Digital Discovery

ROYAL SOCIETY OF CHEMISTRY

PAPER

View Article Online
View Journal | View Issue

Check for updates

Cite this: *Digital Discovery*, 2023, **2**, 1850

Received 12th June 2023
Accepted 4th October 2023

DOI: 10.1039/d3dd00112a

rsc.li/digitaldiscovery

Domain-specific chatbots for science using embeddings†

Kevin G. Yager *

Large language models (LLMs) have emerged as powerful machine-learning systems capable of handling a myriad of tasks. Tuned versions of these systems have been turned into chatbots that can respond to user queries on a vast diversity of topics, providing informative and creative replies. However, their application to physical science research remains limited owing to their incomplete knowledge in these areas, contrasted with the needs of rigor and sourcing in science domains. Here, we demonstrate how existing methods and software tools can be easily combined to yield a domain-specific chatbot. The system ingests scientific documents in existing formats, and uses text embedding lookup to provide the LLM with domain-specific contextual information when composing its reply. We similarly demonstrate that existing image embedding methods can be used for search and retrieval across publication figures. These results confirm that LLMs are already suitable for use by physical scientists in accelerating their research efforts.

1. Introduction

Artificial intelligence and machine-learning (AI/ML) methods are growing in sophistication and capability. The application of these methods to the physical sciences is correspondingly seeing enormous growth.[1] Recent years have seen the convergence of several new trends. Generative AI seeks to create novel outputs that conform to the structure of training data,[2,3] for instance enabling image synthesis[4–6] or text generation. Large language models (LLMs) are generative neural networks trained on text completion, but which can be used for a variety of tasks, including sentiment analysis, code completion, document generation, or for interactive chatbots that respond to users in natural language.[7] The most successful implementations of this concept—such as the generative pre-trained transformer (GPT)[8]— exploit the transformer architecture,[9] which has a self-attention mechanism, allowing the model to weigh the relevance of each input in a sequence and capture the contextual dependencies between words regardless of their distance from each other in the text sequence. LLMs are part of a general trend in ML towards foundation models—extensive training of large deep neural networks on enormous datasets in a task-agnostic manner.[7,10] The performance of LLMs increases with the scale of the training data, network size, and training time. There is growing evidence that LLMs are not merely reproducing surface statistics, but are instead learning a meaningful world model.[11–13] Correspondingly, training shows evidence of sudden leaps in performance and corresponding development of surprising new capabilities, suggesting the emergent learning of generalized concepts with more abstraction and sophistication.[14–18]

Recent work has shown how reinforcement learning using human feedback (RLHF)[19] can be used to further tailor LLMs into generating responses aligned with human desires for helpful and informative text response to user queries. In this way, several efforts have demonstrated high-quality chatbots that can engage in remarkably productive discussion (the most prominent being the ChatGPT system produced by OpenAI). These text response systems allow a user to provide input text—which might include instructions, background information, and user question—and solicit a text completion that answers the query. Key engineering aspects of using such systems are managing the finite context window (maximum size available for input text and generated response) and prompt engineering (crafting the input text to elicit the desired behavior). The field of LLMs and chat interfaces is advancing rapidly. The prompting process can be elaborated to induce more sophisticated responses akin to deliberation, by using self-analysis of generation quality,[20,21] or generating chains of thought through iterative self-prompting.[22,23] Chatbots can be augmented with access to external tools through APIs (application programming interfaces),[24–32] and can be turned into task-oriented autonomous agents by allowing them to iteratively propose and execute solutions.[28,33–35]

As these phenomenal capabilities are demonstrated, it is natural to ask how they can be tailored specifically to accelerate research in the physical sciences. The most obvious option is to

Center for Functional Nanomaterials, Brookhaven National Laboratory, Upton, New York 11973, USA. E-mail: kyager@bnl.gov

† Electronic supplementary information (ESI) available: Examples of chatbot responses to various user queries. See DOI: **https://doi.org/10.1039/d3dd00112a**

train an LLM on the enormous corpus of scientific publications, thereby producing a chatbot that can converse on science topics. The scale and cost of training an LLM from scratch is daunting, excluding all but the largest research groups. Meta AI trained Galactica, an LLM optimized for Science;[36] however its public availability was short-lived, owing to backlash associated with its propensity for fabricating plausible-sounding but ultimately nonsense scientific text. This "hallucination" behavior is a key challenge in deploying LLMs, arising from the inherently interpolative nature of neural networks, combined with human preferences selecting for confident answers in the RLHF step.

Instead of training an LLM from scratch, another option is to fine-tune an existing model on additional domain-specific data. Several efforts have demonstrated highly efficient strategies for performing this step, most notably the low-rank adaptation method,[37,38] which uses rank decomposition matrices to reduce the number of parameters during retraining. Even with such efficiency gains, it remains daunting for the non-expert to deploy, fine-tune, and utilize an LLM locally. In particular, physical scientists typically lack the expertise or inclination to take on such efforts, which suggests that domain-specific chatbots optimized for physical sciences must wait for attention from larger research efforts.

