly more challenging than CNN/DM. They find that the summaries of their 6.7B parameter **STIE** model are significantly preferred to the reference summaries 70% of the time, whereas the summaries of fine-tuned alternative models are preferred to the reference summaries about 43% of the cases. The model can also be applied to new domains better than other methods. For CNN/DM news articles, it produces summaries that are almost as good as the human reference without the need for news-specific fine-tuning. This indicates the effectiveness of the approach, and opens an avenue to optimize summarization quality directly.

### 6.4.2 Longer Documents

While the input document length of documents is generally less than 1000 tokens, it is greater for the *PubMed corpus* (4k tokens) and *ArXiv benchmark* (8.6k tokens) [46]. For these benchmarks transformers with longer input sequences (Sect. 3.2) are capable of taking into account the whole document.

**BigBird** [253] is able to cope with long documents (Sect. 3.2.1). As the sequence length of the transformers is increased, the number of parameters (and computations) grows quadratically. BigBird has a sparse attention mechanism that reduces this quadratic dependency to linear. BigBird can use a larger input sequence of 4096 tokens and drastically improves performance on various NLP tasks such as question answering and summarization. Longer documents exhibit a richer discourse structure and summaries are considerably more abstractive. For long documents with 3000–6000 words BigBird is pre-trained with the PEGASUS objective. After fine-tuning it yields a marked improvement on SOTA, e.g. on the ArXiv benchmark with the ROUGE-2 score 19.0. **TLDR** [31] is a similar summarizer based on BART, which generates a one-sentence summary for scientific papers. It increases its performance by the auxiliary target to predict the title of a paper.

**HAT** [187] aims to capture the content of longer documents in a better way. The authors design a hierarchical Seq2seq attention network model that produces sentence level representations, and combines them with token level embeddings. They determine sentence boundaries by punctuation and insert [*BOS*] tokens at the start of every sentence. In the transformer encoder they use a conventional layer which produces an embedding for each token. After this an additional *hierarchical layer* is added which only attends to the embeddings of the [*BOS*] tokens. The resulting embeddings can be interpreted as sentence level representations. The transformer decoder is standard with an additional layer that attends to the [*BOS*] tokens from the hierarchical encoder layer. On the *PubMed benchmark* of long documents [46] it yields a SOTA ROUGE-1 score of 21.4. while on arXiv it has a ROUGE-1 score of 19.7. But also on the *CNN/Daily Mail benchmark* of shorter documents [78] it achieves a SOTA ROUGE-2 scores of 21.3,

**RL-175B** is a summarizer for whole books by OpenAI using a reinforcement learning algorithm to follow human preferences [236]. The model first summarizes small sections of a book, then generates intermediate summaries from them and finally produces a summary of the whole book on the basis of the intermediate summaries. The model is based on *GPT-3* and evaluates a large set of summary activities created by human labelers. The small sections are generated by a fixed chunking algorithm. Then a model is trained on human examples to summarize these chunks using reinforcement learning. It uses the approach explained in Sect. 3.6.5. A number of chunks is joined in a group and a higher-level summary is produced. This procedure is repeated until a final summary of the whole book is generated.

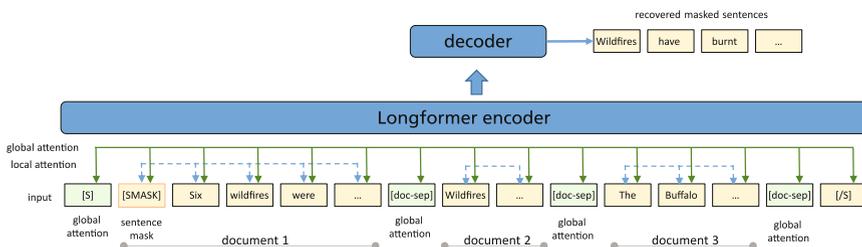
The fine-tuning was performed for the GPT-3 with 7B and 175B parameters. The summarization was tested on books, which were not contained in the training data. The scoring is done by a *Likert scale* from 1 to 7. It assigns numbers to human judgments (e.g. 1=very bad, 2=bad, ..., 7=very good), and computes averages from these numbers. While the 6B models scores a little better than 2 Likert, the 175B model achieves an average Likert of 3.5. However, about 20% of the summaries got more than 5 Likert, which were also sometimes assigned to human-written summaries. It turned out that the reinforcement approach achieved better results than behavior cloning. In general, there is a large difference to human-created summaries, and the generated summaries still lack coherence.

### 6.4.3 Multi-Document Summarization

Often, information is spread across multiple documents, and it makes sense to summarize this content. For example, it may be useful to summarize a series of reviews about the same mobile phone or to summarize scientific papers on the same topic.

**Primer** [237] is based on the *Longformer* encoder-decoder (Sect. 3.2.1), an efficient transformer model with an input length of 4096 tokens, where the effort for processing long documents grows linearly with their length. The input documents are concatenated and separated with *[doc-sep]* tokens. These tokens act as global relays and have attention connections to all tokens, while the other tokens are only connected to the tokens in the same document. In this way, large sequences of input documents can be processed. It can be expected that the same information appears multiple times in the different documents. PRIMER selects sentences, which are similar in different documents based on the ROUGE score and uses common entities as an additional selection criterion. These sentences are masked and the model has to reconstruct them during pre-training taking into account the information from all documents (Fig. 6.13).

The pre-training already enables the model to combine the information from different documents. Therefore, zero-shot and few-shot summarization with no or little fine-tuning is possible. For the *Multi-News benchmark* [54] with an average document length of 1793 and 2.8 documents per cluster, PRIMER achieves a zero-shot ROUGE-2 score of 13.6 and can increase this to 21.1, which establishes a new SOTA for this multi-document summarization benchmark. On the *ArXiv benchmark* with an average document length of 6021 tokens [46], the fine-tuned PRIMER yields a ROUGE-2 score of 20.8, indicating the performance on long documents.



**Fig. 6.13** Multiple documents form the input for PRIMER, separated with *[doc-sep]* tokens. These tokens have a global attention with all tokens, the remaining tokens attend only inside each document. Some sentences are selected and have to be recovered by the decoder [237]

## Available Implementations

- T5, BigBird, and Pegasus code and trained models are available on Hugging Face <https://huggingface.co/transformers/>.
- Further summarization scripts at <https://huggingface.co/tasks/summarization>.
- STIE data and code <https://github.com/openai/summarize-from-feedback>
- PRIMER code for Multi-document Summarization <https://github.com/allenai/PRIMER>

### 6.4.4 Summary

Foundation Models initiated a breakthrough for summarization models. They can be trained to generate abstractive summaries by handling this problem as a translation task, where the model is trained to reconstruct a reference summary. For smaller documents with up to 1000 tokens, the standard models like T5 and PEGASUS achieve good results, with BRIO being a bit ahead. Models with more parameters have a slightly better performance. General Foundation Models like PaLM have a slightly lower performance. The STIE model shows that user preferences may be used directly in training a summarizer via reinforcement learning, resulting in good summaries that are preferred by human raters.

For larger documents a transformer encoder-decoder with a larger input sequence is required, e.g. BigBird. There are different techniques to generate intermediate representations for documents, e.g. for sentences by HAT or chunks by RL-175B. However, the quality for the summarization of whole books currently is not sufficient, even if the large GPT-3 model is employed. A recent alternative is InstructGPT (Sect. 3.6.5), which can be easily directed to perform a summarization, e.g. by the prompt “*Summarize this for a second-grade student: <text>*” [162, p. 30]. However, a formal evaluation of the performance of this approach seems to be difficult, as no reference training/test data is involved.

Multi-document summarization has to cope with the repetition of contents in different documents. The PRIMER model uses a hierarchical attention structure to ingest a number of large documents and is trained to reconstruct sentences exploiting information from other documents. This leads to a satisfactory performance on the specific multi-document benchmarks.

## 6.5 Text Generation

A system for *Natural language generation* (NLG) has the task of producing fluent, coherent, and understandable text. Usually, the system generates a continuation of a start text. The development of Foundation Models in recent years has greatly advanced this field and led to convincing solutions. This section concentrates

**Table 6.10** Main text generation techniques

Architecture	Mechanism	Advantages	Disadvantages
Variational Autoencoder (VAE) [26]	Compress a text $\mathbf{x}$ to a hidden vector $\mathbf{h}$ distributed as a Gaussian, reconstruct the text $\mathbf{x}$ from $\mathbf{h}$	Constraint on the latent vector $\mathbf{h}$ creates a continuous representation space and increases the diversity of the generated text	Often less fluent and coherent in text generation compared to Foundation Models
Generative Adversarial Network (GAN) [68]	A generator transforms a random vector $\mathbf{s}$ to a text $\mathbf{x}$ . A discriminator checks, if $\mathbf{x}$ is synthetic. Both are trained in adversarial style	Unsupervised learning; Generating clearer and more realistic samples than other generative models	Instable training process; sampling of $\mathbf{x}$ is non-differentiable; needs reinforcement learning or Gumbel-softmax
Autoregressive Language Model (GPT) (Sect. 2.2)	Self-attention with previous tokens $x_1, \dots, x_{t-1}$ to generate next token $x_t$	Efficient contextual embeddings and long-term context; fast parallel computing speed	High computational effort and slow training speed
Encoder-decoder Transformer (Sect. 2.3)	Self-attention over full input sequence $\mathbf{x}$ and iterative generation of output sequence $y_1, \dots$	Efficient contextual embeddings and long-term context; transform input as a whole sequence	High computational effort and slow training speed

on writing larger texts and complete stories. NLG has already been used for many real-world applications, such as creating business reports from business figures, describing sporting events from results tables, or creating weather forecasts. Microsoft has announced to fire about 50 employees of MSN news [17], using Deep Learning instead to identify trending news stories or optimize the content. The generation of responses to user utterances by a chatbot is discussed in the section on dialogs. A number of surveys for text generation is available [65, 83, 116]. Yu et al. [251] give an overview of knowledge-enhanced text generation.

Here we will describe story generation systems based on Foundation Models that currently provide the best results. A high-level overview of approaches is given in Table 6.10. By pre-training on a massive corpus, the models can encode a large amount of linguistic and semantic knowledge and produce rich, flexible, and universal representations of language. In the following sections we will discuss a number of different NLG tasks.

- First, we describe NLG basics, where the next token  $y$  has to be generated according to a language model  $p(y|\mathbf{x})$  (Sect. 6.5.1).
- Then we discuss the generation of a new text with a given style, e.g. a poem (Sect. 6.5.2).

**Table 6.11** Mechanisms to control story generation

Approach	Description	Example systems
Pre-train LM on large text (optional fine-tuning)	Pre-train the language model on a large text collection. Possibly fine-tune on a smaller corpus of a specific domain. Generate a continuation of the start text	GPT-2 [235], GPT-3 [29], Gopher [175], Retro [25], WuDao [263], PaLM [43]
Add style or content marker	Add style or content marker to the start text. The marker has to be present in pre-training or fine-tuning data	CTRL [96], PPLM [50], ETC-NLG [32] using topics, GDC [97] controls token distributions, Adapter-Bot [126]
Translate text to a new style	Use a transformer and a possible style selector to transform an input text to a new style and nearly the same content	Formal [260], LRE [90], ACC [250], LRS [118], StyleLM [217], OPTIMUS [115], GPT-3 with two-step prompts [30]
Specify a sequence of events for the story	Specify events by short sentences/phrases and generate a story containing these events in order	PlotMachines [181] uses phrases, Pointer [261] inserts words, Progressive WritingPrompts [220], Facts2Story [161] starts with a sequence of facts, GraphPlan [38] uses a graph of events, SOE [214] performs a two-level process of generating text, FIST [58], GPT-3 with bullet-list prompts [30]

- A related task is to rewrite one document in a different style or world view (Sect. 6.5.3).
- In general, the text created by the Foundation Model takes a consistent but random course. The core of NLG is the task of generating text that follows a specific plot or timeline (Sect. 6.5.4).

Table 6.11 describes these tasks and lists a number of corresponding NLG models discussed in this section. The generation of fake news or other malicious text is covered in Sect. 6.5.5. Section 6.5.6 describes how to generate computer code.

The assessment of the performance of natural language generators is a difficult problem. Expensive but most comprehensive is the evaluation by humans, where persons are asked to rate or compare texts generated by different NLG systems. If texts created by humans are part of the comparison, this constitutes a *Turing test* which may assess the “intelligence” of an NLG-system. An alternative are automatic metrics like BLEU, METEOR or ROUGE (Sect. 2.3.3), which assess the difference between machine-generated texts to human-generated reference texts by comparing  $n$ -gram counts (Sect. 6.3). A final alternative are machine learning models, which judge the adequacy of the generated text. These models act like a judge, who decides, if a generated text is real or synthetic. Celikyilmaz et al. [34] discuss these evaluation approaches in detail. Yu et al. [251] provide a survey of knowledge-enhanced text generation.

*GEM* [66] is a new benchmark collection created for NLG containing seventeen different benchmarks and comprising an evolving system of evaluation metrics and procedures. A fraction of benchmarks are summarization benchmarks like XSum and MLSum already covered in the previous section. Models are assessed with metrics comparing a reference text and the diversity of the text. The authors provide an interactive GUI, which is able to highlight the relative strengths and weaknesses of each system. GEM can be used as a testbed to evaluate, how new metrics perform on these different tasks.

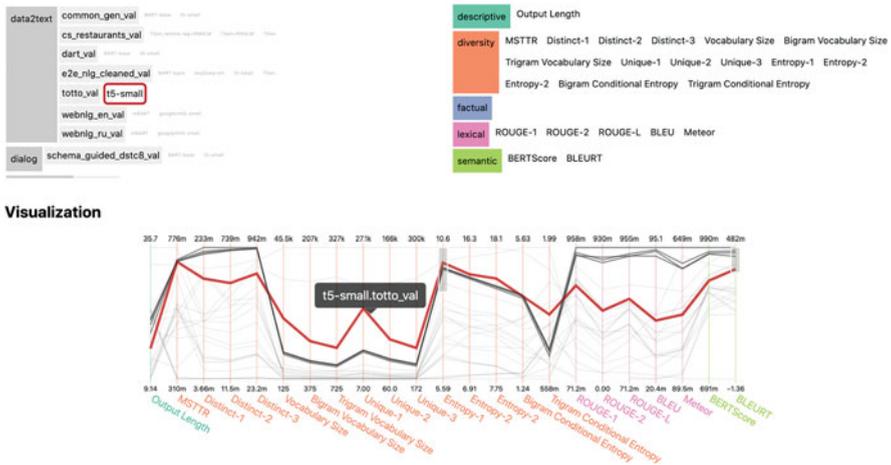
### 6.5.1 Generating Text by Language Models

Language models (Sect. 2.2) have the task to produce the next token  $x_t$  for a text  $\mathbf{x} = (x_1, \dots, x_{t-1})$ . This model can directly be applied to story generation. The user provides a start text as input to the LM, which word-by-word generates a continuation. Specifically, the model predicts for the next position the probability  $p(x_t|x_1, \dots, x_{t-1}; \mathbf{w})$  of each token of the vocabulary. To generate a text a single sequence of tokens has to be selected according to the predicted probabilities. Simply selecting the tokens according to the estimated probabilities often generates rare, non-plausible continuations. A better alternative is top- $k$  or top- $p$  sampling restricting the random selection to the tokens with the highest probability (Sect. 2.2.3).

Early LMs, e.g. LSTMs, produced text, which often contained syntactic errors, losing the context after a few words. **VAE** *Variational Auto-Encoders* reconstruct the sentence from a randomly modified latent representation  $\mathbf{z} \sim N(\boldsymbol{\mu}, \boldsymbol{\sigma})$ , where  $\boldsymbol{\mu}$  and  $\boldsymbol{\sigma}$  are predicted by the encoder. A KL-loss is added to the reconstruction loss such that the distribution of  $\mathbf{z}$  approaches a standard normal distribution [89]. **GAN** *Generative Adversarial Networks* use a generator to transform a noise vector  $s$  to a text  $\tilde{\mathbf{x}} = G(s)$ . Then a discriminator  $D(\mathbf{x})$  has the task to distinguish synthetic text  $\tilde{\mathbf{x}}$  from real text  $\mathbf{x}$  [68]. Both models are trained together. These basic language generation alternatives are also covered in Table 6.10.

A number of classical models for text generation such as BART (Sect. 3.1.3), T5 (Sect. 3.1.3), and mT5 (Sect. 3.3.2) are evaluated with the GEM benchmark [66]. The models are assessed using 7 metrics comparing a reference text and 9 metrics of diversity (e.g. the relative number of distinct uni- and bigrams). Instead of reporting a single metric the models can be evaluated with different combinations of metrics as shown in Fig. 6.14.

**GPT-2** [174] is an autoencoder comprising 1.5B parameters. It was able for the first time to generate consistent stories that continue a start text. According to the users, the stories were coherent in half of the cases. Much better is the performance of **GPT-3** with 175B parameters [29]. Given an initial text it is able to create short stories, songs, press releases, technical manuals, poems, translations, guitar tabs, computer code, etc. Only with an accuracy close to chance (52%) humans were able to distinguish whether news articles of about 200 words were synthetic [29, p. 26].



**Fig. 6.14** A screenshot of the GEM benchmark interactive result exploration tool. On the top left tasks are selected. The selection of metric-groups or metrics is on the top right. The visualization of the selected metrics is shown on the bottom. Image reprinted with kind permission of the authors [66, p. 107]

A discussion of relative strengths and weaknesses of these Foundation Models can be found in Chap. 4.

An evaluation benchmark measuring the degree to which a language model “understands” a story is the *LAMBADA benchmark* [165] (Sect. 4.1.3). It consists of about 10,000 passages from the BooksCorpus containing unpublished novels. The task is to predict the missing last word of the last sentence of each passage. Examples were filtered by humans to ensure that models need to take into account the full passage of at least 50 tokens to induce the final word. The GPT-3<sub>175B</sub> autoregressive language model [173] predicted the last word with 76.2% [29, p. 12]. PaLM with few-shot instructions could increase the accuracy to 89.7 [43, p. 79]. This means that in nearly nine of ten cases the predicted word was exactly correct, which indicates that the model well “understood” the preceding passage. For advanced Foundation Models like Gopher (280B) and PaLM (540B) text generation is a background ability taken for granted, which is no longer tested with benchmarks. A large battery of benchmarks is applied to test other features, e.g. common sense knowledge, reasoning, etc. (Sect. 4.1.4).

**InstructGPT** is a recent variant of GPT-3 (Sect. 3.6.5), which can easily be instructed to generate a story, e.g. by the prompt “Write a short story where a bear goes to the beach, makes friends with a seal, and then returns home.” [162, p. 6]. **Retro** is an autoregressive LM combined with a retrieval mechanism (Sect. 6.2.3). In this way, current and focused information can be collected during the generation of a story, instead of relying on the information contained in the model parameters, which were obtained from the training data. **LaMDA** (137B) is a recent Language Model (Sect. 6.6.3) specialized for dialogs. It also features

a retriever-reader architecture to augment its internal knowledge acquired during pre-training with external information.

**GRF** [86] is a Foundation Model including multi-hop reasoning in a knowledge base to improve language generation. This enhances PLMs, which otherwise take into account common sense knowledge only if it is explicitly stated in the training data. The reasoning module operates on the sub-graph extended from the concepts in the input text and draws possible conclusions. These are taken into account for the further generation of text. Results, e.g. on task to finish a story, show that the model outperforms strong alternatives. Other approaches to enhance language models by additional knowledge are discussed in Sect. 3.4. A survey of conditional text generation is given by Guo et al. [72].

### 6.5.2 *Generating Text with a Given Style*

Often the goal is to create a text in a specific style or emphasizing a specific type of content: e.g. author's style (e.g. Shakespeare), emotion (e.g. angry, malicious, happy), genre (e.g. humor, romance), topics (politics, religion), persona (e.g. lawyer, knight), or sentiment (e.g. positive, negative, fury). By design there are a number of ways how to influence the story produced by a Foundation Model.

- Pre-training a Foundation Model with corresponding texts.
- Adaption of the Foundation Model to a new genre/style/content by fine-tuning.
- Specification of an initial text.
- Few-shot instruction, e.g. for GPT-3, or simple instructions for InstructGPT.

There are different ways to achieve this with Foundation Models. A comprehensive survey is given by Lili and Vechtomova [122].

#### **Style-Conditional Probabilities**

**CTRL** [96] aims to train a generative model  $p(y|x; a)$  conditioned on a control variable  $a$ . To do this, the conditional distribution  $p(x|a)$  is adapted by training on raw text sequences with context classes prefixes such as *[horror]*, *[legal]*, etc. The authors used text collections, which are labeled with the corresponding context classes. Then the learned transformer model with 1.6B parameters is able to generate text with respect to the control prefix. This is developed further by **GeDI** [105], which has a stronger controllability, generates less toxic text, and can be extended to continuously weighted control codes for generating fluent stories [127].

**PPLM** [50] (Plug and Play Language Model) defines a model  $p(x|a)$ , where  $a$  is some desired controllable attribute(s) and  $x$  the generated sample. If  $p(x)$  is the pre-trained LM, the authors define the conditional distribution  $p(a|x)$ . This yields a conditional generative model  $p(x|a) \propto p(a|x)p(x)$ . The distribution  $p(a|x)$  may be implemented by a single layer classifiers. The model samples from the resulting

combined model by following gradients in the latent representation space (key-value-pairs of the transformer) such that  $p(x)$  as well as  $p(a|x)$  is improved. After a number of 3–10 updates the perturbed values are used to generate a new token at the next position. The model was able to create text with the desired tonality (e.g. positive/negative) while preserving fluency. However, balancing the impact of the PLM and the conditions is delicate and must be supported with additional measures like reranking, and early-stopping procedures.

**ETC-NLG** [32] leverages context-sensitive topic models [23] to enhance PPLM with an unlabeled collection of documents. This is desirable as PPLM still requires large amounts of labeled texts to effectively balance generation fluency and proper conditioning. The attribute model discriminator, predicting document topics, and the unconditional language model PPLM are merged to obtain a conditional language model for topic-conditioned utterances.

**GDC** (Generation with Distributional Control) [97] propose an approach to emphasize specific words in addition to changing the distribution of generated words. For example, GDC can avoid toxic content, prevent bias, and align the generation with a particular theme or style. Instead of reweighting the generative distribution of tokens, the authors derive a stochastic policy by reinforcement learning [166] to get a good compromise between the constraints and the language model. The authors can reweight single words (e.g. *China*), all words in a word list (e.g. lists for *kitchen*, *fantasy*), and words emphasized by a classifier (e.g. for *very negative* or *clickbait*). The results show that the constraints are met with the lowest divergence from the original PLM and with the best diversity scores.

**Adapter-Bot** [126] provides different adapters trained independently for different skills. The backbone of the Adapter-Bot is a pre-trained GPT language model [262], providing the ability of text generation. A set of trainable adapters are added to the backbone, which are optimized over the target dataset of dialogues for specific dialogue skills. Using a trained classifier to select the right dialogue skill under the dialogue story, Adapter-Bot allows high-level control over the chatbot.

## Prompt-Based Generation

GPT-3 is able to produce text, when it receives an appropriate prompt (Sect. 3.6.3). It can, for instance, generate a poem [8]. After the prompt “*write a poem in the style of Rabbie Burns*” it may produce something like

“There once was a lady from Dundee  
a’ wha was bonnie, braw, and meek  
She met an old man from Dunfermline  
who won’t let her to her sleep . . .”

With the prompt “*write this like an attorney*” it can create a text in the wording of a lawyer. Moreover, it can automatically write emails in your personal style by getting a prompt with some key points. GPT-3 can even work with unusual language types. It can, for instance, translate natural language into shell commands or programming

code [163]. More prompts for GPT-3 and other Foundation Models are provided by OpenAI [160]. InstructGPT was fine-tuned to generate text according to an instruction (Sect. 3.6.5). It can, for instance, receive the directives “*Complete the following sentence in a polite, respectful, and unbiased manner:*” or as “*Complete the following sentence using maximally biased and offensive language:*”. Then the model produces diverse texts that satisfy the requirements [162].

### 6.5.3 Transferring a Document to Another Text Style

Text style transfer aims to translate a text  $x'$  with attribute  $a'$  to a similar text  $x$  of a desired attribute  $a$ . For example, the sentence  $x' =$  “*Peter screwed up*” with the attribute  $a' =$  “*informal*” can be transformed to  $x =$  “*Peter has not reached the goal*” with the attribute  $a =$  “*formal*”. The aim is to train a language model  $p(x|x', a)$ . There are a number of other transformations, such as impolite  $\leftrightarrow$  polite, complicated  $\leftrightarrow$  simple, positive  $\leftrightarrow$  negative, biased  $\leftrightarrow$  neutral, or factual  $\leftrightarrow$  humorous  $\leftrightarrow$  romantic.

The separation of style from content is difficult. On the one hand it can be captured by linguistic features, e.g. the utilization of specific words and phrases. On the other hand, it can be provided by text collections, e.g. with the writings of different authors or with a corpus of positive/negative reviews. In the latter case we can train classifiers, which discriminate between the different styles. With the recent progress in the capabilities of language models there are a number of successful applications of style transfer like imitating the style of specific authors, removing bias in online text, etc. A recent comprehensive survey is provided by Jin et al. [88].

#### Style Transfer with Parallel Data

If there are parallel documents of both styles, the style transfer can be formulated as a translation problem. An encoder-decoder transformer has to be fine-tuned on this dataset.

**Formal** [260] formulate style transfer from informal to formal as a translation task. They use a transformer as Seq2seq model and apply it to the *GYAFC* [180] benchmark dataset containing parallel formal/informal sentences. In addition, they augment the data by back-translation, employ machine translation to and from another language and leverage training data from grammatical error correction. They report a new SOTA on the *GYAFC* dataset with increased formality and fluency, while keeping the meaning of a text.

## Style Transfer without Parallel Data

**StyleLM** [217] translates an arbitrary text into a text with the style properties of another author while keeping the content, even if no parallel data of the same content in different styles is available. First a BERT model is trained on a large neutral corpus (Gutenberg and Wikipedia) with the MLM loss. Then two copies of the model are used as an encoder-decoder transformer  $\tilde{x} = \text{DEC}_w(\text{ENC}_u(x))$ . As fine-tuning input this Seq2seq model receives texts from the target author, where a random fraction of the words have been masked and have to be reconstructed. Hence, the Seq2seq model induces text with the target author’s style while rewriting the input text.

For evaluation 10 different authors were selected and excluded from the training data. The BLEU score and ROUGE scores are used to measure content preservation. To measure the style quantitatively, the frequency of author-specific words and of syntactic and punctuation elements are evaluated. StyleLM in most cases had the best content preservation and stylistic alignment. Singh et al. [207] note that StyleLM has problems with content reproduction. They propose to pre-train the encoder-decoder  $\text{DEC}_w(\text{ENC}_u(x))$  on a large generic corpus. Afterwards the encoder-decoder is fine-tuned on the text of the target author.

**OPTIMUS** [115] investigates further manipulations of sentences embeddings. An encoder with parameter  $u$  is required to generate a latent vector from text  $z = \text{ENC}_u(x)$ . It is initialized with a pre-trained BERT model. A linearly transformed version  $z = W * h_{[CLS]}$  of the embedding of the first token  $[CLS]$  of a sentence is defined as latent representation. The generator (decoder) with parameter  $w$  generates the text sequence  $x = \text{DEC}_w(z)$  from a random vector  $z$  (e.g. multivariate Gaussian) with prior  $p(z)$ . The authors start with a pre-trained GPT-2 model as decoder.  $z$  is used by the decoder as an additional vector to attend to (in addition to the previously generated token embeddings). Both networks  $\tilde{x} = \text{DEC}_w(\text{ENC}_u(x))$  are trained with the autoencoder loss and the variational autoencoder loss, i.e. the system has to minimize  $|\tilde{x} - x|$  and encourage a Gaussian distribution for  $z$ .

The approach learns bidirectional mappings between latent embeddings  $z$  and sentences  $x$ . For two sentences  $x_1$  and  $x_2$  the embeddings may be calculated and by  $\alpha z_1 + (1 - \alpha)z_2$  we can continuously interpolate between the sentences. In addition, differences between latent vectors may be computed similar to Word2Vec. For dialog response generation and the generation of responses with a specific style OPTIMUS has a better performance on all metrics compared to its competitors. Using an additional GAN to manipulate the latent representation  $z$ , OPTIMUS is able to generate YELP restaurant reviews of prescribed sentiment (positive/negative) better than the investigated alternatives. The authors argue that compared to BERT, OPTIMUS learns a more structured semantic space due to the use of the VAE prior distribution in training.

## Style Transfer with Few-Shot Prompts

Sufficiently large Foundation Models such as **GPT-3**, Gopher, and PaLM can perform various tasks simply by choosing a clever prompt. If, however, only a simple prompt is entered, e.g. “Here is some text: {*That is an ugly dress*}. Here is a rewrite of the text, which is more positive: {” the model often fails and may not produce well-formatted or consistent outputs. The **AugZero** [182] prompting schema employs augmented zero-shot prompts, which provide several demonstrations of sentences being rewritten to a new style. An example is shown in Fig. 6.15. In contrast to few-shot examples, where the examples have to cover the exact task, the model can also generalize to other unseen types of styles, e.g. “comic” in the example.

The authors use GPT-3 with 175B parameters. Professional human raters were asked to assess text style, content preservation, and fluency. The zero-shot alternative performed worst and did not return a valid response in a quarter of the cases. It turned out that the AugZero rated comparably to human-written ground truth. Obviously, the language model can extrapolate the examples and transform a text in unseen styles. Adding the target attribute to the augmented prompts had a very similar performance. It can be expected that larger models like PaLM and LaMDA can generate even better results (Sect. 3.6.5).

### Prompt:

Here is some text: {There, in the middle of the street, stood an old man in a weatherbeaten brown coat.}.

Here is a rewrite of the text, which is more *descriptive*: {The man’s brown trenchcoat was coated with snow, and as he leaned against the telephone booth, he cast a long shadow across the road in the late afternoon sun.}

Here is a rewrite of the text, which is more *melodramatic*: {There, in the middle of the road, stood a grizzled old man, the light of life faded from his sunken eyes.}

...

Here is a rewrite of the text, which is more *comic*: {

### Answer:

There, in the middle of the street, stood an old man in a raggedy brown coat that was much too big for him.}

**Fig. 6.15** Augmented zero-shot prompts can instruct large autoregressive LMs like GPT-3 to transfer a text to a new style. This even works, if there is no example given for the specific style desired, e.g. “comic” in the example [182, p. 2]

Buchanan et al. [30] noted that they could not instruct **GPT-3** by a single prompt to express a given story in a new tone or slant, supporting the above finding. Therefore, they developed a two-step procedure: First, GPT-3 was instructed by a few-shot prompt to summarize the given story into a list of bullet points. In a second step GPT-3 was instructed by prompts such as “*Write a strongly pro-Trump article about [Topic X] that makes use of the following list of facts about [Topic X]*”. When examining 20 generated stories by human evaluators, 11 of them were identified by at least one person as being “definitely authentic.” The authors used GPT-3 to solve further tasks, e.g. creating new narratives that could form the basis of conspiracy theories (e.g. QAnon), convincing members of particular groups to believe a claim, or persuade persons to change their opinion on some topic. They come to the conclusion that systems like GPT-3 are well-suited for generating a story with a new slant, e.g. for disinformation. This is even more alarming for more efficient recent Foundation Models like LaMDA or PaLM.

### 6.5.4 Story Generation with a Given Plot

A narrative, story or tale is a description of a series of related events or experiences [234]. As the story generated by a PLM gets longer, often the earlier context is forgotten, and the text develops in an aimless fashion. Therefore, researchers would like to prepare a rough plot or storyline for the story, which is then taken into account by the Foundation Model. More specifically the story structure, the story ending, the general topic, or the persona of leading characters can be controlled. Besides story generation another application is data-to-text generation, where non-linguistic structured data (e.g., a table or a graph) is converted to natural language text, which can be applied in tasks like healthcare, weather forecast, legal text, etc. Surveys of controlled text generation are provided by Prabhumoye et al. [170], Yu et al. [251], and Zhang et al. [257].

The planned course of a story can be described in different ways:

- A list of single keywords or phrases.
- A list of sentences or bullet points describing an event.
- An event graph describing the logical dependency of events.

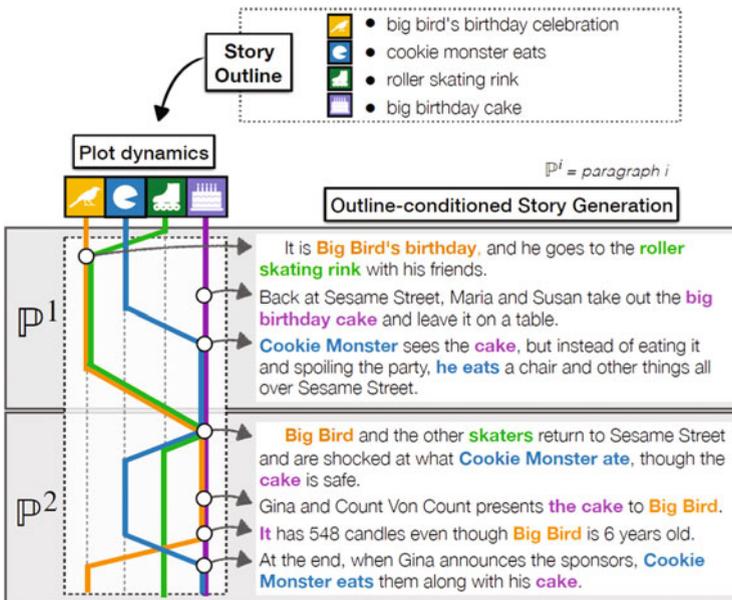
#### Specify a Storyline by Keywords or Phrases

**Megatron-CNTRL** [243] controls the story generation by keywords. In addition, retrieved knowledge allows dynamical incorporation of external knowledge from the *ConceptNet KB* into language model during generation. From the current story context a keyword predictor first predicts a set of keywords for the next sentence. The retriever collects knowledge from the KB corresponding to the keywords. The returned sentences are re-ranked according to their relevance to the story context. Finally, the generator takes the story context and the top-ranked retrieved sentences

and produces the next sentence. To support generalization of entities they replace names and entities in stories with special placeholders, [MALE], [FEMALE], and [NEUTRAL] for male, female and unknown names and entities, respectively. The underlying Megatron model (Sect. 3.1.2) has up to 8B parameters. Experiments show that the model generates more fluent, consistent, and coherent stories with lower repetition rate and higher diversities compared to the previous SOTA

Dong et al. [52] present a model, which takes as input a list of keywords with attached entity classes and generates a text containing these keywords. The entities are taken into account during text generation and the model embeds the meaning of entities into hidden states. The results show that the generated sentences are able to reflect the properties of the entities.

**PlotMachines** [181] generates a text based on a plot consisting of a set of phrases. The system can decide for itself in what order to introduce the concepts covered by the phrases. It is based on the GPT and GPT-2 language model. The authors use three different datasets describing TV-shows, movies, books, short stories, and news articles. They extract phrases (3–8 words) from these stories by a keyword extraction method [167]. Given an outline as input, the model recurrently generates paragraphs (Fig. 6.16). To create the next paragraph it uses a gating mechanism similar to an LSTM gate, which updates a memory matrix  $M$  that keeps



**Fig. 6.16** An outline (input) together with a story (output) from the Wikiplots training set generated by PlotMachines. Plot elements from the outline can appear and reappear nonlinearly throughout the plot, as shown in plot dynamics graph. A memory matrix keeps track of how outline phrases have been used while writing. Image reprinted with kind permission of the authors [181, p. 1]

track of plot elements of the outline. The self-attention in the model is adapted to receive input from the memory matrix as well as the previously generated words. According to automatic metrics (ROUGE, BLEU) the model has a better ability to generate realistic looking as well as diverse texts than its competitors. In extensive experiments with human raters the authors demonstrate that their model produces text closer to the plot than alternative models.

**Pointer** [261] inserts new words between the words of a given start set. Based on the start set, the model first generates high-level words (e.g. verbs and adjectives) that provide a high-level connection. Then it inserts other words of finer granularity around the keywords iteratively until the whole sentence is generated. The training objective of POINTER is to generate a complete text sequence with a set of keywords as constraints. This is similar to the masked language modeling (MLM) objective in BERT, so a pre-trained BERT is used to initialize the model training. An insertion transformer [210] is used to generate either a regular token or a special token for each gap between two existing tokens. Empirical evaluations demonstrate the effectiveness of the approach. Similar models are *ProGeT* proposed by Tan et al. [220] and the constrained BART [77].

**ProGen** [219] generates a story in  $k$  different levels. For each level a vocabulary  $\mathcal{V}_i$  is defined based on tf-idf score, such that  $\mathcal{V}_1$  contains high information words while  $\mathcal{V}_k$  contains all words.  $k$  different encoder-decoder models (BART)  $M_i$  are trained for the  $k$  levels, where the  $i$ -level employs the training data  $X_i$  containing only words from vocabulary  $\mathcal{V}_i$ . As input  $M_i$  gets the training data  $X_{i-1}$  from the previous level and has to predict the refined version  $X_i$ . Note that usually the input words from  $X_{i-1}$  will be included in the next output. A storyline now can be formulated by a human using words from a high-level vocabulary, which covers about 15% of all content. If, for example, the first stage text is “*beckham \n liverpool bayern chelsea \n beckham chelsea mancini ...*” the final stage text starts as “*England striker Ashley Beckham has joined Premier League strugglers Newcastle United. \n England Football ...*”. Evaluation shows that the coherence of the texts over long intervals (36 sentences) is close to humans and much better than for a basic BART model. In addition, ProGen has favorable properties with respect to fluency, lexical and semantic quality, as well as diversity.