Here, we demonstrate how existing and available tools can be easily chained together to build a domain-adapted chatbot that can discuss scientific topics. Our example implementation can take advantage of available scientific documents in the portable document format (PDF), does not require LLM fine-tuning, and addresses the hallucination problem by making document text extractions available to the chatbot through the input prompt. A critical aspect of scientific documents is the technical figures contained within. We demonstrate how image embedding methods can be used to find semantically related content among publication figures or image datasets. Together, these demonstrations suggest that domain-specific chatbots can already be easily deployed by any researcher in the physical sciences, and that there is a corresponding opportunity to accelerate the fundamental research enterprise by embracing these new tools.

2. Results and discussion

2.1 Chatbot design

In order to demonstrate the viability of domain-specific chatbots for science topics, we developed a demonstration implementation. Although simple and unrefined, this demo system allows us to investigate utility, and acts as a blueprint for other researchers wishing to deploy similar systems. The core operating principle of our implementation is to take advantage of text embeddings to retrieve potentially-relevant text extracts ("chunks") from the corpus of domain-specific documents.[39,40] Text embedding is a natural language processing (NLP) method whereby text is converted into a real-valued vector that encodes the meaning. This conversion is normally performed using a neural network trained to convert text into a concise vector representation, which can then be thought of as a semantic latent space. For instance, words that are close in the embedding space are expected to be similar in meaning.

A typical LLM chatbot lookup involves constructing an input prompt that involves the user query, where one optionally prepends some additional contextual information (such as the chat history, so that the LLM can assess the context of the most recent user comment). In the embedding strategy, we take advantage of the space provided by the context window, adding in text chunks relevant to the query. Procedurally (Fig. 1), this involves first computing the text embedding of the user query (q). This embedding vector ($\vec{v_q}$) is compared to the pre-computed embeddings across all text chunks (stored in a database). Semantically relevant text chunks are identified using the cosine similarity between the user query and each chunk ($\vec{v_c}$) :

$$s_{qc} = \frac{\vec{v_q} \times \vec{v_c}}{\|\vec{v_q}\|\|\vec{v_c}\|} \tag{1}$$

The cosine similarity measures the angle between vectors, and thus assesses whether they point in the same direction in the semantic space. This thus provides a measure of thematic similarity of the two texts being analyzed, as opposed to measuring the similarity in full meaning between the two. A small set (5–10) of the most relevant chunks are concatenated and prepended to the user query. This constructed prompt is then sent to the LLM, which generates a coherent response to the query using the available text. The availability of domain-specific text segments allows more specific and meaningful response, including direct quotation and citation of source.

While conceptually simple, this design involves several implementation details that must be considered. The first is document format. The version of record for scientific publications tends to be stored in the portable document format (PDF), which is optimized for consistent layout and readability across

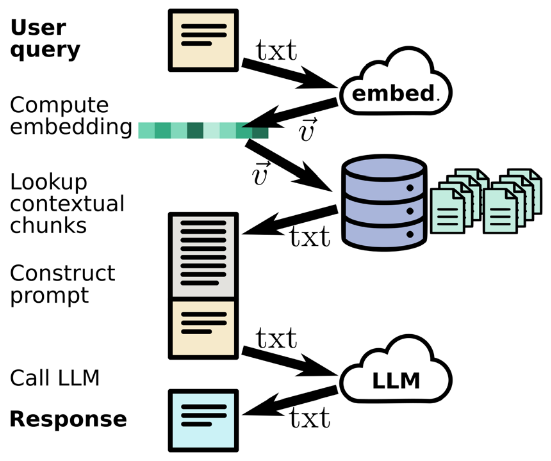

Fig. 1 Workflow for generating domain-tailored chatbot response. The user query is first sent to a ML embedding model, which computes an embedding vector ($\vec{v}$) that captures the semantic content of the input. This vector is used to query a pre-computed database of text chunks. Text snippets that are similar to the query ("close" in the embedding space) are prepended to the user query to construct a prompt. The prompt is sent to a large language model (LLM), which generates a text response for the user.

devices. However, this format entangles content and presentation, making unambiguous extraction of the underlying text quite difficult. While scientific documents of course exist in a more machine-readable format earlier in their development (raw text, latex source, rich text), it is impractical to ask scientists to track down such documents for the myriad of publications relevant to them. Instead, automated conversion of PDF to text is necessary. The simplest and most widely used conversion tools (Adobe Acrobat, Grahl PDF Annotator, IntraPDF, PDFTron, PDF2Text, *etc.*) typically do not correctly handle layout, introducing errors such as breaking text within sentences, or mixing between main text and footnotes. The resultant extraction lacks the required coherence. As a result, a series of efforts have arisen to use more sophisticated methods to provide layout-aware document conversion, including ParsCit,[41] LA-PDFText,[42] CERMINE,[43] OCR++,[44] Grobid,[45] and DeepPDF.[46] We elected to use the Grobid system, which employs ML extraction, provides a clean containerized implementation that acts as server, and converts input PDF files into extensible markup language (XML) outputs that separate the document into meaningful components (title, authors, main text, figures, references, *etc.*).