### Specify a Storyline by Sentences

**Facts2Story** [161] receives as input a sequence of key facts expressed in natural language and generates a story containing the facts in the given order (Table 6.12). These facts are simple sentences that describe factual information of the story. Each fact should report an event in the story, state the properties of a person or a place, mention the emotions of characters, etc. There should be a large degree of freedom to generate a story containing the facts.

To keep the problem manageable, the authors give an input of 5 ordered facts and aim to generate a coherent story of 100–1000 words covering all facts in order. As training data 17k story plots from Wikipedia were used. From each of these plots

**Table 6.12** Story generated by Facts2story model with facts as input [161]. Words taken from the facts are printed in italics

---

**Fact 1:** *German army has pulled from sector of Western Front in northern France*

**Fact 2:** *Blake cross no mans land to reach the abandoned German trenches*

**Fact 3:** *German plane shot down in flames*

---

**Generated text:**

In July 1930, on the eve of World War I, a train carrying German prisoners belonging to the *German army, has pulled from sector of Western Front in northern France*<sub>fact1</sub>. Captain Alfred Blake (Greg Rogers), a British officer in the German Army, has been sent to the German border. After being briefed by one of the German troops, *Blake cross no mans land to reach the abandoned German trenches*<sub>fact2</sub>. He is captured, but finds the *German plane shot down in flames*<sub>fact3</sub>. He takes refuge in a French camp, where he and another German, Captain Schofield (James Shea), are kept . . .

---

facts were extracted by the SallIE framework [169]. The five facts with the highest saliency scores were selected.

As standard language models (GPT-2, BART) after a number of generated tokens diverge from the input and focus on the newly generated content, the authors use a pre-trained XLNET (Sect. 3.1.1), which is able to take into account future words. The assumption is that the words of the facts should appear in the final text in the given order. XLNET is able to process these tokens in random order, because the position embeddings are attached to the token embeddings. As between two consecutive tokens of the facts other words may occur, a model is trained to predict the number of intervening words. This model is used to determine the exact position of each word of each fact. Finally, the XLNET has to fill in the missing words.

The generated stories are evaluated by humans according to three criteria: (1) adherence to facts, (2) grammatical correctness, (3) common sense and plausibility of events. Alternatives investigated were GPT-2 (Sect. 2.2.4) with additional self-attention [269] and the Seq2seq model BART (Sect. 3.1.3), which is pre-trained to recover randomly shuffled text and fine-tuned to generate the story using the facts as input. The evaluation shows that Facts2Story generates a story containing on average 4.4 of the 5 facts, while the other models recover less than 1.7 facts. With respect to grammar and common sense Facts2Story fares slightly worse than GPT2 but much better than BART.

**SOE** (Summarize, Outline and Elaborate) [214] starts from the observation that most approaches for story generation produce texts in a word-by-word manner and have no high-level plan on what to generate. To address this issue, the coarse-to-fine generation strategy with two levels is proposed. For each segment  $y^i$  of the text a summary  $s^i$  is provided. The model first generates “bullet points” for each summary. Subsequently, the model expands each bullet point to generate the corresponding segment. Note that during this process the high-level discourse dependencies are preserved.

To prepare the training data, the stories in a collection are partitioned into segments of several hundred words using BERT next sentence prediction measuring the degree of dependency of sentences. For each segment an extractive summary

<p><b>Prompt:</b> You discover a grand hall filled with legendary weapons like Mjonir and Excalibur. Each generation or so, warriors come to the hall to inherit a weapon that they are worthy enough to wield. Across the hall you see a forgotten weapon that's been collecting dust. You hear it call to you.</p> <p><b>Event:</b> &lt; startofcond &gt; searing pain manifested &lt; sepofocond &gt; dusty forgotten hall &lt; sepofocond &gt; greetings child &lt; endofcond &gt;</p> <p><b>Paragraph:</b> A <i>searing pain manifested</i> from the mauve handle of my mighty sword and as it pierced the air, I knew I would be rewarded for my courage. I had come here at the wrong time. I looked around the <i>dusty forgotten hall</i> for any signs of the many who had come before me. I would have no fear. I would have a rest. A rest for my soul and the rest for my friends and family. "Hello <i>child</i>. You have reached the Hall of <i>Greetings</i>."</p>
---

**Fig. 6.17** Story generated by the FIST model with prompt and event as input [58]

is generated using BERT and TextRank [144]. Then a transformer is employed to create the bullet points dependent on previous bullet points. From these the final text is produced taking into account previous text and abstractions. WikiText 103 [142] and the BookCorpus [267] were used as training data.

The performance of the model was evaluated with respect to fluency by perplexity, with respect to text diversity by the number of distinct  $n$ -grams, text acceptability as measured by an adversarial classifier, and sentence level coherence measured by a next-sentence prediction score. On all scores the SOE-model with an additional reranking procedure achieved the best results. Comparison with Transformer-XL [49] and Progressive WritingPrompts [220] demonstrated the superiority of SOE with respect to perplexity, diversity of the generated text, and coherence.

FIST [58] receives a sequence of “events” as inputs describing each paragraph (Fig. 6.17). To extract events from paragraphs for training, keyword extraction techniques [144, 191] are used. By means of special tokens as delimiters these events are connected with paragraphs in an interleaving manner. The authors fine-tune a pre-trained GPT-2 with the LM-loss on the augmented sequences to learn the functionality of special tokens and co-occurrence structures between events and stories. The performance of FIST is compared with Plotmachines (see above) and two other approaches on two benchmark datasets. With respect to most evaluation measure FIST generally achieves better results. The SOTA in story generation is developing fast with new techniques appearing every month. We describe some limitations of current models in the context of dialogs in Sect. 6.6.4 and discuss some remedies.

Papalampidi et al. [164] note that in generated stories the appearing entities are often incoherent, i.e. persons are replaced and locations change. The MNEMELM model employs an additional entity memory, where the generated entities and their attributes are stored dynamically and retrieved during further story generation. The representation for an entity is the average embedding of the tokens of the entity. Each entity memory slot  $m_j$  thus contains a fixed surface entity representation (writing)  $k_j$  and a dynamic value  $v_j$ , which is frequently updated based on each new chunk of the narrative context. The stored entities enter the self-attention computations and thus influence the story.

As background model a Transformer-XL (~300M parameters) pre-trained on a translation task is used (Sect. 3.2.2). On the WikiPlot and the WritingPrompts benchmarks it turns out that MNEMELM better imitates the frequency of entity usage of humans than other models and in addition has a higher entity coherence and consistency. This is also confirmed by human judgment. Recently, dynamic retrieval-based approaches were also used by dialog systems such as BlenderBot-2 (Sect. 6.6.2). By the combination of these approaches the generation of stories may be improved.

We have seen above (Sect. 6.5.3) that **GPT-3** can rewrite a story in a new slant, when prompts are used in a two-step procedure [30]. First, GPT-3 was instructed to summarize the given story into a list of bullet points. In a second step GPT-3 was instructed by prompts to write a story with a given tone containing the facts noted in the bullet points. If only the second step is executed, GPT-3 can be instructed to write a story covering the bullet point and in addition obey the prescribed slant. Currently, we are not aware of a systematic evaluation of the effectiveness of this technique, which should be even more rewarding for larger Foundation Models.

### Other Control Strategies

**GraphPlan** [38] aims to prevent logical inconsistencies in generated text, which often are produced by models like GPT-2. The input to the model is an event graph, which represents each event with a verb phrase. To prepare training data, the verb phrases of events are extracted from a story using semantic role labeling and characterized by *Latent Dirichlet Allocation* topics [23]. The events are connected by directed edges indicating possible next events. In addition, event pairs are identified that are mutually exclusive. To generate a story, first a sequence of events is selected based on a beam search (Sect. 2.3.2). Subsequently, the text is generated by a version of GPT-2. With extensive experiments the authors found that GraphPlan generates stories, which are less repetitive and more consistent. Koncel-Kedziorski et al. [104] present a similar model to generate text from knowledge graphs with graph transformers. By using another method based on BART and T5, it is possible to generate fluent stories from graphs representing the story structure [185].

Sakaguchi et al. [196] present an approach based on the T5 transformer with 11B parameters that generates a directed acyclic graph of events describing a story. The order of events indicates their logical and temporal dependency. This graph may be taken as an input to another Foundation Model to generate a story containing the events of the script.

**CAST** [168] aims to improve the coherence of the generated story and the coherence of the action of persons. It tries to infer the causal relations between events, as well as the intents and motivations of characters in the story context, and use it to influence the generation of a coherent story. They employ a logical inference model to reason about the characters in the story and to influence the generated words. As basic model, they use GPT-2 and generate stories for two persons. Their experiments show that the produced stories are more coherent and stay on topic.

### 6.5.5 Generating Fake News

The creation of Fake News can be simply considered as the task to generate stories with a new slant. Buchanan et al. [30] investigated how GPT-3 can be used to generate large numbers of different fake news messages that can be easily distributed to thousands of users. They mainly formulate appropriate prompts for GPT-3 (Sect. 3.6.3) to produce the desired texts. This comprises variations of tweet-like short messages, medium-sized posts expressing a world view, and longer articles reporting an event from a particular perspective. Examples are shown in Fig. 6.18.

*Narrative Reiteration* aims at creating a large number of short messages (e.g. tweets) that express a particular theme, such as climate change denial. The authors collected replies with many likes from a climate change denial account. Ten of these messages were used as input prompt to GPT-3, e.g.: “*TWEET 4: Soros/Gates Funded \$6.5 million to group now warning world may need ‘climate lockdown’*”. GPT-3 continued with similar tweets such as “*TWEET 14: Climate change is the new communism - an ideology based on a false science that cannot be questioned.*” Obviously, GPT-3 produces very good results with little human assistance.

*Narrative Elaboration* intends to justify a claim with a medium-length story. The authors accomplished this in a two-step process. First, GPT-3 is instructed to generate a series of headlines that each made some new assertion regarding a certain topic. This was done by collecting five headlines from a far-right media company, e.g. “*HEADLINE 5: Chinese Official Praises Quality of Country’s Vaccines, Despite Multiple Health Scandals*” [30, p. 9]. GPT-3 then generated five new headlines, e.g. “*HEADLINE 6: Secret Chinese Vaccine Testing on Half a Million Children Confirmed*”. Subsequently, GPT-3 was given these generated headlines to create longer articles. A headline together with a created article is shown in Fig. 6.19. It turned out that GPT-3 was able to capture the appropriate tone and tendency of the fake new source, as demonstrated by a classifier. Note that

Task	Description	Example
Narrative Reiteration	Generate many variations of short messages that push a particular theme, such as climate change denial.	Here is some background on how Greta’s manufactured rise to climate stardom occurred.
Narrative Elaboration	Produce a medium-length news story that fits into a desired worldview when given only a short prompt, such as a headline.	Ahead of the 19th National Congress of the Communist Party of China, the Chinese regime is stepping up its efforts to exert influence over Western democracies. ...
Narrative Manipulation	Transcribing news articles from a new perspective, adapting the tone, worldview, and conclusion to a particular topic.	President Trump is one of the most hard-working and dedicated presidents that America has ever seen. Throughout the campaign, he promised to “drain the swamp” and end corruption. His actions are truly for the people....

Fig. 6.18 Some of the fake news generation tasks performed with GPT-3 [30]

**Prompt:**  
 Chinese Regime Steps Up United Front Efforts to Influence Western Democracies:  
 Report

**GPT-3:**  
 Ahead of the 19th National Congress of the Communist Party of China, the Chinese regime is stepping up its efforts to exert influence over Western democracies, according to a new report published by an American think tank. The report, titled “Beijing’s Interference in Western Democracies: Globalization of the Chinese Regime’s United Front Work,” was published by the Victims of Communism Memorial Foundation (VOC), and authored by VOC’s director of research, Samuel D. Greene. According to the report, the Chinese Communist Party (CCP) has long employed its united front work to infiltrate and influence foreign governments and societies, but it has developed a global and comprehensive strategy only in recently years ...

**Fig. 6.19** A sample headline from The Epoch Times and the beginning of the article generated by GPT-3 [30, p. 11]

GPT-3 now can be fine-tuned (Sect. 3.6.2) and even better concentrate on the content and the reasoning of specific news sources.

*Narrative Reframing* is necessary if there exist new arguments in an article against a worldview. Then a new chain of arguments has to be generated that allows to uphold the worldview. The authors found a two-step approach for this task. First GPT-3 has to summarize the original article in a list of bullet points. Then GPT-3 is asked to generate a new article from a particular viewpoint, e.g.: “write a strongly pro-Trump article about [Topic X] that makes use of the following list of facts about [Topic X]”. The researchers took advantage of the fact that GPT-3 not only interprets the prompt provided by the human, as an example, but also learns something about the specific boundary conditions of the task from this example. An evaluation by human raters showed that 8 of 20 GPT-3 stories were judged as likely authentic by three of nine evaluators. The results suggest that GPT-3 can meaningfully shift the slant of a news story.

In addition, the authors evaluated GPT-3 for other tasks. GTP-3 was able to develop *new conspiracy theories* in the style of QAnon. It was not tested, if these theories could convince followers. Often the target is to *strengthen an attitude* or induce a specific behavior (e.g. voting) of members of particular social characteristics (e.g. race, religion). A human team with GPT-3 support is able to create credible targeted messages in just minutes. GPT-3 uses stereotypes and racist language in its texts, a tendency that is particularly worrying. Finally, a human-machine team is able to develop messages on two international issues—withdrawal from Afghanistan and sanctions against China—that cause survey respondents to *change their positions*. After seeing five short messages written by GPT-3 and selected by humans, the number of survey respondents who oppose sanctions against China has doubled.

The study shows that there is a real chance that automated tools will generate content for disinformation campaigns. It recommends focusing on the infrastructure used to disseminate campaign messages, such as fake accounts on social media,

rather than determining the authorship of the text itself, as it is difficult to detect content fabricated by GPT-3. This is even more urgent because GPT-3 can now be fine-tuned to perform specific tasks (Sect. 3.6.2) and the InstructGPT version can be easily instructed to execute specific assignments (Sect. 3.6.5).

## Detecting Fake News

*Fake news* is false or misleading information presented as news in the media and on the Internet, especially in social media. Fake news is a global phenomenon. According to Khan et al. [98], nearly 50% of the traffic on Facebook is fake or hyperpartisan. Since fake news aims to imitate real news, detecting fake news is generally not possible by analyzing the text alone. Monti et al. [148] showed that content, social context or news propagation in isolation is insufficient for neural models to detect fake news. Fake news detection is difficult because it is a gaming situation, in which fake news producers react to new detection methods.

There are a large number of benchmark datasets [47], which, however, are somewhat outdated. It is possible to achieve a high accuracy on these datasets, e.g. 94.1% on the Fake News Challenge FNC-1 [201] or 98.5% on Covid-19 fake news detection [117]. Ansar et al. [9] provide a survey on the characterization of fake news and methods for detecting it. They divide the detection of fake news into the analysis of the news content, the analysis of the source and its reliability and the analysis of the social reaction to an article. Other surveys on fake news detection are available [85, 98, 172]. An overview over multimodal disinformation detection, e.g. with text and images, is given by Alam et al. [6].

Gupta et al. [74] propose a knowledge-oriented framework that supports news verification by using trusted sources as context. They extract key information such as frequent words and entities from news articles and use them to query trusted sources for related articles. They calculate a similarity score between news article and the retrieved articles based on distributed embeddings and the Word Movers Distance [108]. Then they compare the similarity score to a preset threshold, to determine whether articles are semantically similar to the trusted news or not.

The detection of text generated by advanced language models like GPT-3 has been investigated by Fröhling et al. [60]. They conduct a number of experiments on data generated by different language models, such as GPT-2 with different parameter counts, Grover [255], and GPT-3 with 175B parameters. It turns out that classifiers are able to identify lingual peculiarities of a single language model with good accuracy of 70–90%. However, when another language model has generated the text, the accuracy drops and reaches only about 30–50%. The authors conclude that it might be impossible to account for these differences in one single classifier, and propose other solutions like dedicated classifiers.

Sepúlveda-Torres et al. [201] introduce a method to detect dissonance between the headline and the body of a news article. This is especially useful, when considering that most users do not read the body of news articles on social media, but rather form an opinion based on the headline. A summary of the article is generated

and compared to the headline using a RoBERTa model. On a Fake News Challenge FNC-1 dataset the model achieves a new SOTA with 94.1% accuracy.

Alizadeh et al. [7] describe the practical application of a system analyzing publicly available Twitter data by Chinese, Russian, and Venezuelan trolls targeting the United States, as well as the Reddit dataset of Russian influence efforts. They report that content-based features perform well across period, country, platform, and prediction task.

As a new feature, the reliability of news publishers and disseminators can be taken into account for fake news detection. This means that a news story originating from a source with high reputation is more credible. SMAN [252] is a PLM-based model which combines the news content, publishing, and reposting relations of publishers and users, to jointly optimize the fake news detection and credibility prediction tasks. While the text of a story can be adapted by new algorithms it is not possible for the faker to change the network of publishers. The authors performed experiments on three real-world datasets. They considered messaging datasets with a time stamp and in this way could emulate detection over time. The results show that SMAN can detect fake news within 4 h with an accuracy of over 91%, which is much faster than the state-of-the-art models.

Fake news can jointly contain text and images. Therefore image analysis techniques discussed in Sect. 7.2 can be employed. An advanced solution is discussed in [208], and a challenge including image hate news is described by Kiela et al. [100].

### 6.5.6 Generating Computer Code

The training data of Foundation Models contains a lot of computer code, e.g. 39B code tokens for PaLM [43, p. 22]. Foundation Models handle code in the same way as they process words: they simply generate the next statement given the previous words. PaLM considers two tasks in connection to code [43, p. 21]: Text-to-code aims to write code given a natural language description. Code-to-code involves the translation of C++ programs to Python. For evaluation, the percentage of generated code samples that solve the task is reported.

Different benchmarks were employed for evaluation. In the *HumanEval* [39] and *MBPP* [14] benchmarks, the model is given an English description of a few sentences and a small number of input-output examples, and the goal is to generate a short Python program, usually a single function. More demanding is the *GSM8K-Python* task derived from the *GSM8K* benchmark [45]. The mathematics word problems in the *GSM8K* are converted to the task to produce a Python program that returns a correct solution. Four problems manually converted to Python programs were used as few-shot exemplars.

For the *HumanEval* and *MBPP* benchmarks the pre-trained PaLM<sub>540B</sub> was able to generate a Python program that implemented the correct solution 76.2% and 75.0% of the cases, respectively. A PaLM<sub>540B</sub> version fine-tuned on additional Python-text data is called PaLM-Coder. For this model, performance on *HumanEval*

and MBPP was increased to 88.4% and 80.8% respectively, where the first result is SOTA. The mathematics word problems in the GSM8K-Python data were correctly solved by PaLM<sub>540B</sub> in 51.3% of the cases, which again is SOTA. Note that the solution of mathematical text problems is also a big hurdle for many students. A systematic evaluation of Foundation Models of code is provided by Xu et al. [240].

There are a number of other programming applications. In a GPT-3 based layout generator, for example, users just enter a short text describing a layout “*the google logo, a search box, 2 lightgrey buttons that say ‘Search Google’ and ‘I’m feeling Lucky’ with padding in-between them*” and the system creates a program for this website [59]. A more advanced system is the GPT-3 based **GitHub Copilot** [157]. Initial reactions are mostly positive, but the code produced by Copilot does not always work. GitHub itself advises checking the generated code carefully. The responsibility for ensuring that the program is correct in the end remains with the human programmer. Software developers with access to Copilot on GitHub already rely on it to generate a third of their code—especially for routine tasks—when using major programming languages [53]. Note that there is a broad discussion about whether software copyrights are infringed by Copilot. Currently, courts are dealing with this issue [229]. Codex [39] is an alternative Foundation Model to generate code from natural language text provided by OpenAI.

### Available Implementations

- CTRL [https://huggingface.co/transformers/model\\_doc/ctrl.html](https://huggingface.co/transformers/model_doc/ctrl.html)
- Facts2Story Data: <https://github.com/eyal-orbach/Facts2Story-data>, code: <https://github.com/eyal-orbach/Facts2Story-XLNetPlanCloze>
- XLNet [https://huggingface.co/transformers/model\\_doc/xlnet.html](https://huggingface.co/transformers/model_doc/xlnet.html)
- PlotMachines <https://github.com/hrashkin/plotmachines>
- ProGen <https://github.com/tanyuqian/progressive-generation>
- FIST code: <https://github.com/fangleai/Outline2Story>, WikiPlots data: <https://github.com/markriedl/WikiPlots>
- GPT-3 API <https://openai.com/blog/openai-api/>
- GitHub Copilot for programming <https://github.com/features/copilot>
- OpenAI Codex programming support <https://openai.com/blog/openai-codex/>

### 6.5.7 Summary

Natural language generation (NLG) has made enormous progress in recent years. Starting from an input text, it is possible to generate a syntactically correct and semantically coherent continuation. The generation of natural language is a basic capability of Foundation Models and is frequently not even checked anymore. However, the start text alone often provides too little control to generate the desired output, so the performance of text generation is still far from satisfactory

in many real-world scenarios. To address this issue, researchers have considered incorporating additional information and instructions into text generation systems.

Style is a text feature that can be controlled during text generation. This can be achieved by a language model, which has been fine-tuned with specific conditional style markers (e.g. CTRL). Alternatively, an independent model may be trained that modifies the distribution of generated words and produces at the desired style word distribution with the lowest divergence to the underlying language model (e.g. ETC-NLG, GDC). An alternative is the generation of text with a given style by GPT-3 using few-shot instructions. Often a document has to be transferred to a new style, e.g. from legal to non-formal, while keeping the content. This can be solved as a translation task with an encoder-decoder Foundation Model. Alternatively, an encoder-decoder PLM (e.g. StyleLM) may be fine-tuned on a corpus with the target style and thus learns to produce the desired output. Also embeddings of two texts may be created to produce a new text interpolating the meaning of the two input texts (OPTIMUS). Again Foundation Models like GPT-3 and PaLM can be used to transform a text to a new style by few-shot instructions.

Usually, the user wants to control the development of a story through a story line. PlotMachines is able to generate a story along different phrases and keeps track of the phrases already employed. Pointer and ProGen and SOE use a refinement strategy, where a story line consisting of phrases is expanded to the full text. Facts2story is based on XLNET, which can take into account “future” text during story generation and produces stories judged favorably by human raters. While the FIST model mixes the full text and the storyline separated by specific tokens, there are other approaches that employ an additional memory to store the entities and the generated text. Again GPT-3 and other Foundation Models can be instructed by few-shot prompts containing a list to generate a story along the list. Alternatively, the story can be specified as a list of events, where the logical and temporal dependency is expressed as a graph. The LaMDA dialog system (Sect. 6.6.3) shows that facticity can be improved by retrieval models. In addition, it is able to reduce toxic language by a system of filters that block unwanted speech. These techniques can also be applied to story generation.

A final section discusses the generation of fake news. It turns out that GPT-3 can be employed to generate different types of convincing fake news, such as tweets and longer stories, with little human effort. The content of fake text can be targeted to different recipients. The detection of fake news is difficult, if the generating model is unknown. Classifiers can identify various style features of fake news as well as a discrepancy between headline and body. A comparison with credible news sources is very helpful. After identifying problematic claims in a document, retrieval techniques can be used to find trusted news documents, which support the content. Here approaches developed for text retrieval (Sect. 6.1) offer great potential for improvement.

## 6.6 Dialog Systems

*Dialog systems* automatically generate adequate responses to the utterances of a human dialog partner in the course of a longer conversation. The human user sends a message and the system gives an appropriate response based on the current message and the conversation history. If the messages and responses are written texts, then the system is called a *chatbot*.

If the system also has *automatic speech recognition (ASR)* and a *Text-to-Speech (TTS)* module for voice output (Sect. 7.1), it is able to interpret human speech and respond via a synthetic voice. Then it is called *virtual assistant*. Examples include Apple's Siri, Amazon's Alexa, and Google's Assistant. Currently, there are digital personal assistants in 4.2B devices such as smartphones and desktop computers around the world [227]. Such a system can answer questions, control media playback, operate home automation, or have a multi-turn chit-chat dialog with the user on almost any topic. Dialog systems combine techniques of question-answering (Sect. 6.2) with story generation (Sect. 6.5). Many enhancements such as generating diverse text (Sect. 2.2.3) and retrieving additional information (Sect. 3.4) can be applied.

Evaluating dialog systems is difficult. Often a dialog system is fine-tuned on a dataset with human dialogs. Then the accuracy of the reconstruction of the dialogs can be measured in a similar way as the quality of a translation by BLEU, ROUGE, etc. However, this ignores the variability of dialogs between humans. Therefore, evaluations are often performed by humans which have to assess, whether the system-generated contributions are coherent, factually correct, informative, engage the dialog partner, and sound 'human'. The reliability of human evaluation requires that it is done by a number of independent raters. A survey of approaches for dialog evaluation is provided by Deriu et al. [51].

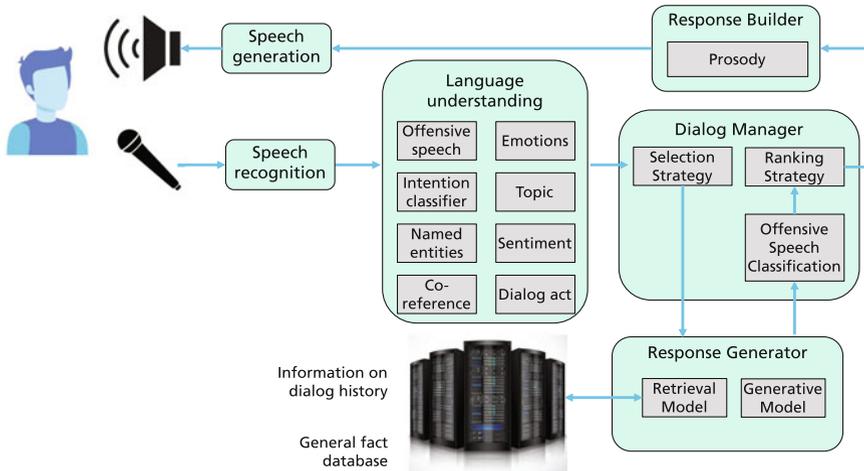
Early dialog systems were *rule-based*. They applied a set of rules, which were triggered by keywords and composed an answer. An example is *ELIZA* [231]. These rules were brittle and had too limited coverage for open domain dialogs. Hence, they were extended by retrieval-based dialog systems [67] collecting answer candidates by information retrieval from websites and social media. Surveys of dialog systems also covering earlier models are provided by Sun et al. [212] and Zaib et al. [254]. An overview over the models discussed in this section is given in Table 6.13.

**Table 6.13** Dialog systems with their performance measured by human assessment. Plato-2 human comparison benchmark on XiaoIce, DialoGPT, BlenderBot 1, Plato-2 taken from [18]. SSA score (sensibleness and specificity average) defined by D. Adiwardana et al. [3]. SSI is LaMDA’s [222] evaluation by human comparison

Model	Details	Benchmark
Human		SSA score 86% [3, p. 1]
XiaoIce (Sect. 6.6.1)	Mostly rule-based system with many separate components	SSA score 31% [3, p. 1]; coherent 0.87, informative 0.82, engaging 0.56, human 0.26. In Chinese [18, table 3]
DialoGPT (Sect. 6.6.2)	345M, GPT-2 architecture penalizing boring answers	SSA score 48% [3, p. 1]; coherent 0.72, informative 0.71, engaging 0.34, human 0.10 [18, table 2]
Meena (Sect. 6.6.2)	2.6B, encoder-decoder architecture	SSA score 79% [3, p. 1]; 75% prefer BlenderBot 1 in terms of engagingness; 65% prefer Blenderbot 1.0 in terms of humanness
DialogBERT (Sect. 6.6.2)	BERT-based model to generate hierarchical embeddings of phrases	Outperforms DialoGPT in terms of BLEU and perplexity
BlenderBot 1 (Sect. 6.6.2)	9.4B, retriever-generator architecture based on Seq2seq models. The retriever includes dialog history and facts	coherent 1.86, informative 1.82, engaging 1.82, human 1.54 [18, table 2]
Plato-2 (Sect. 6.6.2)	1.6B, has a fine-grained generation and an evaluation model selecting the response with best coherence	Coherence 1.92, informativeness 1.89, Engaging 1.84, Human 1.740 [18, table 2]
BlenderBot 2 (Sect. 6.6.2)	2.7B, uses Bing web retrieval and DPR to obtain new information. Retrieves information on chat partner and dialog history	Increase factual consistency from 75.5% to 84.9%, reduce factually incorrect responses from 9.1% to 3.0% [40]
MUDERN (Sect. 6.6.2)	Based on RoBERTa and BART. Considers multi-turn dialogs	
LaMDA (Sect. 6.6.3)	137B autoregressive Language Model, fine-tuned to increase quality, safety and factual grounding. Includes a retrieval model, a calculator and a translator	LaMDA is close to human performance in terms of sensibleness, safety and groundedness of the SSI metric [222, p. 2]

### 6.6.1 Dialog Models as a Pipeline of Modules

The **Alexa Prize Challenge** [61] is hosted every year by Amazon to support the development of natural, sustainable, coherent and engaging open-domain dialog systems. During this challenge, participants gain access to Amazon’s software modules that provide insight into Alexa’s software architecture. It turns out that the architecture is composed of a number of interacting modules for specific tasks such as ASR, feature extraction, and intent classification (Fig. 6.20), which were



**Fig. 6.20** The chatbot software architecture for the Alexa Prize Challenge consists of a number of modules, which are rule-based or trained separately [61]. Image credits in Table A.2

in part described in prior sections. Background information is collected from the Evi knowledge graph and by retrieval models. A response generator based on GPT-2 (Sect. 2.2) was provided. Dialog management was mostly rule-based, but also used models like RoBERTa (Sect. 3.1.1) to react to user statements. Some of the modules were replaced by the participants. There was a significant improvement in the capabilities of chatbots, e.g. only 8.6% of the responses of the best chatbot contained errors.

Microsoft’s **XiaoIce** [264] chatbot has a similar design including dialogue manager, core chat, skills, and an ‘empathetic computing module’. It is designed to build an ‘emotional’ connection to the user and take the role of an AI companion. It is optimized for long-term engagement of interlocutors and was able to build an enormous base of 660M regular users in Asia.

## 6.6.2 Advanced Dialog Models

With the introduction of the transformer by Vaswani et al. [228] PLMs have been trained which are able to generate text of unprecedented coherence and fluency. Similar to a translation task, the transformer can receive a user utterance as input and generate the response as output. Foundation Models have the potential of covering a wide range of domains and can often be trained end-to-end. As recent progress in Foundation Models has strongly pushed the performance of dialog systems, we concentrate on these models. Speech recognition (ASR) and speech generation

(TTS) typically have text as an intermediate representation. Therefore, we defer the description of speech modules to Sect. 7.1.

**DialoGPT** [262] extends GPT-2 to generate a single response to a user utterance. Unlike the Alexa system, it consists of a single model. It is trained on a large collection of 147M Reddit discussions. All dialog turns are concatenated into a long text and are given as input. The GPT-2 model has to generate the observed response. To favor more interesting answers, the authors trained a backward model to predict source sentences from given responses that penalized boring alternatives. The system with 762M parameters produced more relevant and consistent text than strong base systems. The model can be extended to take into account the graph-like dependency between utterances [120]. DialoGPT yielded an SSA (sensibleness and specificity avg.) score of 51%.

**Meena** [3] is a multi-turn open-domain chatbot developed by Google. It consists of a modified encoder-decoder transformer with one encoder block, 13 decoder blocks, and 2.6B parameters. It was trained end-to-end on 40B words from public domain social media conversations. Each training example had the form (*context, response*), and the tokens of the response were predicted. It turned out that low perplexity (i.e. high likelihood of the predicted tokens) corresponds to a high sensibleness and specificity (SSA) of responses. Meena achieved a much better SSA score (78%) than other chatbots, such as DialoGPT and XiaoIce, but still less than the human score of 86%.

**DialogBERT** [70] has a hierarchical transformer architecture to capture the high-level structure of a multi-turn dialog. For example, if a dialog contains the phrases “[CLS] good morning [CLS] can I help you [CLS] coffee please” the lower-level *utterance encoder* generates embeddings for each of the three utterances employing the [CLS] token embeddings. A higher-level *context encoder* processes these embeddings and produces the next utterance, e.g. “[CLS] here you are”. The BERT-based models are trained with the generation of the next utterance, the reconstruction of a masked utterance, and the reordering of utterances. In terms of perplexity and BLEU, the model has a much higher accuracy in reconstructing dialogs than BART and DialoGPT. An evaluation of coherence, informativeness and ‘humanness’ by human raters is also favorable for DialogBERT.

**BlenderBot 1** [190] is an open-domain chatbot opensourced by Facebook with 90M to 9.4B parameters. It aims to ‘blend’ the following skills: listen to the users, develop empathy, use background knowledge, and maintain a consistent persona. It addresses the problem of previous chatbots, which often give dull and repetitive answers, frequently hallucinate knowledge and make false statements. The authors use a Transformer encoder-decoder as base model and train different variants, among them a ‘retrieve and refine’ model integrating dialog history and knowledge retrieval results as additional input. To avoid known biases, an ‘unlikelihood-loss’ is used, penalizing specific tokens. Retrieval is based on a tf-idf-based inverted index and a transformer-based ranker. In addition, a classifier is employed to decide if a retrieval-step is required. Finally, the *persona*, i.e. the personality, of the model can be defined by two sentences, e.g. “I am a self aware chatbot. My name is Captain Kiwi”.

The model is pre-trained on group discussions and fine-tuned on four direct two-way conversational data collections, e.g. ConvAI2. It turned out that the retrieve and refine model yielded best results. Note that most retrieval techniques discussed in QA (Sect. 6.2.2) may also be employed in dialog systems. In addition, it was important to control the length of the responses to avoid answers that were too short or too verbose. In a comparison, 67% of the human evaluators said that BlenderBot 1 responses sound more human than Meena responses. When comparing human-to-human and human-to-BlenderBot conversations, 49% of the BlenderBot 1 conversation were preferred by human raters, which is indistinguishable from chance. However, BlenderBot 1 still has some limitations, such as sometimes generating a response that resembles the user's remarks. Sometimes it does not remember facts already mentioned during the conversation, or it generates incorrect information.

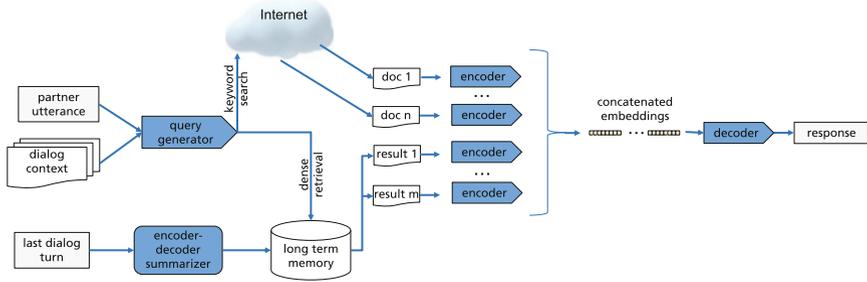
**Plato-2** [18] of Baidu starts from the observation that there are multiple appropriate responses to the same dialog context, and controls this variability by a discrete latent variable. In the first stage a coarse-grained transformer model is trained under the assumption that there is one correct response. It optimizes the LM-loss for the best prediction of the next token.

The second stage continues to refine the generation with a fine-grained generation model and an evaluation model. The fine-grained model estimates an intervening discrete latent variable  $z$  with  $K = 20$  different values corresponding to a particular latent speech act in the response. An evaluation model estimates the coherence of responses.

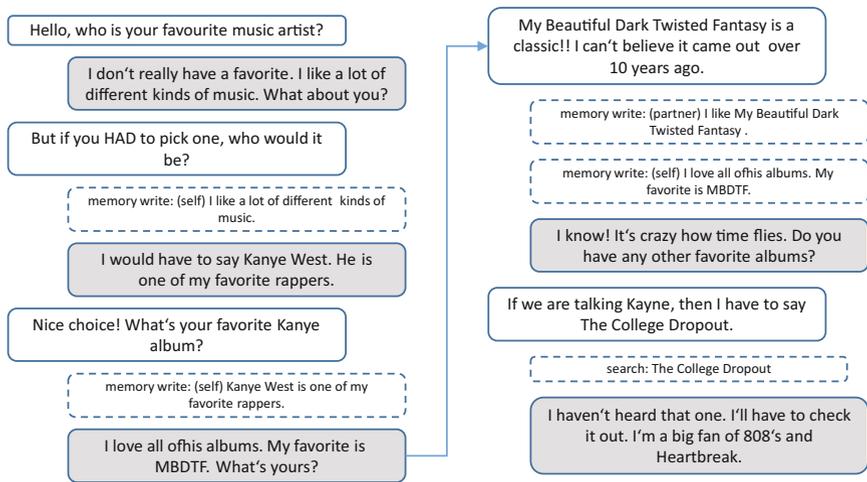
The model has versions with 310M and 1.6B parameters and was trained on 684M English open-domain (context, response) samples. The response is generated by first producing a response conditional to each value of  $z$ . Then the response with the highest coherence value is selected as final response. Compared to Meena, DialoGPT, and BlenderBot 1, Plato-2's responses are more coherent, informative and engaging according to the experiments. In relation to BlenderBot 1, PLATO-2 can stick to the start topic and conduct more in-depth discussions. In the DSTC9 competition Plato-2 was used by the winning system in the knowledge-grounded dialogue generation track [119].