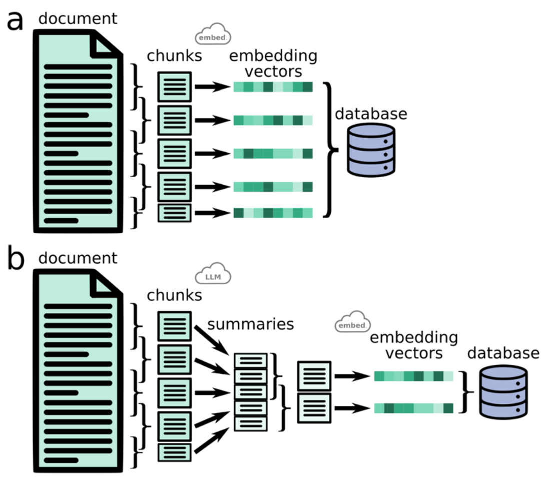

Fig. 2 Workflow for ingesting documents for later lookup. (a) Document text is broken up into a set of overlapping chunks. Each is converted into a vector using a text embedding model. The text chunks and corresponding embedding vectors are stored in a database for later retrieval. (b) The raw text chunks can be compressed using an LLM operating as a summarizer. This shorter summary document can be chunked and stored, along with embedding vector, as previously described. These compressed chunks afford the opportunity to avoid redundant information and maximize the information content of the constructed prompt.

The structured XML versions of the input documents can be easily parsed, chunked, and stored in a database. The publication title and author list is extracted to compose a concise document name, while the main text is broken into a set of overlapping chunks (Fig. 2a). While segmenting the text could be performed in a text-aware manner (*e.g.* by paragraph), breaking at an arbitrary character count is simpler and in fact affords the opportunity for a single chunk containing an extended argument or discussion. The overlapping of chunks guards against the error of mid-sentence truncation, and increases the probability that a sentence relevant to the user query is accompanied by the required contextual information. This overlapping means there is some redundancy between chunks, but this is a small inefficiency. Each chunk is converted to an embedding vector by a lookup in a text embedding model.

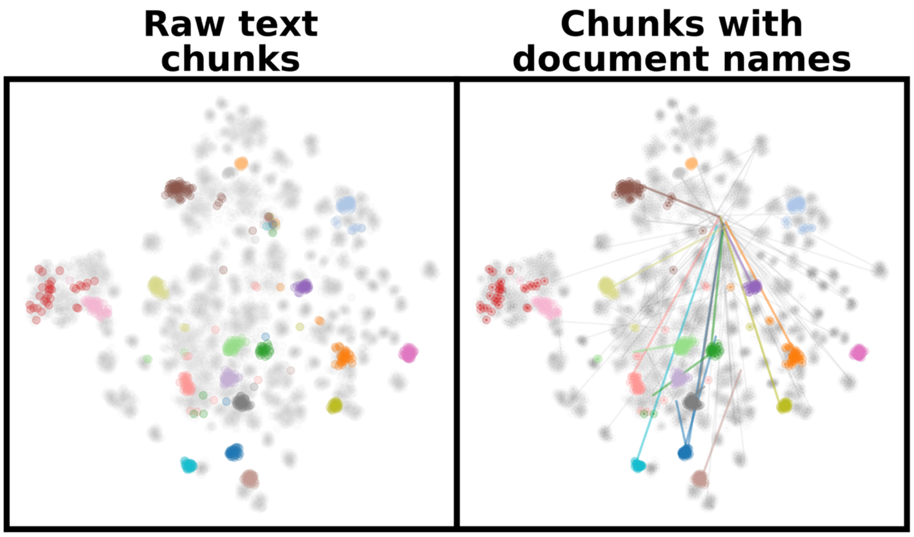

Fig. 3 (left) The text chunks derived from documents can be positioned in a high-dimensional (1536) semantic space. For visualization, this space is projected into two dimensions using the t-SNE method.[47] Each chunk is visualized using a grey dot, but twenty (randomly selected) documents are assigned a particular color. The grouping of chunks from a particular document confirms that the text embedding is succeeding in capturing meaning. A small number of chunks are far from their document cluster. (right) By prepending the document name to each text chunk before computing the embedding, otherwise "orphaned" chunks are grouped with other chunks from that document. The displacement of chunks from their raw position to this improved position is shown using a connecting line. While most chunks are not displaced, orphaned chunks become correctly grouped.