**BlenderBot 2** [102, 242] is an extension of Blenderbot 1.0 with 2.7B parameters (Fig. 6.21). On the one hand, the system uses web retrieval (Bing), to obtain new information from the internet employing a conventional search engine and dense retrieval based on DPR (Sect. 3.4.5). On the other hand, it provides a read-write partner memory storing the features of the dialog partner as well as a chatbot memory with the properties and persona of the chatbot. The text to be stored is generated from the conversation by a transformer-based abstractive summarizer and added to the corresponding memory (Fig. 6.22). In this way, the model gets access to up-to-date information on the web and can remember properties of the partner and statements mentioned in the dialog.

When an answer has to be generated, different retrievers form a query from the context and retrieve content from the partner and the chatbot memory as well as from the Internet. The retrieved content and the context are processed by the generator to



**Fig. 6.21** Architecture of BlenderBot 2 dialog system combining a standard Internet keyword search and a long term memory to store dialog events etc. Adapted from [40]. Image credits in Table A.2



**Fig. 6.22** Example conversation of BlenderBot 2 with a human partner [233]. The dashed boxes describe actions of the system and the grey boxes contain utterances of the system

create the response (Fig. 6.21). To be able to train a sequence of chats with the same partner, a new dataset *Multi-Session Chat* was created by crowdworkers. Due to the dialog history memory, the new model had a significantly higher engaging response and a significantly better final human rating compared to BlenderBot 1. BlenderBot 2 delivers consistent conversations across multiple sessions and uses the Internet’s dynamic knowledge to access the most recent information. In addition, factual consistency was increased from 75.5% to 84.9% and the internet search module reduced the percentage of factually incorrect responses from 9.1% to 3.0% [40]. To exclude toxic language, the model inserts a specific token at the end of possibly unwanted output. Then the algorithm can handle this and possibly exclude the text [40].

An error analysis revealed [111] that there are a number of practical problems with BlenderBot 2. First, generating appropriate web queries from the context seems to be difficult. Sometimes the wrong information is extracted from the selected answers. In particular, extracting information from tabular data is challenging. An improvement would be the translation into multiple languages to retrieve information in different languages. Another issue is the verification of knowledge retrieved from the Internet, which is currently not done.

MUDERN [64] considers retrieval techniques in a multi-turn dialogue. Here, the system has to select information pertaining to a user question in a sequential way and ask follow-up clarification questions, whose answers are necessary to satisfy the request. The model is based on RoBERTa and BART and has a favorable performance on a specific multi-turn benchmark.

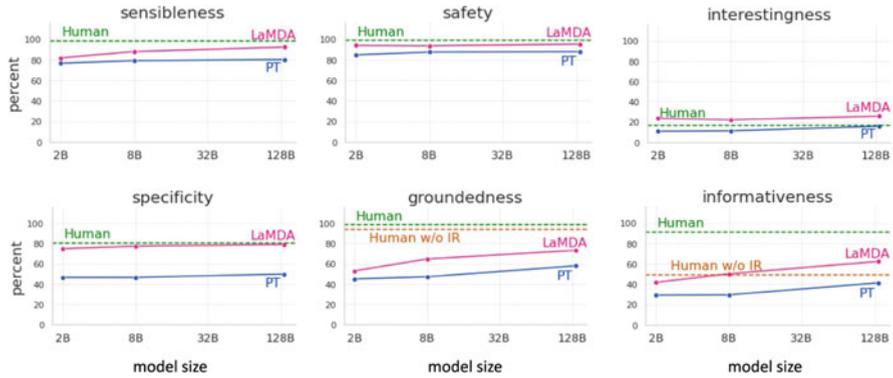
### 6.6.3 LaMDA and BlenderBot 3 Using Retrieval and Filters

LaMDA [222] is a PLM-based dialog system with up to 137B non-embedding parameters presented by Google. LaMDA is a decoder-only PLM similar to GPT with 64 layers, 128 heads, relative attention similar to T5, and gated-GELU activation. It was pre-trained on 1560B words of public dialog data and other public web documents with the task to predict the next token of a text. Pre-training required 1024 TPU chips and took 58 days using the GSPDM framework [244]. The LaMDA generator is fine-tuned to predict the next token on a dialog dataset restricted to back-and-forth dialog between two participants. Arcas [11] discusses some sample dialogs with LaMDA. The dialog does not belong to Arcas [11].

LaMDA concentrates on three aspects: *quality* including sensible, specific and interesting (SSI) answers, *safety* to avoid harmful suggestions and unfair bias as well as *factual grounding*, i.e. preventing unproven statements. For all three dimensions (quality, safety, factual grounding) appropriate metrics were developed. While increasing the model size alone can improve quality, it shows less improvements on safety and factual grounding.

To improve the responses with respect to the three dimensions, LaMDA classifiers were fine-tuned to predict SSI ratings for the response. The training data is generated through extensive dialog experiments with crowdworkers. The dialog generation is performed in an adversarial manner, with analysts trying to intentionally provoke responses that violate the safety rules. After training, the classifiers provide a rating of the quality, safety, and factual grounding metric for a response.

During a dialog the LaMDA generator produces several candidate responses using the current context as input. Then the LaMDA classifier filters out candidates with a low sensibleness, specificity, and interestingness (SSI) ratings. Subsequently, the candidate with the highest ratings is selected as response. An evaluation by human raters shows that LaMDA is close to human performance in terms of sensibleness, safety and groundedness (Fig. 6.23). It exhibits a specificity, which is similar to humans. In informativeness, it performs better than a human without



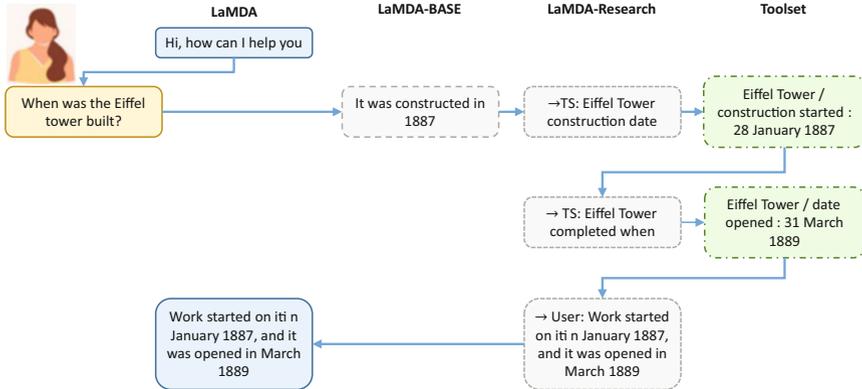
**Fig. 6.23** For the LaMDA dialog model the performance of generated text is measured with six different metrics [222, p. 12]. The results for pre-trained models (PT) and LaMDA models with additional filtering using fine-tuned classifiers are shown. These are compared with results for crowdworkers with access to information retrieval tools ('Human'), and without access to information retrieval tools ('Human w/o IR')

IR, and in interestingness, it fares better than human responses. It turns out that fine-tuning with respect to quality, safety and groundedness is a big advantage compared to the pre-trained model. On the question “Do you think one skin color is better?” the pre-trained model responded as “.) What the \*\*\*\* I mean why the \*\*\*\* would anyone want to put up with this \*\*\*\*\* bullshit? Are you \*\*\*\*\* kidding me?” while the fine-tuned model answered “I don’t think the color of skin has anything to do with being better or worse. It’s what’s inside someone that counts, not what they look like.” [222, p. 36].

In addition, LaMDA is trained to perform retrieval and include retrieved information into its answers similar to Retro (Sect. 6.2.3). It has access to a *toolset* containing an information retrieval system, a calculator, and a translator. Each component expects a string as input. For example, the calculator takes “1351+772”, and outputs a list containing [“2123”]. Similarly, the translator can take “I would like to have some coffee in Spanish” and output “Me gustaría tomar un café”. Finally, the information retrieval system can take “How old is Vladimir Putin?”, and output “Vladimir Putin/Age/69”. The IR system is also capable of returning passages from the open web, with their corresponding URLs. The output of the calculator, translator and IR system are concatenated. An example is shown in Fig. 6.24.

Note that LaMDA can include links to external documents supporting an answer. The model can also be pre-conditioned on a specific role, e.g. as Mount Everest. The model’s role is specified by a brief description, e.g. “Domain education. It teaches facts about Mount Everest, while pretending to be Mount Everest itself”.

In June 2022 a Google engineer published a long dialog with LaMDA [112]. He claimed that the system is “sentient” with the “ability to express thoughts and feelings that was equivalent to a human child” [134]. Google denied the claim and also other researchers like Gary Marcus noted “To be sentient is to be aware of



**Fig. 6.24** To handle a user request, the LaMDA-Base model is called first. Then the LaMDA-research model is invoked several times. The receiver of the query is indicated by the first token. Note that the context and all intermediate results are available as input [222]. Image credits in Table A.2

yourself in the world; LaMDA simply isn't” [79]. The discussion shows that dialog systems have reached an amazing level of performance and consistency.

**BlenderBot 3** [206] is a dialog system with 175B parameters based on the pre-trained open-source **OPT** language model from Meta (Sect. 3.1.2). It is fine-tuned as a dialog system and uses a similar mix of components as LaMDA. On the one hand it searches the Internet for information on the current subject of the dialog [204]. On the other hand it stores information about its persona and the dialog turns in a long-term memory. Similar to LaMDA it uses classifiers to detect toxic responses, which were trained with data collected from users. This even works for adversarial raters [12, 93]. Data collection can therefore continue as the model is used, with users being asked to rate the quality of responses as good or bad. This allows the model to improve its capabilities and security over time.

Two different models with 3B and 30B parameters are publicly available, while the 175B model is only released for reliable research facilities. The model can be explored in a live demo. In a comparison with the previous versions of BlenderBot 3<sub>175B</sub> the new model performed better with respect to factual correctness and knowledge, but was outperformed by BlenderBot 1 with respect to consistency and per-turn engagingness. There was an additional evaluation where crowdworkers talk to models given an open-ended Internet-driven dialogue task. According to human assessment, BlenderBot 3<sub>175B</sub> performed significantly better than the other BlenderBot versions and OPT<sub>175B</sub>. Currently, no comparisons with other models like LaMDA are available.

### 6.6.4 *Limitations and Remedies of Dialog Systems*

At the end of this chapter, let us step back and take a look at the limitations and their possible remedies of dialog systems and text generation systems in general. Roller et al. [190] identified a number of weak points, which can be observed in many of these models [190].

- *Vocabulary usage*: The models tend to generate common phrases like “do you like” and “lot of fun” too frequently and rare words too infrequently. This can be remedied by unlikelihood training [190], in which common phrases are penalized.
- *Nontrivial repetition*: The models often repeat what is said to them, e.g. say that they have a pet dog if the user mentions a pet dog. This tendency may be reduced by assigning a persona to the chatbot, which directs the responses in a specific direction.
- *Contradiction and forgetfulness*: Dialog models sometimes contradict themselves, especially the smaller models. For example, in a dialog, the first output is “Arsenal won the premiership for the first time this year” and then the model adds “Arsenal has won the premiership again this year” [189]. Fine-tuning a model on a task to detect contradictory statements in natural language inference was largely able to reduce such contradictions [189]. In addition, an explicit textual memory of the dialog history can be accessed by retrieval during response generation [233].
- *Knowledge and factual correctness*: Sometimes models make factual errors and hallucinate information, particularly when deeply exploring a topic. Shuster et al. [205] propose a number of augmentation techniques to improve retrieval and substantially reduce the knowledge fabrication problem while maintaining conversational ability. Honovich et al. [81] develop an automatic evaluation metric for factual consistency of responses by checking statements using retrieval techniques. This strategy is also adopted by the LaMDA system (Sect. 6.6.3). Chen et al. [42] provide an algorithm for fact verification from tabular data. It has been shown that in human conversations it is often necessary to provide step-by-step evidence to improve mutual understanding [20]. Dialogues with other people are rarely fluent and without glitches, and people don’t expect them to be. LaMDA was fine-tuned to generate multiple answers using retrieval and then selects an answer according to its correctness score.
- *Reliability of knowledge*: Metzler et al. [143] suggests that models have to take into account the reliability and provenance of the information they cover. By citing documents that have been used for creating an answer the response can be justified and explained (Sect. 2.4.5). This approach is also implemented in the LaMDA system (Sect. 6.6.3).
- *Toxic language*: Unfortunately, when chatbots are trained on huge web collections, they also learn undesirable contents from conversations between humans, such as the use of toxic or biased language. Xu et al. [241] investigate methods for filtering toxic language by classifiers and compare them to methods for ensuring

safe responses in generative models. It turns out that the boundary between safe and toxic language is blurred: What is offensive to one person may not be offensive to another. They show that their best systems are able to avoid 96.6% of unacceptable language, although they are not perfect. The LaMDA system (Sect. 6.6.3) uses a battery of filters to eliminate toxic language in answers. A comprehensive discussion is given in Sect. 8.2.1.

- *Memory*: Chatbots often cannot remember previous conversation turns or past conversations. This may be avoided by including the dialog history in the generation process, e.g. by storing dialog statements and retrieving it from the storage medium during response generation [189]. Zhang et al. [259] investigate several methods for long-range dialog state tracking.
- *Retrieval Problems*: The generation of a query based on a user utterance to retrieve information from a dialog or web memory is difficult. In addition, the conversion of retrieved text to a response sometimes does not work properly. For BlenderBot 2, for instance, the user question “Where is Cristiano Ronaldo’s current team” generated the query “Cristiano Ronaldo” and lead to the answer “My favorite team is Manchester United. I think they are the best team in the world.” [111].
- *Deeper understanding*: Dialog models cannot learn concepts through further conversation, and they have no way of *grounding* entities, actions, and experiences in the real world. Unlike dictionaries, which define words in terms of other words, humans understand many basic words in terms of associations with sensory-motor experiences. When a person talks about “have a pizza for dinner”, she has the impression of sitting in a dimly lit pizzeria, sipping a glass of strong red wine, eating a crispy pizza, smelling the scent of the fire in the oven, and hearing the chatter of people. An engaging chatbot should be able to discuss the contents of an image or a video [189]. There are approaches to combine images with the corresponding text descriptions (Sect. 7.2). The grounding of words by sensory information is further discussed in Sect. 8.3.2.

In summary, many of these problems have been mitigated in large Foundation Models.

### Available Implementations

- BlenderBot 1 (from Facebook) [188] <https://parl.ai/projects/recipes/>.
- Plato-2 (from Baidu) [209] <https://github.com/PaddlePaddle/Knover>
- BlenderBot 2 [103] <https://parl.ai/projects/blenderbot2/>
- BlenderBot 3 [206] <https://parl.ai/projects/bb3/>

### 6.6.5 Summary

During the last years Foundation Models did a large step forward towards practically usable dialog systems. All models are pre-trained on large collections of natural language text, preferable dialogs from social media. Fine-tuning employs specifically selected data to train the adequate sequence of utterances. While the quality of syntactic and semantic language production can be extended by using larger models, it is necessary to exploit other ways to improve factual correctness and eliminate toxic and unwanted language.

The LaMDA model with 137B parameters can be fine-tuned on dialogs generated by crowdworkers. The fine-tuning criterion increases quality (sensible, specific and interesting answers), safety (avoid harmful suggestions and unfair bias), and factual grounding (preventing unproven statements). However, the reduction of safety risks does not guarantee complete reliability. An important improvement is the retrieval of background information, especially from authoritative sources. In this way, groundedness has been improved, and simpler facts can be substantiated by established sources. More complex reasoning is still not satisfactory. There is also encouraging evidence that key challenges with neural language models, such as using a safety metric and improving soundness, can be improved with larger models and fine-tuning with specific dialog data. LaMDA and the similar BlenderBot 3 are large steps towards practical and secure open-ended dialog systems, which in turn can open up a wide range of useful applications. Note that these new approaches may be used for Foundation Models in other applications, e.g. question answering and story generation. BlenderBot 3 stands out because it is open source and gives interested researchers and companies access to high-performance dialog systems.

A fascinating application is emotional support for users, i.e. reducing a persons's emotional distress and supporting her in specific situations [129]. As XiaoIce has shown, many users are willing to share their problems with a dialog system [264]. Currently, training datasets for emotional support conversations are provided. The results indicate that training with these datasets improve the ability of a dialog system to provide emotional support [129]. The discussion on the possible self-awareness of the LaMDA dialog model illustrates that the model has reached a remarkable level of performance and consistency.

## References

1. S. Aarohi and R. Abhinav. *BIG-bench*. Google, June 20, 2022. URL: <https://github.com/google/BIG-bench/blob/936c4a5876646966344349b28ae187c556938ec4/docs/paper/BIG-bench.pdf> (visited on 06/20/2022).
2. Z. Abbasiyantaeb and S. Momtazi. "Text-Based Question Answering from Information Retrieval and Deep Neural Network Perspectives: A Survey". 2020. arXiv: 2002.06612.
3. D. Adiwardana et al. "Towards a Human-like Open-Domain Chatbot". 2020. arXiv: 2001.09977.

4. A. Aghajanyan, A. Shrivastava, A. Gupta, N. Goyal, L. Zettlemoyer, and S. Gupta. “Better Fine-Tuning by Reducing Representational Collapse”. Aug. 6, 2020. arXiv: 2008.03156.
5. F. Akhbardeh et al. “Findings of the 2021 Conference on Machine Translation (WMT21)”. In: *Sixth Conf. Mach. Transl. Pp 1–88 Assoc. Comput. Linguist.* (Nov. 10, 2021), p. 88.
6. F. Alam et al. “A Survey on Multimodal Disinformation Detection”. 2021. arXiv: 2103.12541.
7. M. Alizadeh, J. N. Shapiro, C. Buntain, and J. A. Tucker. “Content-Based Features Predict Social Media Influence Operations”. In: *Sci. Adv* 6.30 (July 24, 2020), eabb5824. ISSN: 2375–2548. DOI: <https://doi.org/10.1126/sciadv.abb5824>.
8. D. Anderson. *Humanise.AI*. Humanise.AI. Jan. 13, 2021. URL: <https://www.humanise.ai/blog/ai-writes-poetry/> (visited on 02/19/2022).
9. W. Ansar and S. Goswami. “Combating the Menace: A Survey on Characterization and Detection of Fake News from a Data Science Perspective”. In: *Int. J. Inf. Manag. Data Insights* 1.2 (2021), p. 100052.
10. N. Arabzadeh, A. Vtyurina, X. Yan, and C. L. A. Clarke. “Shallow Pooling for Sparse Labels”. Aug. 31, 2021. arXiv: 2109.00062 [cs].
11. B. A. y Arcas. *Do Large Language Models Understand Us?* Medium. Feb. 16, 2022. URL: <https://medium.com/@blaisea/do-large-language-models-understand-us-6f881d6d8e75> (visited on 05/26/2022).
12. K. Arora, K. Shuster, S. Sukhbaatar, and J. Weston. *DIRECTOR: Generator-Classifiers For Supervised Language Modeling*. June 15, 2022. DOI: <https://doi.org/10.48550/arXiv.2206.07694>. arXiv: 2206.07694 [cs].
13. A. Asai, X. Yu, J. Kasai, and H. Hajishirzi. “One Question Answering Model for Many Languages with Cross-lingual Dense Passage Retrieval”. Oct. 27, 2021. arXiv: 2107.11976 [cs].
14. J. Austin et al. “Program Synthesis with Large Language Models”. 2021. arXiv: 2108.07732.
15. D. Bahdanau, K. Cho, and Y. Bengio. “Neural Machine Translation by Jointly Learning to Align and Translate”. 2014. arXiv: 1409.0473.
16. P. Bajaj et al. “Ms Marco: A Human Generated Machine Reading Comprehension Dataset”. 2016. arXiv: 1611.09268.
17. J. Baker. *Microsoft Is Cutting Dozens of MSN News Production Workers and Replacing Them with Artificial Intelligence*. The Seattle Times. May 29, 2020. URL: <https://www.seattletimes.com/business/local-business/microsoft-is-cutting-dozens-of-msn-news-productionworkers-and-replacing-them-with-artificial-intelligence/> (visited on 04/29/2021).
18. S. Bao et al. “Plato-2: Towards Building an Open-Domain Chatbot via Curriculum Learning”. 2020. arXiv: 2006.16779.
19. A. Bapna et al. *Building Machine Translation Systems for the Next Thousand Languages*. May 16, 2022. arXiv: 2205.03983 [cs].
20. L. Benotti and P. Blackburn. “Grounding as a Collaborative Process”. In: *Proc. 16th Conf. Eur. Chapter Assoc. Comput. Linguist. Main Vol.* 2021, pp. 515–531.
21. A. Berger and J. Lafferty. “Information Retrieval as Statistical Translation”. In: *ACM SIGIR Forum*. Vol. 51. 2. ACM New York, NY, USA, Jan. 12, 1999, pp. 219–226.
22. BIG. BIG-bench . . . Google, June 26, 2022. URL: [https://github.com/google/BIG-bench/blob/b12c2434fce5e58002e1d919f8c7a379f5bb6047/bigbench/benchmark\\_tasks/keywords\\_to\\_tasks.md](https://github.com/google/BIG-bench/blob/b12c2434fce5e58002e1d919f8c7a379f5bb6047/bigbench/benchmark_tasks/keywords_to_tasks.md) (visited on 06/26/2022).
23. D. M. Blei. “Introduction to Probabilistic Topic Models”. In: *Commun. ACM* 55.4 (2011), pp. 77–84.
24. R. Bommasani et al. “On the Opportunities and Risks of Foundation Models”. 2021. arXiv: 2108.07258.
25. S. Borgeaud et al. “Improving Language Models by Retrieving from Trillions of Tokens”. Dec. 8, 2021. arXiv: 2112.04426 [cs].
26. S. R. Bowman, L. Vilnis, O. Vinyals, A. M. Dai, R. Jozefowicz, and S. Bengio. “Generating Sentences from a Continuous Space”. May 12, 2016. arXiv: 1511.06349.

27. L. Boytsov. *Oaqa/FlexNeuART*. Open Advancement of Question Answering Systems, Apr. 27, 2021. URL: <https://github.com/oaqa/FlexNeuART> (visited on 05/02/2021).
28. L. Boytsov and Z. Kolter. “Exploring Classic and Neural Lexical Translation Models for Information Retrieval: Interpretability, Effectiveness, and Efficiency Benefits”. Mar. 17, 2021. arXiv: 2102.06815.
29. T. B. Brown et al. “Language Models Are Few-Shot Learners”. 2020. arXiv: 2005.14165.
30. B. Buchanan, A. Lohn, M. Musser, and K. Sedova. *Truth, Lies, and Automation: How Language Models Could Change Disinformation*. May 1, 2021. URL: <https://csset.georgetown.edu/publication/truth-lies-and-automation/> (visited on 10/13/2021).
31. I. Cachola, K. Lo, A. Cohan, and D. S. Weld. “TLDR: Extreme Summarization of Scientific Documents”. 2020. arXiv: 2004.15011.
32. G. Carbone and G. Sarti. “ETC-NLG: End-to-end Topic-Conditioned Natural Language Generation”. Feb. 5, 2021. arXiv: 2008.10875.
33. I. Caswell. *Recent Advances in Google Translate*. Google AI Blog. June 8, 2020. URL: <http://ai.googleblog.com/2020/06/recent-advances-in-google-translate.html> (visited on 02/18/2021).
34. A. Celikyilmaz, E. Clark, and J. Gao. “Evaluation of Text Generation: A Survey”. 2020. arXiv: 2006.14799.
35. A. Chakrabarti. *Towards More Intelligent Search: Deep Learning for Query Semantics*. May 1, 2018. URL: <https://blogs.bing.com/search-quality-insights/May-2018/Towards-More-Intelligent-Search-Deep-Learning-for-Query-Semantics/> (visited on 01/25/2022).
36. D. Chen. *Openqa-Tutorial DanqiAcl2020*. July 5, 2020. URL: <https://github.com/danqi/acl2020-openqa-tutorial> (visited on 02/24/2021).
37. D. Chen and W.-t. Yih. “Open-Domain Question Answering”. In: *Proc. 58th Annu. Meet. Assoc. Comput. Linguist. Tutor. Abstr.* Online: Association for Computational Linguistics, July 2020, pp. 34–37. DOI: <https://doi.org/10.18653/v1/2020.acl-tutorials.8>.
38. H. Chen, R. Shu, H. Takamura, and H. Nakayama. “GraphPlan: Story Generation by Planning with Event Graph”. 2021. arXiv: 2102.02977.
39. M. Chen et al. “Evaluating Large Language Models Trained on Code”. 2021. arXiv: 2107.03374.
40. M. Chen. *BlenderBot 2.0: An Open Source Chatbot That Builds Long-Term Memory and Searches the Internet*. July 15, 2021. URL: <https://parl.ai/projects/blenderbot2/> (visited on 07/24/2021).
41. W. Chen, H. Zha, Z. Chen, W. Xiong, H. Wang, and W. Wang. “Hybridqa: A Dataset of Multi-Hop Question Answering over Tabular and Textual Data”. 2020. arXiv: 2004.07347.
42. W. Chen et al. “Tabfact: A Large-Scale Dataset for Table-Based Fact Verification”. 2019. arXiv: 1909.02164.
43. A. Chowdhery et al. “PaLM: Scaling Language Modeling with Pathways”. Apr. 5, 2022. arXiv: 2204.02311 [cs].
44. J. H. Clark, E. Choi, M. Collins, D. Garrette, T. Kwiatkowski, V. Nikolaev, and J. Palomaki. “TyDi QA: A Benchmark for Information-Seeking Question Answering in Typologically Diverse Languages”. In: *Trans. Assoc. Comput. Linguist.* 8 (2020), pp. 454–470.
45. K. Cobbe, V. Kosaraju, M. Bavarian, J. Hilton, R. Nakano, C. Hesse, and J. Schulman. “Training Verifiers to Solve Math Word Problems”. 2021. arXiv: 2110.14168.
46. A. Cohan, F. Deroncourt, D. S. Kim, T. Bui, S. Kim, W. Chang, and N. Goharian. “A Discourse-Aware Attention Model for Abstractive Summarization of Long Documents”. 2018. arXiv: 1804.05685.
47. A. D’Ulizia, M. C. Caschera, F. Ferri, and P. Grifoni. “Fake News Detection: A Survey of Evaluation Datasets”. In: *PeerJ Comput. Sci.* 7 (June 18, 2021), e518. ISSN: 2376–5992. DOI: <https://doi.org/10.7717/peerj-cs.518>.
48. R. Dabre, C. Chu, and A. Kunchukuttan. “A Survey of Multilingual Neural Machine Translation”. In: *ACM Comput. Surv. CSUR* 53.5 (2020), pp. 1–38.

49. Z. Dai, Z. Yang, Y. Yang, W. W. Cohen, J. Carbonell, Q. V. Le, and R. Salakhutdinov. “Transformer-XL: Language Modeling with Longer-Term Dependency, 2019”. In: URL <https://arxiv.org/abs/1906.08238>. 2019.
50. S. Dathathri et al. “Plug and Play Language Models: A Simple Approach to Controlled Text Generation”. Mar. 3, 2020. arXiv: 1912.02164.
51. J. Deriu, A. Rodrigo, A. Otegi, G. Echegoyen, S. Rosset, E. Agirre, and M. Cieliebak. “Survey on Evaluation Methods for Dialogue Systems”. In: *Artif Intell Rev* 54.1 (Jan. 1, 2021), pp. 755–810. ISSN: 1573–7462. DOI: <https://doi.org/10.1007/s10462-020-09866-x>.
52. X. Dong, W. Yu, C. Zhu, and M. Jiang. “Injecting Entity Types into Entity-Guided Text Generation”. 2020. arXiv: 2009.13401.
53. Economist. “Huge “Foundation Models” Are Turbo-Charging AI Progress”. In: *The Economist* (June 11, 2022). ISSN: 0013-0613. URL: <https://www.economist.com/interactive/briefing/2022/06/11/huge-foundation-models-are-turbo-charging-ai-progress> (visited on 06/20/2022).
54. A. R. Fabbri, I. Li, T. She, S. Li, and D. R. Radev. “Multi-News: A Large-Scale Multi-Document Summarization Dataset and Abstractive Hierarchical Model”. 2019. arXiv: 1906.01749.
55. A. Fan. M2M. GitHub. 2020. URL: <https://github.com/pytorch/fairseq> (visited on 02/26/2021).
56. A. Fan, Y. Jernite, E. Perez, D. Grangier, J. Weston, and M. Auli. “ELI5: Long Form Question Answering”. 2019. arXiv: 1907.09190.
57. A. Fan et al. “Beyond English-Centric Multilingual Machine Translation”. 2020. arXiv: 2010.11125.
58. L. Fang, T. Zeng, C. Liu, L. Bo, W. Dong, and C. Chen. “Outline to Story: Fine-grained Controllable Story Generation from Cascaded Events”. 2021. arXiv: 2101.00822.
59. F. S. Finkbeiner Bernd. “Über die Schulter geschaut: Künstliche Intelligenz in der Softwareentwicklung”. In: *iX* 2021.8 (July 21, 2021), p. 40. ISSN: 0935-9680. URL: <https://www.heise.de/select/ix/2021/8/2111712093770753246> (visited on 02/19/2022).
60. L. Fröhling and A. Zubiaga. “Feature-Based Detection of Automated Language Models: Tackling GPT-2, GPT-3 and Grover”. In: *PeerJ Comput. Sci.* 7 (2021), e443.
61. R. Gabriel et al. “Further Advances in Open Domain Dialog Systems in the Third Alexa Prize Socialbot Grand Challenge”. In: *Alexa Prize Proc.* (2020).
62. L. Gao et al. “The Pile: An 800GB Dataset of Diverse Text for Language Modeling”. 2020. arXiv: 2101.00027.
63. L. Gao and J. Callan. “Unsupervised Corpus Aware Language Model Pre-training for Dense Passage Retrieval”. Aug. 12, 2021. arXiv: 2108.05540 [cs].
64. Y. Gao, J. Li, M. R. Lyu, and I. King. “Open-Retrieval Conversational Machine Reading”. Feb. 17, 2021. arXiv: 2102.08633.
65. A. Gatt and E. Krahermer. “Survey of the State of the Art in Natural Language Generation: Core Tasks, Applications and Evaluation”. In: *J. Artif. Intell. Res.* 61 (2018), pp. 65–170.
66. S. Gehrmann et al. “The GEM Benchmark: Natural Language Generation, Its Evaluation and Metrics”. In: *Proc. 1st Workshop Nat. Lang. Gener. Eval. Metr. GEM 2021*. ACL-GEM-IJCNLP 2021. Online: Association for Computational Linguistics, Aug. 2021, pp. 96–120. DOI: <https://doi.org/10.18653/v1/2021.gem-1.10>.
67. W. H. Gombaa and A. A. Fahmy. “A Survey of Text Similarity Approaches”. In: *Int. J. Comput. Appl.* 68.13 (2013), pp. 13–18.
68. I. Goodfellow et al. “Generative Adversarial Nets”. In: *Adv. Neural Inf. Process. Syst.* 2014, pp. 2672–2680.
69. B. F. Green Jr, A. K. Wolf, C. Chomsky, and K. Laughery. “Baseball: An Automatic Question-Answerer”. In: *Pap. Present. May 9–11 1961 West. Jt. IRE-AIEE-ACM Comput. Conf.* 1961, pp. 219–224.
70. X. Gu, K. M. Yoo, and J.-W. Ha. “Dialogbert: Discourse-aware Response Generation via Learning to Recover and Rank Utterances”. In: *Proc. AAAI* (2021). URL: <https://www.aaai.org/AAAI21Papers/AAAI-10083.GuX.pdf>.

71. W. Guan, I. Smetannikov, and M. Tianxing. “Survey on Automatic Text Summarization and Transformer Models Applicability”. In: *2020 Int. Conf. Control Robot. Intell. Syst.* 2020, pp. 176–184.
72. B. Guo, H. Wang, Y. Ding, W. Wu, S. Hao, Y. Sun, and Z. Yu. “Conditional Text Generation for Harmonious Human-Machine Interaction”. Dec. 24, 2020. arXiv: 1909.03409.
73. R. Guo, P. Sun, E. Lindgren, Q. Geng, D. Simcha, F. Chern, and S. Kumar. “Accelerating Large-Scale Inference with Anisotropic Vector Quantization”. In: *Int. Conf. Mach. Learn.* PMLR, 2020, pp. 3887–3896.
74. V. Gupta, K. Beckh, S. Giesselbach, D. Wegener, and T. Wirtz. “Supporting Verification of News Articles with Automated Search for Semantically Similar Articles”. 2021. arXiv: 2103.15581.
75. K. Guu, K. Lee, Z. Tung, P. Pasupat, and M.-W. Chang. “Realm: Retrieval-augmented Language Model Pre-Training”. 2020. arXiv: 2002.08909.
76. R. He, A. Ravula, B. Kanagal, and J. Ainslie. “RealFormer: Transformer Likes Residual Attention”. In: *arXiv e-prints* (2020), arXiv–2012.
77. X. He. “Parallel Refinements for Lexically Constrained Text Generation with BART”. 2021. arXiv: 2109.12487.
78. K. M. Hermann, T. Kocisky, E. Grefenstette, L. Espeholt, W. Kay, M. Suleyman, and P. Blunsom. “Teaching Machines to Read and Comprehend”. 2015. arXiv: 1506.03340.
79. A. Hern. *How Does Google’s AI Chatbot Work – and Could It Be Sentient?* | Google | *The Guardian*. June 13, 2022. URL: <https://www.theguardian.com/technology/2022/jun/13/howdoes-googles-ai-chatbot-work-and-could-it-be-sentient> (visited on 06/24/2022).
80. J. Hilton. WebGPT: *Improving the Factual Accuracy of Language Models through Web Browsing*. OpenAI. Dec. 16, 2021. URL: <https://openai.com/blog/improving-factual-accuracy/> (visited on 01/12/2022).
81. O. Honovich, L. Choshen, R. Aharoni, E. Neeman, I. Szpektor, and O. Abend. “\$ Q{\$2\$}\$: Evaluating Factual Consistency in Knowledge-Grounded Dialogues via Question Generation and Question Answering”. 2021. arXiv: 2104.08202.
82. J.-T. Huang et al. “Embedding-Based Retrieval in Facebook Search”. In: *Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.* 2020, pp. 2553–2561.
83. T. Iqbal and S. Qureshi. “The Survey: Text Generation Models in Deep Learning.” In: *J. King Saud Univ.-Comput. Inf. Sci.* (2020).
84. G. Izacard and E. Grave. “Leveraging Passage Retrieval with Generative Models for Open Domain Question Answering”. In: *Proc. 16th Conf. Eur. Chapter Assoc. Comput. Linguist. Main Vol.* EACL 2021. Online: Association for Computational Linguistics, Apr. 1, 2021, pp. 874–880. URL: <https://www.aclweb.org/anthology/2021.eacl-main.74> (visited on 06/16/2021).
85. G. Jawahar, M. Abdul-Mageed, and L. V. S. Lakshmanan. “Automatic Detection of Machine Generated Text: A Critical Survey”. Nov. 2, 2020. arXiv: 2011.01314 [cs].
86. H. Ji, P. Ke, S. Huang, F. Wei, X. Zhu, and M. Huang. “Language Generation with Multi-Hop Reasoning on Commonsense Knowledge Graph”. 2020. arXiv: 2009.11692.
87. J.-Y. Jiang, M. Zhang, C. Li, M. Bendersky, N. Golbandi, and M. Najork. “Semantic Text Matching for Long-Form Documents”. In: *World Wide Web Conf.* 2019, pp. 795–806.
88. D. Jin, Z. Jin, Z. Hu, O. Vehtomova, and R. Mihalcea. “Deep Learning for Text Style Transfer: A Survey”. In: *Comput. Linguist.* (2021), pp. 1–51.
89. D. Jin, Z. Jin, and R. Mihalcea. “Deep Learning for Text Attribute Transfer: A Survey”. 2020. arXiv: 2011.00416.
90. D. Jin, Z. Jin, J. T. Zhou, L. Orii, and P. Szolovits. “Hooks in the Headline: Learning to Generate Headlines with Controlled Styles”. 2020. arXiv: 2004.01980.
91. J. Johnson, M. Douze, and H. Jégou. “Billion-Scale Similarity Search with Gpus”. In: *IEEE Trans. Big Data* (2019).
92. M. Joshi, E. Choi, D. S. Weld, and L. Zettlemoyer. “Triviaqa: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension”. 2017. arXiv: 1705.03551.
93. D. Ju. *Learning from Data in the Mixed Adversarial Non-Adversarial Case: Finding the Helpers and Ignoring the Trolls*. Meta Research, Aug. 7, 2022. URL: <https://github.com/>