The set of chunks and corresponding vectors are stored in a database for later retrieval.

A valid concern with such a procedure is that many chunks will be "orphaned" in that their content will lack context and be correspondingly meaningless when read alone. Such chunks might contain useful information, but would not be reliably retrieved since their isolated content would not be semantically similar to the user query. A simple improvement to naive chunking is to prepend to each chunk the document name, and use that augmented chunk for embedding calculation and later retrieval. This anchors each chunk to the context provided by the title, and allows the eventual chatbot LLM to identify the source of each provided chunk. In Fig. 3, we visualize the distribution of text chunks in the embedding space. Since the high-dimensional (1536) embedding space cannot be easily understood, we project it into a two-dimensional (2D) space using the t-distributed stochastic neighbor embedding (t-SNE) method,[47] which stochastically redistributes points while maintaining pairwise similarities and thus clustering. What can be observed (Fig. 3, left) is that the different documents are naturally clustered, confirming that the embedding space provides a semantically meaningful organization. However, a small number of document chunks are found to be extremely far from the centroid cluster for their parent document. When recomputing the embeddings with the document name, these orphaned chunks are typically returned to the same neighborhood as the other chunks from that document, confirming that this strategy is helpful in maintaining context for each chunk.

Overall, this chatbot configuration provides a robust means of delivering meaningful answers to user queries. When comparing raw LLM output to the LLM with context chunks (refer to ESI Section 1† for examples), we find a vast improvement in the quality of responses by providing chunks. The raw LLM is prone to fabricating plausible-sounding answers that are nonsense (including adding citations to nonexistent papers), making the responses unsuitable for serious scientific research. However, with access to contextual information, the LLM much more reliably provides true answers (drawn from the provided text) and more helpful analysis. The quality of the chatbot response thereby becomes limited by the quality of the provided context chunks, which is limited by the size of the context window, and the quality of the embedding similarity lookup. In general, we find that embedding lookup is successful in identifying relevant document extracts; although it is not guaranteed to find all relevant chunks, especially for complex queries. Thus, there are clearly opportunities to refine these methods by more carefully selecting and aggregating contextual information into the prompt.

2.2 Prompt engineering

Instead of using the raw document text, other strategies can be pursued. Scientific documents often contain some informational redundancy, *e.g.* due to writing style. For instance, careful and elaborated arguments may be used in documents to educate readers or guide them to a conclusion; whereas only a concise summary of key findings might be relevant for tasks such as giving a chatbot contextual data. Thus, one option in building the chunk database is to create a store of more concise or "compressed" text chunks, which should correspondingly allow more concepts to be placed in the LLM context window. We investigated this possibility by generating a set of summary chunks, by passing each text extract to an LLM with instructions to summarize the chunk (Fig. 2b). In terms of distribution in the embedding space, we find that this summarization step retains semantic meaning (Fig. 4), at least at a coarse level, while greatly reducing the size of the corpus (by a factor of ≈10×).

Although this summarization procedure is an attractive option for increasing lookup speed (smaller corpus size) and

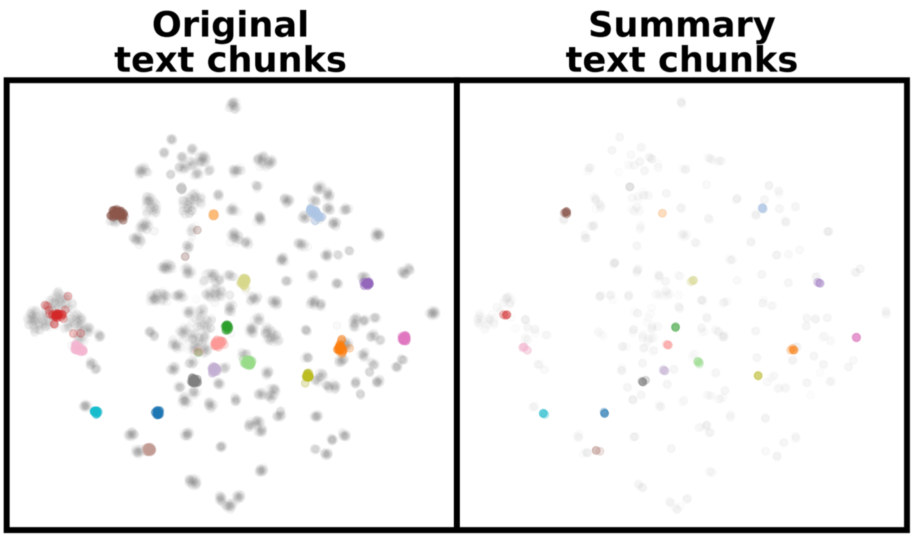

Fig. 4 (left) The 6157 text chunks obtained from 176 documents are clustered in a semantic space (2D t-SNE projection shown, with 20 documents randomly assigned colors). (right) Rather than generating chunks from the raw document text, a set of 707 chunks can be created from LLM summaries of the original documents. This much smaller number of chunks are organized identically in the semantic space, suggesting that the meaning of the original documents is being preserved during summarization.

response quality (more chunks in prompt), we find that in general the chatbot response quality suffers (refer to ESI Section 2† for examples). With access to only summarized information, the chatbot occasionally makes small errors—essentially misinterpreting results or misunderstanding their context—owing to the successive text reinterpretation. In this sense, access to raw text material is preferable. Interestingly, we observe that providing access to both raw and summarized information is useful. While this procedure means the prompt ends up with redundant information, this may serve to reemphasize salient points (phrased in different ways) and guard against mistakes. There is growing evidence that LLMs effectively perform analysis within the output generation itself.[48] Thus, LLM prompts that suggest (for instance) "Let's think step by step" increase response quality by inducing the LLM to build a chain of reasoning in the output. Similarly, providing pre-computed LLM rewording of text chunks affords the system an opportunity to pre-build some textual analysis.

There are ample opportunities for further increasing the performance of domain-specific chatbots by more carefully crafting the chunking and prompt construction steps. For instance, rather than simple summarization, an LLM could be used to preprocess input documents in more sophisticated ways, such as specifically extracting items of interest, or performing contrastive analysis between chunks from different (but related) documents. Or, the embedding of the user query could be used to select among different construction strategies (or even among different LLM systems for the generation step). These and other refinements will no doubt be studied in the near future, thereby refining chatbot performance. However, even the simplest implementation presented here is already able to reliably identify useful documents (and sub-sections thereof) and converse about that content in a useful manner.

2.3 Usage evaluation

It is worth briefly considering an alternate strategy to the embedding lookup described here. In particular, one could imagine inputting the entire text corpus into the LLM, allowing it to select relevant text through the transformer attentional mechanism. A crucial limit to LLMs is the finite context window, into which one must add completion instructions and relevant context data. Typically available LLMs have context windows of 4 k tokens to 100 k tokens (where a token is the atomic unit of LLM parsing; typically a word or word-fragment). Such a context window is insufficient for a corpus of multiple scientific publications (the test corpus used here of 176 documents is >1 M words in length). There is exciting ongoing research into greatly extending the context window of LLMs.[49–51] However, even with larger context windows, there may remain advantages to using embedding strategies to isolate relevant text. Firstly, larger context window sizes increase computational cost and completion time; embedding lookup is typically faster since embeddings are precomputed. Secondly, there are open questions about how the attentional mechanism behaves in extremely large context windows; whereas embedding lookup allows the user to craft retrieval to their particular needs. Nevertheless, it is clear that expanded context windows will yield enormous benefits for domain-specific chatbots, by allowing them to reason about larger and more complex sets of text data; *e.g.* performing more sophisticated comparison between publications, or summarizing an entire domain of research.

ChatBots and LLM systems more generally operate on natural language data, and provide complex linguistic replies that are typically non-deterministic. This makes rigorous evaluation difficult. Nevertheless, many efforts have been formulated to quantify LLM performance,[52–55] and enable ranking of implementations. Rigorous evaluation of science-specific usage is challenging owing to the lack of available community testing datasets. To assess the advantages of the present implementation, the test questions and answers would need to be tailored to the input document corpus, since for questions outside this domain the system would simply revert to the general capabilities of the underlying language model. In our testing, we found that using our optimal embedding strategy and reasonable queries (that a human could answer by looking through the documents), the system usually returns responses that are valid and lack hallucinations ($\approx 90\%$ success, as compared to $<14\%$ success without embedding).

To further evaluate the LLM, we devised other quantitative tests. Language models can be used to sort documents by arbitrary and imprecise criteria. An efficient and scalable strategy is to repeatedly ask the LLM to perform pairwise comparisons, and use this set of comparisons to construct an ordering.[56] The set of comparisons need not be exhaustive, and additional documents can be added to the list with only a small number of additional comparisons (to identify the location in the sorted list for the new item). We tested the ability of an LLM to sort scientific publications by predicted impact; we find that its output roughly correlates to the impact factor of the journal the work was published in, implying that the LLM is capturing some aspects that humans use to predict impact (Fig. S1 and S2 ESI†).

As another test, the LLM was tasked with assigning the scientific documents into a set of human-selected categories. This classification task can be compared to human selections for the same task, in order to quantify performance (Table 1). This is an inherently imprecise task, especially given the overlap in the selected categories. Nevertheless, the LLM is highly successful at this challenging task (accuracy 81–99%), with the majority of errors being reasonable (*e.g.* ambiguous classification between materials category, or self-assembly category more specifically). The strong performance across a diverse set of tasks, as presented here, helps to support the argument that a chatbot with access to domain-specific documents can assist researchers in a variety of meaningful tasks.