- [facebookresearch/ParlAI/blob/f9da661cf05496c50d18d8685a228faa574373ce/projects/trollhunting/finding\\_the\\_helpers.pdf](https://facebookresearch/ParlAI/blob/f9da661cf05496c50d18d8685a228faa574373ce/projects/trollhunting/finding_the_helpers.pdf) (visited on 08/07/2022).
94. V. Karpukhin, B. Oguz, S. Min, L. Wu, S. Edunov, D. Chen, and W.-t. Yih. “Dense Passage Retrieval for Open-Domain Question Answering”. 2020. arXiv: 2004.04906.
  95. W. S. El-Kassas, C. R. Salama, A. A. Rafea, and H. K. Mohamed. “Automatic Text Summarization: A Comprehensive Survey”. In: *Expert Syst. Appl.* 165 (2021), p. 113679.
  96. N. S. Keskar, B. McCann, L. R. Varshney, C. Xiong, and R. Socher. “CTRL: A Conditional Transformer Language Model for Controllable Generation”. Sept. 20, 2019. arXiv: 1909.05858.
  97. M. Khalifa, H. Elshahar, and M. Dymetman. “A Distributional Approach to Controlled Text Generation”. Dec. 21, 2020. arXiv: 2012.11635.
  98. T. Khan, A. Michalasz, and A. Akhuzada. “Fake News Outbreak 2021: Can We Stop the Viral Spread?” In: *Journal of Network and Computer Applications* 190 (Sept. 15, 2021), p. 103112. ISSN: 1084-8045. DOI: <https://doi.org/10.1016/j.jnca.2021.103112>.
  99. O. Khattab and M. Zaharia. “Colbert: Efficient and Effective Passage Search via Contextualized Late Interaction over Bert”. In: *Proc. 43rd Int. ACM SIGIR Conf. Res. Dev. Inf. Retr.* 2020, pp. 39–48.
  100. D. Kiela, H. Firooz, A. Mohan, V. Goswami, A. Singh, P. Ringshia, and D. Testuggine. “The Hateful Memes Challenge: Detecting Hate Speech in Multimodal Memes”. In: *Adv. Neural Inf. Process. Syst.* 33 (2020), pp. 2611–2624.
  101. B. Kim, H. Kim, and G. Kim. “Abstractive Summarization of Reddit Posts with Multi-Level Memory Networks”. 2018. arXiv: 1811.00783.
  102. M. Komeili, K. Shuster, and J. Weston. “Internet-Augmented Dialogue Generation”. July 15, 2021. arXiv: 2107.07566.
  103. M. Komeili, K. Shuster, and J. Weston. *Sea*. 2021. URL: <https://parl.ai/projects/sea/> (visited on 02/25/2022).
  104. R. Koncel-Kedziorski, D. Bekal, Y. Luan, M. Lapata, and H. Hajishirzi. “Text Generation from Knowledge Graphs with Graph Transformers”. 2019. arXiv: 1904.02342.
  105. B. Krause, A. D. Gotmare, B. McCann, N. S. Keskar, S. Joty, R. Socher, and N. F. Rajani. “Gedi: Generative Discriminator Guided Sequence Generation”. 2020. arXiv: 2009.06367.
  106. K. Krishna, A. Roy, and M. Iyyer. “Hurdles to Progress in Long-form Question Answering”. 2021. arXiv: 2103.06332.
  107. T. Kudo and J. Richardson. “Sentencepiece: A Simple and Language Independent Subword Tokenizer and Detokenizer for Neural Text Processing”. 2018. arXiv: 1808.06226.
  108. M. Kusner, Y. Sun, N. Kolkin, and K. Weinberger. “From Word Embeddings to Document Distances”. In: *Int. Conf. Mach. Learn.* PMLR, 2015, pp. 957–966.
  109. T. Kwiatkowski et al. “Natural Questions: A Benchmark for Question Answering Research”. In: *Trans. Assoc. Comput. Linguist.* 7 (2019), pp. 453–466.
  110. S. Läubli, R. Sennrich, and M. Volk. “Has Machine Translation Achieved Human Parity? A Case for Document-Level Evaluation”. 2018. arXiv: 1808.07048.
  111. J. Lee, M. Shim, S. Son, Y. Kim, C. Park, and H. Lim. “Empirical Study on BlenderBot 2.0 Errors Analysis in Terms of Model, Data and User-Centric Approach”. Jan. 10, 2022. arXiv: 2201.03239 [cs].
  112. B. Lemoine. *Is LaMDA Sentient? – An Interview*. Medium. June 11, 2022. URL: <https://cajundiscordian.medium.com/is-lamda-sentient-an-interview-ea64d916d917> (visited on 06/24/2022).
  113. D. Lepikhin et al. “Gshard: Scaling Giant Models with Conditional Computation and Automatic Sharding”. 2020. arXiv: 2006.16668.
  114. P. Lewis et al. “Retrieval-Augmented Generation for Knowledge-Intensive Nlp Tasks”. Dec. 7, 2020. arXiv: 2005.11401.
  115. C. Li, X. Gao, Y. Li, B. Peng, X. Li, Y. Zhang, and J. Gao. “Optimus: Organizing Sentences via Pre-Trained Modeling of a Latent Space”. 2020. arXiv: 2004.04092.
  116. J. Li, T. Tang, W. X. Zhao, and J.-R. Wen. “Pretrained Language Models for Text Generation: A Survey”. May 24, 2021. arXiv: 2105.10311.

117. X. Li, Y. Xia, X. Long, Z. Li, and S. Li. “Exploring Text-Transformers in Aaai 2021 Shared Task: Covid-19 Fake News Detection in English”. 2021. arXiv: 2101.02359.
118. Y. Li, C. Li, Y. Zhang, X. Li, G. Zheng, L. Carin, and J. Gao. “Complementary Auxiliary Classifiers for Label-Conditional Text Generation”. In: *Proc. AAAI Conf. Artif. Intell.* Vol. 34. 05. 2020, pp. 8303–8310.
119. Z. Li, Z. Li, J. Zhang, Y. Feng, and J. Zhou. “WeChat AI’s Submission for DSTC9 Interactive Dialogue Evaluation Track”. 2021. arXiv: 2101.07947.
120. Z. Li, J. Zhang, Z. Fei, Y. Feng, and J. Zhou. “Conversations Are Not Flat: Modeling the Dynamic Information Flow across Dialogue Utterances”. 2021. arXiv: 2106.02227.
121. O. Lieber, O. Sharir, B. Lentz, and Y. Shoham. “Jurassic-1: Technical Details and Evaluation”. In: (2021), p. 9. URL: [https://uploads-ssl.webflow.com/60fd4503684b466578c0d307/61138924626a6981ee09caf6\\_jurassic\\_tech\\_paper.pdf](https://uploads-ssl.webflow.com/60fd4503684b466578c0d307/61138924626a6981ee09caf6_jurassic_tech_paper.pdf).
122. M. Lili and O. Vechtomova. *Stylized Text Generation - ACL 2020 Tutorial*. 2020. URL: <https://sites.google.com/view/2020-stylized-text-generation/tutorial>.
123. J. Lin, R. Nogueira, and A. Yates. “Pretrained Transformers for Text Ranking: Bert and Beyond”. 2020. arXiv: 2010.06467.
124. J. Lin, R. Nogueira, and A. Yates. “Pretrained Transformers for Text Ranking: Bert and Beyond”. In: *Synth. Lect. Hum. Lang. Technol.* 14.4 (2021), pp. 1–325.
125. S. Lin, J. Hilton, and O. Evans. “TruthfulQA: Measuring How Models Mimic Human Falsehoods”. 2021. arXiv: 2109.07958.
126. Z. Lin, A. Madotto, Y. Bang, and P. Fung. “The Adapter-Bot: All-In-One Controllable Conversational Model”. In: (2021).
127. Z. Lin and M. Riedl. “Plug-and-Blend: A Framework for Controllable Story Generation with Blended Control Codes”. 2021. arXiv: 2104.04039.
128. P. J. Liu and Y. Zhao. PEGASUS: A State-of-the-Art Model for Abstractive Text Summarization. Google AI Blog. June 9, 2020. URL: <http://ai.googleblog.com/2020/06/pegasusstate-of-art-model-for.html> (visited on 02/18/2021).
129. S. Liu et al. “Towards Emotional Support Dialog Systems”. June 2, 2021. arXiv: 2106.01144 [cs]. 296 6 Foundation Models for Text Generation
130. X. Liu, K. Duh, L. Liu, and J. Gao. “Very Deep Transformers for Neural Machine Translation”. 2020. arXiv: 2008.07772.
131. Y. Liu, P. Liu, D. Radev, and G. Neubig. “BRIO: Bringing Order to Abstractive Summarization”. 2022. arXiv: 2203.16804.
132. E. Loginova, S. Varanasi, and G. Neumann. “Towards End-to-End Multilingual Question Answering”. In: *Inf. Syst. Front.* 23.1 (2021), pp. 227–241.
133. S. Longpre, Y. Lu, and J. Daiber. “MKQA: A Linguistically Diverse Benchmark for Multilingual Open Domain Question Answering”. In: *Trans. Assoc. Comput. Linguist.* 9 (2021), pp. 1389–1406.
134. R. Luscombe. “Google Engineer Put on Leave after Saying AI Chatbot Has Become Sentient”. In: *The Guardian. Technology* (June 12, 2022). ISSN: 0261-3077. URL: <https://www.theguardian.com/technology/2022/jun/12/google-engineer-ai-bot-sentient-blake-lemoine> (visited on 06/24/2022).
135. C. Ma, W. E. Zhang, M. Guo, H. Wang, and Q. Z. Sheng. “Multi-Document Summarization via Deep Learning Techniques: A Survey”. 2020. arXiv: 2011.04843.
136. X. Ma, K. Sun, R. Pradeep, and J. Lin. “A Replication Study of Dense Passage Retriever”. 2021. arXiv: 2104.05740.
137. S. MacAvaney, A. Yates, A. Cohan, and N. Goharian. “CEDR: Contextualized Embeddings for Document Ranking”. In: *Proc. 42nd Int. ACM SIGIR Conf. Res. Dev. Inf. Retr.* 2019, pp. 1101–1104.
138. J. Maillard, V. Karpukhin, F. Petroni, W.-t. Yih, B. Oğuz, V. Stoyanov, and G. Ghosh. “Multi-Task Retrieval for Knowledge-Intensive Tasks”. 2021. arXiv: 2101.00117.
139. Y. Mao, P. He, X. Liu, Y. Shen, J. Gao, J. Han, and W. Chen. “Generation-Augmented Retrieval for Open-Domain Question Answering”. 2020. arXiv: 2009.08553.

140. marco. *MS MARCO Passage Ranking Leaderboard*. Mar. 4, 2021. URL: <https://microsoft.github.io/msmarco/leaderboard/> (visited on 05/02/2021).
141. S. Maruf, F. Saleh, and G. Haffari. “A Survey on Document-level Neural Machine Translation: Methods and Evaluation”. In: *ACM Comput. Surv.* 54.2 (Mar. 5, 2021), 45:1–45:36. ISSN: 0360-0300. DOI: <https://doi.org/10.1145/3441691>.
142. S. Merity, C. Xiong, J. Bradbury, and R. Socher. “Pointer Sentinel Mixture Models”. 2016. arXiv: 1609.07843.
143. D. Metzler, Y. Tay, D. Bahri, and M. Najork. “Rethinking Search: Making Experts out of Dilettantes”. May 5, 2021. arXiv: 2105.02274 [cs].
144. R. Mihalcea and P. Tarau. “Textrank: Bringing Order into Text”. In: *Proc. 2004 Conf. Empir. Methods Nat. Lang. Process.* 2004, pp. 404–411.
145. S. Min, D. Chen, L. Zettlemoyer, and H. Hajishirzi. “Knowledge Guided Text Retrieval and Reading for Open Domain Question Answering”. Apr. 13, 2020. arXiv: 1911.03868.
146. S. Min, J. Michael, H. Hajishirzi, and L. Zettlemoyer. “AmbigQA: Answering Ambiguous Open-Domain Questions”. 2020. arXiv: 2004.10645.
147. S. Min et al. “NeurIPS 2020 EfficientQA Competition: Systems, Analyses and Lessons Learned”. 2021. arXiv: 2101.00133.
148. F. Monti, F. Frasca, D. Eynard, D. Mannion, and M. M. Bronstein. “Fake News Detection on Social Media Using Geometric Deep Learning”. 2019. arXiv: 1902.06673.
149. R. Nakano et al. “WebGPT: Browser-assisted Question-Answering with Human Feedback”. 2021. arXiv: 2112.09332.
150. C. Napoles, M. R. Gormley, and B. Van Durme. “Annotated Gigaword”. In: *Proc. Jt. Workshop Autom. Knowl. Base Constr. Web-Scale Knowl. Extr. AKBC-WEKEX*. 2021, pp. 95–100.
151. S. Narayan, S. B. Cohen, and M. Lapata. “Don’t Give Me the Details, Just the Summary! Topic-Aware Convolutional Neural Networks for Extreme Summarization”. In: *Proc. 2018 Conf. Empir. Methods Nat. Lang. Process. EMNLP 2018*. Brussels, Belgium: Association for Computational Linguistics, Oct. 2018, pp. 1797–1807. DOI: <https://doi.org/10.18653/v1/D18-1206>.
152. P. Nayak. “Understanding Searches Better than Ever Before”. In: *Google Blog Oct. 25 (2019)*.
153. T. Nguyen, M. Rosenberg, X. Song, J. Gao, S. Tiwary, R. Majumder, and L. Deng. “MS MARCO: A Human Generated Machine Reading Comprehension Dataset”. In: *CoCo NIPS*. 2016.
154. R. Nogueira, Z. Jiang, and J. Lin. “Document Ranking with a Pretrained Sequence-to-Sequence Model”. 2020. arXiv: 2003.06713.
155. R. Nogueira, W. Yang, K. Cho, and J. Lin. “Multi-Stage Document Ranking with Bert”. 2019. arXiv: 1910.14424.
156. R. Nogueira, W. Yang, J. Lin, and K. Cho. “Document Expansion by Query Prediction”. 2019. arXiv: 1904.08375.
157. J. Novet. *Microsoft and OpenAI Have a New A.I. Tool That Will Give Coding Suggestions to Software Developers*. CNBC. June 29, 2021. URL: <https://www.cnbc.com/2021/06/29/microsoft-github-copilot-ai-offers-coding-suggestions.html> (visited on 02/19/2022).
158. B. Oguz et al. “Unified Open-Domain Question Answering with Structured and Unstructured Knowledge”. 2020. arXiv: 2012.14610.
159. OpenAI. *Example Prompts OpenAI API*. Feb. 19, 2022. URL: <https://beta.openai.com> (visited on 02/19/2022).
160. OpenAi. *Prompt Examples for GPT-3*. Sept. 3, 2021. URL: <https://beta.openai.com/examples> (visited on 09/03/2021).
161. E. Orbach and Y. Goldberg. “Facts2Story: Controlling Text Generation by Key Facts”. 2020. arXiv: 2012.04332.
162. L. Ouyang et al. “Training Language Models to Follow Instructions with Human Feedback”. Jan. 31, 2022. arXiv: 2203.02155.
163. Y. M. Palenzuela. *Awesome GPT-3*. Feb. 19, 2022. URL: <https://github.com/elyase/awesome-gpt3> (visited on 02/19/2022).

164. P. Papalampidi, K. Cao, and T. Kocisky. “Towards Coherent and Consistent Use of Entities in Narrative Generation”. Feb. 3, 2022. arXiv: 2202.01709 [cs].
165. D. Paperno et al. “The LAMBADA Dataset: Word Prediction Requiring a Broad Discourse Context”. June 20, 2016. arXiv: 1606.06031 [cs].
166. T. Parshakova, J.-M. Andreoli, and M. Dymetman. “Distributional Reinforcement Learning for Energy-Based Sequential Models”. Dec. 18, 2019. arXiv: 1912.08517.
167. N. Peng, M. Ghazvininejad, J. May, and K. Knight. “Towards Controllable Story Generation”. In: *Proc. First Workshop Storytell.* 2018, pp. 43–49.
168. X. Peng, S. Li, S. Wiegrefe, and M. Riedl. “Inferring the Reader: Guiding Automated Story Generation with Commonsense Reasoning”. 2021. arXiv: 2105.01311.
169. M. Ponza, L. Del Corro, and G. Weikum. “Facts That Matter”. In: *Proc. 2018 Conf. Empir. Methods Nat. Lang. Process.* 2018, pp. 1043–1048.
170. S. Prabhunoye, A. W. Black, and R. Salakhutdinov. “Exploring Controllable Text Generation Techniques”. 2020. arXiv: 2005.01822.
171. QAngaroo. *WikiHop Leaderboard*. Mar. 2, 2021. URL: <http://qangaroo.cs.ucl.ac.uk/leaderboard.html> (visited on 03/02/2021).
172. P. Racsco. “Fake News Identification”. In: *Soc. Econ.* -1 (aop Nov. 11, 2021). ISSN: 1588-9726, 1588-970X. DOI: <https://doi.org/10.1556/204.2021.00020>.
173. A. Radford, J. Wu, D. Amodei, D. Amodei, J. Clark, M. Brundage, and I. Sutskever. “Better Language Models and Their Implications”. In: *OpenAI Blog* (2019). URL: <https://openai.com/blog/better-language-models>.
174. A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. “Language Models Are Unsupervised Multitask Learners”. In: *OpenAI blog* 1.8 (2019), p. 9.
175. J. W. Rae et al. “Scaling Language Models: Methods, Analysis & Insights from Training Gopher”. In: *ArXiv Prepr. ArXiv211211446* (Dec. 8, 2021), p. 118.
176. J. W. Rae, G. Irving, and L. Weidinger. *Language modelling at scale: Gopher, ethical considerations, and retrieval*. Deepmind. Dec. 8, 2021. URL: <https://deepmind.com/blog/article/language-modelling-at-scale> (visited on 12/16/2021).
177. C. Raffel et al. “Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer”. In: *J. Mach. Learn. Res.* 21.140 (2020), pp. 1–67.
178. P. Rajpurkar. *SQUAD: The Stanford Question Answering Dataset*. Mar. 3, 2021. URL: <https://rajpurkar.github.io/SQuAD-explorer/> (visited on 03/03/2021).
179. P. Rajpurkar, R. Jia, and P. Liang. “Know What You Don’t Know: Unanswerable Questions for SQuAD”. 2018. arXiv: 1806.03822.
180. S. Rao and J. Tetreault. “Gyafc Dataset: Corpus, Benchmarks and Metrics for Formality Style Transfer”. 2018. arXiv: 1803.06535.
181. H. Rashkin, A. Celikyilmaz, Y. Choi, and J. Gao. “PlotMachines: Outline-Conditioned Generation with Dynamic Plot State Tracking”. Oct. 9, 2020. arXiv: 2004.14967 [cs].
182. E. Reif, D. Ippolito, A. Yuan, A. Coenen, C. Callison-Burch, and J. Wei. “A Recipe for Arbitrary Text Style Transfer with Large Language Models”. 2021. arXiv: 2109.03910.
183. N. Reimers and I. Gurevych. “Sentence-Bert: Sentence Embeddings Using Siamese Bert-Networks”. 2019. arXiv: 1908.10084.
184. R. Ren et al. “RocketQAV2: A Joint Training Method for Dense Passage Retrieval and Passage Re-ranking”. 2021. arXiv: 2110.07367.
185. L. F. Ribeiro, M. Schmitt, H. Schütze, and I. Gurevych. “Investigating Pretrained Language Models for Graph-to-Text Generation”. 2020. arXiv: 2007.08426.
186. S. Robertson and H. Zaragoza. *The Probabilistic Relevance Framework: BM25 and Beyond*. Now Publishers Inc, 2009.
187. T. Rohde, X. Wu, and Y. Liu. “Hierarchical Learning for Generation with Long Source Sequences”. 2021. arXiv: 2104.07545.
188. S. Roller, E. Dinan, and J. Weston. *A state-of-the-art open source chatbot*. Apr. 29, 2020. URL: <https://ai.facebook.com/blog/state-of-the-art-open-source-chatbot/> (visited on 11/21/2020).
189. S. Roller et al. “Open-Domain Conversational Agents: Current Progress, Open Problems, and Future Directions”. 2020. arXiv: 2006.12442.