2.4 Image data

A key aspect of research, and scientific publications, is the data visualization used to reason about trends and describe results to others. Thus, it is important to consider how to aid researchers in image-based search and retrieval. Image

Table 1 Evaluation of the ability of LLM (OpenAI GPT 3.5) to classify scientific documents. Each document was manually classified into one of 6 thematic categories. The central 6 $\times$ 6 cells show the distribution of LLM classifications. The rightmost columns provide prediction metrics, including precision (Pr), recall (Re), and accuracy (Ac). Overall, the LLM is successful at this imprecise task

	LLM assignment						Metrics		
Ground truth	Self-assembly	Materials	Scattering	Machine-learning	Photo-responsive	Other	Pr	Re	Ac
Self-assembly	**95%**	3%	0%	0%	0%	2%	79%	95%	89%
Materials	28%	**53%**	17%	2%	0%	0%	84%	53%	81%
Scattering	0%	0%	**100%**	0%	0%	0%	41%	100%	91%
Machine-learning	0%	4%	22%	**70%**	0%	4%	94%	70%	95%
Photo-responsive	0%	9%	0%	0%	**91%**	0%	100%	91%	99%
Other	0%	40%	20%	0%	0%	**40%**	50%	40%	97%

embedding methods can generate a vector describing an image's semantic content, similar to text embedding, allowing retrieval of related images or other image–image reasoning operations. These capabilities can be exploited to help researchers perform search and retrieval related to the figures contained in scientific documents. To demonstrate the utility of image embedding for scientific documents, we added a simple image similarity search component to our system. The publication figures identified by Grobid were extracted from the input PDF documents, and converted into an image embedding vector using the contrastive language-image pre-training (CLIP)[64] method. Because CLIP is a multi-modal embedding trained on images and text, it acquires visual understanding embodying meaningful semantics. This affords the opportunity for similarity measures based on human concepts. This approach enables the user to input an image, and find publication figures with semantic similarity (Fig. 5). Importantly, these search and retrieval operations are performed on the user-provided corpus of scientific documents, allowing tasks to be much more domain-specific than when using generic (*e.g.* web-based) image search systems. In addition to exploring publication figures, one can compute image embeddings for a set of unlabelled experimental data (such as electron micrographs, scanning probe images, or scattering/diffraction detector images) and thereby perform similarity lookup on input images (Fig. 6, S3 and S4 ESI†). This method is extremely successful in rapidly identifying relevant images, capturing aspects of similarity well beyond simple pixelwise or sub-structural match. Moreover, the user can select among similarity measures to achieve different kinds of retrieval. For instance, Euclidian distance in the embedding space assesses how similar images are, while the dot product between embedding vectors measures a looser kind of overlap between the underlying concepts. This retrieval can be viewed as a form of zero-shot ML, in the sense that the CLIP model was not trained on scientific images explicitly, and yet it can provide a meaningful descriptor of these images. In other words, the semantic understanding of CLIP is sufficiently broad and robust that it generalizes to the kinds of images used in scientific contexts. This suggests that these existing models can be immediately deployed on scientific instruments to assist in organizing and classifying images, and can be valuable for searching through publications to find relevant data (*e.g.* to find examples of a particular kind of measurement result).

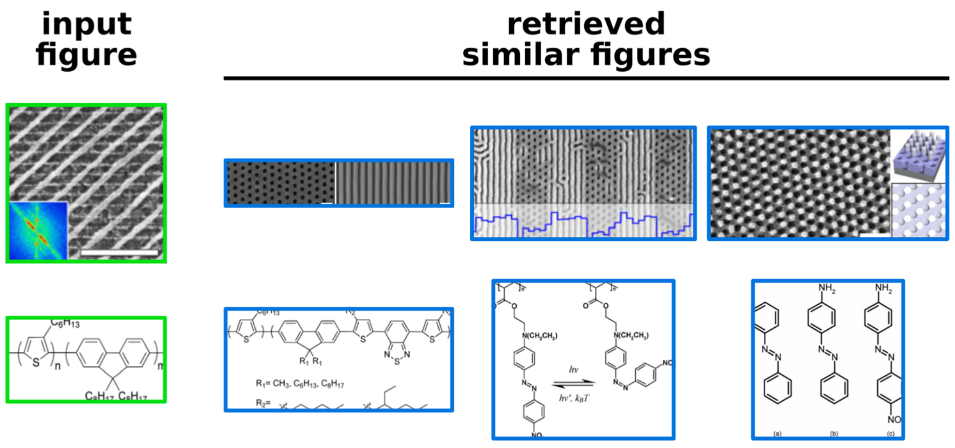

Fig. 5 Image embeddings can be used to identify semantically related figures (or portions thereof) from a datastore of pre-processed documents. An electron micrograph of a mesh nanostructure (Fig. 3d from ref. 57) yields suggestions of other micrographs of nanostructure arrays (Fig. 3h and i from ref. 58, Fig. 5b from ref. 58, Fig. 3e from ref. 59). The chemical structure of a polymer (Scheme 1 from ref. 60) yields chemical structures from other publications in the database (Fig. 1 from ref. 61, Fig. 1 from ref. 62, Fig. 1 from ref. 63). Cosine similarity was used to identify relevant images.

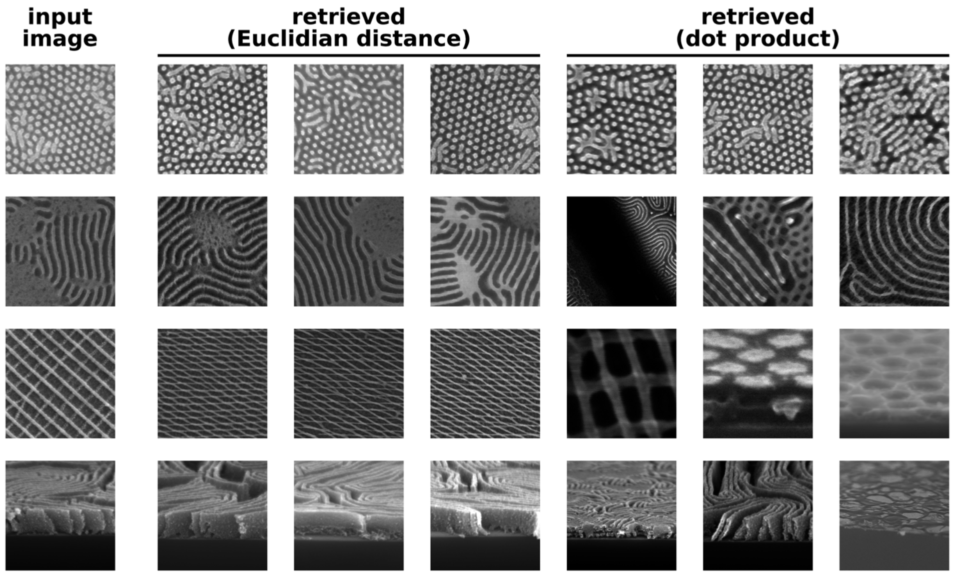

Fig. 6 Image embedding and retrieval can be applied to arbitrary image data. Scanning electron microscope (SEM) micrographs for samples were used as image inputs, to search for semantically similar images in a pre-computed dataset of 20 302 other SEM images. The retrieval is rapid and meaningful, with relevant figures being retrieved. Crops of the SEMs are shown for clarity; the embeddings were computing on the full SEM image. Examples are shown for retrieval using Euclidian distance (which measures similarity in assessed meaning), as well as dot product similarity (measures overlap in concepts).

3. Perspectives

We have demonstrated that existing technologies are already suitable for use by physical scientists to speed up their work in identifying useful publications (and sub-sections or figures therein), and to provide rapid and meaningful answers to scientific questions. By using a domain-specific chatbot, with access to relevant documents, scientists should be able to accelerate several aspects of their workflow. For instance, such a digital assistant should be useful during literature searching, proposal writing, manuscript drafting, hypothesis testing, and ideation.[65–68]

The presented implementation is simple and quite limited; it could be improved in several ways. We have used a traditional relational database to store text and embeddings. For the modest corpus sizes considered here, lookup time is not performance limiting. Yet a more scaleable solution would be to use a special-purpose vector databases (such as Pinecone,[69] Milvus,[70] or Chroma[71]). As previously discussed, there are opportunities to further refine the presented prompt construction strategy. Herein, we have used a cloud LLM; an alternative would be to deploy an LLM locally to avoid the latency and privacy concerns associated with cloud lookup.[72,73] More targeted chatbot behavior could also be obtained by fine-tuning a local LLM by retraining on the relevant corpus of documents. Note that even with fine-tuning the LLM itself, there remains an advantage in providing text extracts in the constructed prompt. Namely, it provides relevant information to assist LLM reasoning, and enables direct quoting and citation. It is also obvious that as LLMs increase in sophistication (as previously discussed), the corresponding domain-specific chatbots will correspondingly increase in sophistication. This suggests that domain-specific endeavors should in fact be designed in a way that is decoupled from any specific LLM implementation. This will allow them to take advantage of future improvements in LLMs by simply changing the system being accessed.

An exciting possibility afforded by LLMs is literature-based discovery (LBD),[74] which seeks to make discoveries by mining the literature and proposing/testing hypotheses. For instance, trends or commonalities can be automatically extracted by LLM scanning the literature, generating lists of conclusions, and aggregating the results. A chatbot can acceleration human ideation, by providing immediate feedback on hypotheses, and retrieving relevant documents from the literature. We expect to see increasingly sophisticated literature discovery paradigms emerge as domain-specific chatbots are deployed more broadly.