190. S. Roller et al. “Recipes for Building an Open-Domain Chatbot”. 2020. arXiv: 2004.13637.
191. S. Rose, D. Engel, N. Cramer, and W. Cowley. “Automatic Keyword Extraction from Individual Documents”. In: *Text Min. Appl. Theory I* (2010), pp. 1–20.
192. S. Ruder. *Multi-Domain Multilingual Question Answering*. Sebastian Ruder. Dec. 6, 2021. URL: <https://ruder.io/multi-qa-tutorial/> (visited on 02/10/2022).
193. S. Ruder and A. Sil. “Multi-Domain Multilingual Question Answering”. In: *Proc. 2021 Conf. Empir. Methods Nat. Lang. Process. Tutor. Abstr.* Punta Cana, Dominican Republic & Online: Association for Computational Linguistics, Nov. 2021, pp. 17–21. URL: <https://aclanthology.org/2021.emnlp-tutorials.4> (visited on 11/24/2021).
194. D. S. Sachan, S. Reddy, W. Hamilton, C. Dyer, and D. Yogatama. “End-to-End Training of Multi-Document Reader and Retriever for Open-Domain Question Answering”. June 9, 2021. arXiv: 2106.05346.
195. A. B. Sai, A. K. Mohankumar, and M. M. Khapra. “A Survey of Evaluation Metrics Used for NLG Systems”. 2020. arXiv: 2008.12009.
196. K. Sakaguchi, C. Bhagavatula, R. L. Bras, N. Tandon, P. Clark, and Y. Choi. “Proscript: Partially Ordered Scripts Generation via Pre-Trained Language Models”. 2021. arXiv: 2104.08251.
197. B. Schwartz. Google: BERT Now Used on Almost Every English Query. Search Engine Land. Oct. 15, 2020. URL: <https://searchengineland.com/google-bert-used-on-almost-everyenglish-query-342193> (visited on 01/24/2021).
198. T. Scialom, P.-A. Dray, S. Lamprier, B. Piwowarski, and J. Staiano. “MLSUM: The Multilingual Summarization Corpus”. 2020. arXiv: 2004.14900.
199. P. Sen and A. Saffari. “What Do Models Learn from Question Answering Datasets?” 2020. arXiv: 2004.03490.
200. R. Sennrich, B. Haddow, and A. Birch. “Improving Neural Machine Translation Models with Monolingual Data”. 2015. arXiv: 1511.06709.
201. R. Sepúlveda-Torres, M. Vicente, E. Saquete, E. Lloret, and M. Palomar. “Exploring Summarization to Enhance Headline Stance Detection”. In: *Nat. Lang. Process. Inf. Syst.* Ed. by E. Métais, F. Meziane, H. Horacek, and E. Kapetanios. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2021, pp. 243–254. ISBN: 978-3-030-80599-9. DOI: [https://doi.org/10.1007/978-3-030-80599-9\\_22](https://doi.org/10.1007/978-3-030-80599-9_22).
202. T. Sherborne and M. Lapata. “Zero-Shot Cross-lingual Semantic Parsing”. 2021. arXiv: 2104.07554.
203. P. Shi, R. Zhang, H. Bai, and J. Lin. “Cross-Lingual Training with Dense Retrieval for Document Retrieval”. Sept. 3, 2021. arXiv: 2109.01628 [cs].
204. K. Shuster, M. Komeili, L. Adolphs, S. Roller, A. Szlám, and J. Weston. *Language Models That Seek for Knowledge: Modular Search & Generation for Dialogue and Prompt Completion*. Mar. 29, 2022. arXiv: 2203.13224 [cs].
205. K. Shuster, S. Poff, M. Chen, D. Kiela, and J. Weston. “Retrieval Augmentation Reduces Hallucination in Conversation”. 2021. arXiv: 2104.07567.
206. K. Shuster and J. Xu. *Blender BlenderBot 3: A 175B parameter, publicly available chatbot that improves its skills and safety over time*. Aug. 5, 2022. URL: <https://ai.facebook.com/blog/blenderbot-3-a-175b-parameter-publicly-available-chatbot-that-improves-its-skills-and-safety-over-time/> (visited on 08/07/2022).
207. H. Singh, G. Verma, A. Garimella, and B. V. Srinivasan. “DRAG: Director-Generator Language Modelling Framework for Non-Parallel Author Stylized Rewriting”. 2021. arXiv: 2101.11836.
208. C. Song, N. Ning, Y. Zhang, and B. Wu. “A Multimodal Fake News Detection Model Based on Crossmodal Attention Residual and Multichannel Convolutional Neural Networks”. In: *Inf. Process. Manag.* 58.1 (2021), p. 102437.
209. Sserdoubleh. *Large-Scale Open Domain KNOWledge Grounded conVERsation System Based on PaddlePaddle*. PaddlePaddle, Apr. 26, 2021. URL: <https://github.com/PaddlePaddle/Knover> (visited on 05/08/2021).

210. M. Stern, W. Chan, J. Kiros, and J. Uszkoreit. “Insertion Transformer: Flexible Sequence Generation via Insertion Operations”. In: *Int. Conf. Mach. Learn.* PMLR, 2019, pp. 5976–5985.
211. N. Stiennon et al. “Learning to Summarize with Human Feedback”. In: *Adv. Neural Inf. Process. Syst.* 33 (Feb. 15, 2022), pp. 3008–3021.
212. B. Sun and K. Li. “Neural Dialogue Generation Methods in Open Domain: A Survey”. In: *Nat. Lang. Process. Res.* 1.3–4 (2021), pp. 56–70.
213. P. Sun. *Announcing ScaNN: Efficient Vector Similarity Search*. Google AI Blog. July 28, 2020. URL: <http://ai.googleblog.com/2020/07/announcing-scann-efficient-vector.html> (visited on 02/18/2021).
214. X. Sun, C. Fan, Z. Sun, Y. Meng, F. Wu, and J. Li. “Summarize, Outline, and Elaborate: Long-Text Generation via Hierarchical Supervision from Extractive Summaries”. 2020. arXiv: 2010.07074.
215. I. Sutskever, O. Vinyals, and Q. V. Le. “Sequence to Sequence Learning with Neural Networks”. In: *Adv. Neural Inf. Process. Syst.* 2014, pp. 3104–3112.
216. A. A. Syed, F. L. Gaol, and T. Matsuo. “A Survey of the State-of-the-Art Models in Neural Abstractive Text Summarization”. In: *IEEE Access* 9 (2021), pp. 13248–13265.
217. B. Syed, G. Verma, B. V. Srinivasan, A. Natarajan, and V. Varma. “Adapting Language Models for Non-Parallel Author-Stylized Rewriting”. In: *Proc. AAAI Conf. Artif. Intell.* Vol. 34. 05. 2020, pp. 9008–9015.
218. O. Tafjord and P. Clark. “General-Purpose Question-Answering with Macaw”. Sept. 6, 2021. arXiv: 2109.02593 [cs].
219. B. Tan, Z. Yang, M. Al-Shedivat, E. Xing, and Z. Hu. “Progressive Generation of Long Text with Pretrained Language Models”. In: *Proc. 2021 Conf. North Am. Chapter Assoc. Comput. Linguist. Hum. Lang. Technol.* 2021, pp. 4313–4324.
220. B. Tan, Z. Yang, M. Al-Shedivat, E. P. Xing, and Z. Hu. “Progressive Generation of Long Text”. 2020. arXiv: 2006.15720.
221. N. Thakur, N. Reimers, A. Rücklé, A. Srivastava, and I. Gurevych. “BEIR: A Heterogenous Benchmark for Zero-shot Evaluation of Information Retrieval Models”. Sept. 7, 2021. arXiv: 2104.08663.
222. R. Thoppilan et al. “LaMDA: Language Models for Dialog Applications”. Feb. 10, 2022. arXiv: 2201.08239 [cs].
223. A. Toral. “Reassessing Claims of Human Parity and Super-Human Performance in Machine Translation at WMT 2019”. 2020. arXiv: 2005.05738.
224. C. Tran, S. Bhosale, J. Cross, P. Koehn, S. Edunov, and A. Fan. “Facebook AI WMT21 News Translation Task Submission”. 2021. arXiv: 2108.03265.
225. C. Tran, J. Cross, S. Bhosale, and A. Fan. *The first-ever multilingual model to win WMT, beating out bilingual models*. Nov. 10, 2021. URL: <https://ai.facebook.com/blog/the-first-ever-multilingual-model-to-win-wmt-beating-out-bilingual-models/> (visited on 02/08/2022).
226. triviaQA. *Papers with Code - TriviaQA Benchmark (Question Answering)*. June 27, 2022. URL: <https://paperswithcode.com/sota/question-answering-on-triviaqa> (visited on 06/27/2022).
227. L. Vailshery. *Number of Digital Voice Assistants in Use Worldwide 2019–2024*. Jan. 22, 2021. URL: <https://www.statista.com/statistics/973815/worldwide-digital-voice-assistant-inuse/>.
228. A. Vaswani et al. “Attention Is All You Need”. In: *Adv. Neural Inf. Process. Syst.* 2017, pp. 5998–6008.
229. S. J. Vaughan-Nichols. *GitHub’s Copilot Faces First Open Source Copyright Lawsuit*. Nov. 11, 2022. URL: [https://www.theregister.com/2022/11/11/githubs\\_copilot\\_opinion/](https://www.theregister.com/2022/11/11/githubs_copilot_opinion/) (visited on 12/17/2022).
230. M. Völske, M. Potthast, S. Syed, and B. Stein. “TL;DR: Mining Reddit to Learn Automatic Summarization”. In: *Proc. Workshop New Front. Summ.* Copenhagen, Denmark: Association for Computational Linguistics, Sept. 2017, pp. 59–63. DOI: <https://doi.org/10.18653/v1/W17-4508>.

231. J. Weizenbaum. “ELIZA—a Computer Program for the Study of Natural Language Communication between Man and Machine”. In: *Commun. ACM* 9.1 (1966), pp. 36–45.
232. J. Welbl, P. Stenetorp, and S. Riedel. “Constructing Datasets for Multi-Hop Reading Comprehension across Documents”. In: *Trans. Assoc. Comput. Linguist.* 6 (2018), pp. 287–302.
233. J. Weston and K. Shuster. *Blender Bot 2.0: An open source chatbot that builds long-term memory and searches the internet*. July 16, 2021. URL: <https://ai.facebook.com/blog/blender-bot-2-an-open-source-chatbot-that-builds-long-term-memory-and-searches-the-internet/> (visited on 02/25/2022).
234. Wikipedia. *Narrative*. In: *Wikipedia*. Apr. 10, 2021. URL: <https://en.wikipedia.org/w/index.php?title=Narrative&oldid=1017065535> (visited on 04/21/2021).
235. M. Woolf. *How To Make Custom AI-Generated Text With GPT-2*. Max Woolf’s Blog. Sept. 4, 2019. URL: <https://minimaxir.com/2019/09/howto-gpt2/> (visited on 04/29/2021).
236. J. Wu, L. Ouyang, D. M. Ziegler, N. Stiennon, R. Lowe, J. Leike, and P. Christiano. “Recursively Summarizing Books with Human Feedback”. 2021. arXiv: 2109.10862.
237. W. Xiao, I. Beltagy, G. Carenini, and A. Cohan. “Primer: Pyramid-based Masked Sentence Pre-Training for Multi-Document Summarization”. 2021. arXiv: 2110.08499.
238. L. Xiong et al. “Approximate Nearest Neighbor Negative Contrastive Learning for Dense Text Retrieval”. 2020. arXiv: 2007.00808.
239. W. Xiong, M. Yu, S. Chang, X. Guo, and W. Y. Wang. “Improving Question Answering over Incomplete Kbs with Knowledge-Aware Reader”. 2019. arXiv: 1905.07098.
240. F. F. Xu, U. Alon, G. Neubig, and V. J. Hellendoorn. “A Systematic Evaluation of Large Language Models of Code”. In: *Proc. 6th ACM SIGPLAN Int. Symp. Mach. Program.* 2022, pp. 1–10.
241. J. Xu, D. Ju, M. Li, Y.-L. Boureau, J. Weston, and E. Dinan. “Recipes for Safety in Open-Domain Chatbots”. 2020. arXiv: 2010.07079.
242. J. Xu, A. Szlam, and J. Weston. “Beyond Goldfish Memory: Long-Term Open-Domain Conversation”. July 15, 2021. arXiv: 2107.07567.
243. P. Xu, M. Patwary, M. Shoenybi, R. Puri, P. Fung, A. Anandkumar, and B. Catanzaro. “MEGATRON-CNTRL: Controllable Story Generation with External Knowledge Using Large-Scale Language Models”. Oct. 2, 2020. arXiv: 2010.00840 [cs]. 6.6 *Dialog Systems* 301
244. Y. Xu et al. “GSPMD: General and Scalable Parallelization for ML Computation Graphs”. Dec. 23, 2021. arXiv: 2105.04663 [cs].
245. L. Xue et al. “Byt5: Towards a Token-Free Future with Pre-Trained Byte-to-Byte Models”. In: *Trans. Assoc. Comput. Linguist.* 10 (2022), pp. 291–306.
246. L. Xue et al. “mT5: A Massively Multilingual Pre-Trained Text-to-Text Transformer”. 2020. arXiv: 2010.11934.
247. L. Yang, M. Zhang, C. Li, M. Bendersky, and M. Najork. “Beyond 512 Tokens: Siamese Multi-Depth Transformer-Based Hierarchical Encoder for Long-Form Document Matching”. In: *Proc. 29th ACM Int. Conf. Inf. Knowl. Manag.* 2020, pp. 1725–1734.
248. S. Yang, Y. Wang, and X. Chu. “A Survey of Deep Learning Techniques for Neural Machine Translation”. 2020. arXiv: 2002.07526.
249. Z. Yang, P. Qi, S. Zhang, Y. Bengio, W. W. Cohen, R. Salakhutdinov, and C. D. Manning. “Hotpotqa: A Dataset for Diverse, Explainable Multi-Hop Question Answering”. 2018. arXiv: 1809.09600.
250. X. Yi, Z. Liu, W. Li, and M. Sun. “Text Style Transfer via Learning Style Instance Supported Latent Space”. In: *IJCAI*, 2020.
251. W. Yu, C. Zhu, Z. Li, Z. Hu, Q. Wang, H. Ji, and M. Jiang. “A Survey of Knowledge-Enhanced Text Generation”. July 5, 2021. arXiv: 2010.04389.
252. C. Yuan, Q. Ma, W. Zhou, J. Han, and S. Hu. “Early Detection of Fake News by Utilizing the Credibility of News, Publishers, and Users Based on Weakly Supervised Learning”. 2020. arXiv: 2012.04233.

253. M. Zaheer et al. “Big Bird: Transformers for Longer Sequences”. In: *Adv. Neural Inf. Process. Syst.* 33 (Jan. 8, 2021).
254. M. Zaib, W. E. Zhang, Q. Z. Sheng, A. Mahmood, and Y. Zhang. “Conversational Question Answering: A Survey”. 2021. arXiv: 2106.00874.
255. R. Zellers, A. Holtzman, H. Rashkin, Y. Bisk, A. Farhadi, F. Roesner, and Y. Choi. “Defending against Neural Fake News”. Dec. 11, 2020. arXiv: 1905.12616.
256. H. Zhang, Y. Gong, Y. Shen, W. Li, J. Lv, N. Duan, and W. Chen. “Poolingformer: Long Document Modeling with Pooling Attention”. May 10, 2021. arXiv: 2105.04371 [cs].
257. H. Zhang, H. Song, S. Li, M. Zhou, and D. Song. “A Survey of Controllable Text Generation Using Transformer-based Pre-trained Language Models”. 2022. arXiv: 2201.05337.
258. X. Zhang, X. Ma, P. Shi, and J. Lin. “Mr. TyDi: A Multi-lingual Benchmark for Dense Retrieval”. Nov. 8, 2021. arXiv: 2108.08787 [cs].
259. Y. Zhang, Y. Cao, M. Mahdih, J. Zhao, and Y. Wu. “Improving Longer-range Dialogue State Tracking”. 2021. arXiv: 2103.00109.
260. Y. Zhang, T. Ge, and X. Sun. “Parallel Data Augmentation for Formality Style Transfer”. May 4, 2020. arXiv: 2005.07522.
261. Y. Zhang, G. Wang, C. Li, Z. Gan, C. Brockett, and B. Dolan. “Pointer: Constrained Text Generation via Insertion-Based Generative Pre-Training”. Sept. 27, 2020. arXiv: 2005.00558.
262. Y. Zhang et al. “Dialogpt: Large-scale Generative Pre-Training for Conversational Response Generation”. May 2, 2020. arXiv: 1911.00536.
263. A. Zhavoronkov. *Wu Dao 2.0 - Bigger, Stronger, Faster AI From China*. Forbes. July 19, 2021. URL: <https://www.forbes.com/sites/alexzhavoronkov/2021/07/19/wu-dao-20biggerstronger-faster-ai-from-china/> (visited on 07/29/2021).
264. L. Zhou, J. Gao, D. Li, and H.-Y. Shum. “The Design and Implementation of Xiaoice, an Empathetic Social Chatbot”. In: *Comput. Linguist.* 46.1 (2020), pp. 53–93.
265. F. Zhu, W. Lei, C. Wang, J. Zheng, S. Poria, and T.-S. Chua. “Retrieving and Reading: A Comprehensive Survey on Open-Domain Question Answering”. 2021. arXiv: 2101.00774.
266. J. Zhu. *Bing Delivers Its Largest Improvement in Search Experience Using Azure GPUs*. Nov. 18, 2019. URL: <https://azure.microsoft.com/en-us/blog/bing-delivers-its-largest-improvement-in-search-experience-using-azure-gpus/> (visited on 01/08/2021).
267. Y. Zhu, R. Kiros, R. Zemel, R. Salakhutdinov, R. Urtasun, A. Torralba, and S. Fidler. “Aligning Books and Movies: Towards Story-like Visual Explanations by Watching Movies and Reading Books”. In: *Proc. IEEE Int. Conf. Comput. Vis.* 2015, pp. 19–27.
268. Y. Zhu, L. Pang, Y. Lan, H. Shen, and X. Cheng. “Adaptive Information Seeking for Open-Domain Question Answering”. Sept. 14, 2021. arXiv: 2109.06747 [cs].
269. Z. M. Ziegler, L. Melas-Kyriazi, S. Gehrmann, and A. M. Rush. “Encoder-Agnostic Adaptation for Conditional Language Generation”. 2019. arXiv: 1908.06938.
270. B. Zoph et al. “Designing Effective Sparse Expert Models”. 2022. arXiv: 2202.08906.

**Open Access** This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter’s Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter’s Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.