An emerging trend in experimental sciences is autonomous experimentation (AE), wherein the measurement loop is closed using a decision-making algorithm that selects high-quality experiments to perform.[75–77] For instance, researchers have demonstrated that a synchrotron X-ray scattering beamline can autonomously explore physical parameter spaces,[78,79] reconstructing a high-quality model of the space and even discovering new materials or structures.[80] Existing approaches have typically used grounded ML modeling approaches (such as Gaussian process regression). It is interesting to consider whether the more flexible and general-purpose understanding of LLMs can be directly leverage as a decision-making agent in experimental loops. There is early evidence that LLMs can indeed engage in autonomous scientific discovery,[35] and further elaboration of these methods is an exciting avenue for future study.

4. Methods

4.1 Database preparation

For testing, we assembled a dataset of 176 PDF files (the author's full set of peer-reviewed scientific journal publications and book chapters), with a cumulative file size of 703 MB. We use the open-source Grobid[45] system to convert these PDF files into XML files. The XML files were parsed using the Python library BeautifulSoup. For further analysis, only the main text was considered, eliminating input PDF boilerplate and references sections. In total, the text corpus is 1 061 967 words ($\approx$3500 pages of textual data). Text chunks were generated by breaking the input document main text into segments 1400 characters in length, with an overlap of 280 characters between subsequent chunks. The overlap accounts for the random truncation of sentences, and increases the probability of a given block of text being found in a chunk along with relevant contextual information. An embedding vector was computed for each text chunk using the OpenAI cloud API, and the text-embedding-ada-002 model, which returns a 1536 length vector. The text chunks and vectors were stored in a MySQL database. For retrieval efficiency, the list of embedding vectors was cached in a binary file using the numpy Python library.[81]

Summaries of raw text chunks were obtained by calling the gpt-3.5-turbo model (OpenAI) with a prompt that included instructions to "summarize in a concise way." These summaries were concatenated into a summary document, which was in turn chunked. Embeddings for each chunk were computed as before. Thus, the chunk summaries are smaller in number than the raw text chunks, representing a substantial compression of the original text (707/6157 $\approx$ 11%).

4.2 Visualization

Visualization of the semantic organization of document chunks was performed using the t-distributed stochastic neighbor embedding (t-SNE) method.[47,82] This computes a statistical non-linear mapping of points from a high-dimensional space into a lower-dimensional space, attempting to maintain pairwise similarity. We project from the 1536 dimensional text embedding space defined by the embedding model into a two-dimensional (2D) space, using a perplexity of 40 and 10 000 iterations. Images were plotted using the matplotlib[83] package.

4.3 Chatbot querying

The results discussed primarily used the gpt-3.5-turbo-0301, accessed *via* the OpenAI cloud API using Python code. We assume an overall context window of 16 384 characters (based on a model limit of 4096 tokens). Prompts were constructed by providing an instruction to answer user queries using provided text, followed by a sequence of text extracts, followed by the user query. Relevant chunks were identified based on cosine similarity (eqn (1)). By assessing the relative angle between vectors, this measure assesses thematic similarity. Since the selected embedding is normalized, this is equivalent to Euclidian distance. As many chunks as possible were added to the context window, while reserving 3564 characters ($\approx$900 tokens) for chatbot response. A single query executes in $\approx$10 s, with the chunk lookup and prompt construction requiring <1 s, and the majority of execution time resulting from the cloud LLM lookup.

For comparisons of response quality (refer to ESI†), we also tested the ChatGPT4 model, using the web interface. In this case, constructed prompts were manually copied into the web interface (using a new conversation thread with no history) to generate a response. The ChatGPT4 system uses the more powerful and knowledgeable GPT4 model, and also leverages the fine-tuning performed by OpenAI in developing the ChatGPT versions of their system. This system emphasizes useful responses, while minimizing fabrications.

The models used herein have a temperature parameter that can be used to influence model output (refer to ESI† for examples). Low values of this parameter have limited variability and induce more deterministic output. Higher values of this parameter lead to more variable output, with sufficiently large values leading to outputs corrupted by irrelevant text completions. The default value (1.0) was used for the presented results (except where noted otherwise), and was found to yield reasonable responses for the tasks explored. Repeatability tests confirm that at this setting, the model output varies in exact wording, but retains the same general semantic meaning.

4.4 Evaluation

Input questions were manually crafted and selected in an attempt to cover a distribution of use-cases relevant to the training corpus. Questions were constructed such that a science-trained human would be able to answer them if they were familiar with the input documents and given time to look through the documents, but without spending time performing extensive background research or thinking. Model outputs were manually scored to identify components that were incorrect (or fabricated) *versus* correct. Answers were judged overall correct when they provided valid information without introducing erroneous ideas. Based on this manual assessment, it was found that the embedding strategy can respond successfully to $\approx$90% of queries, which can be compared to a <14% success rate when embedding is not performed.

In order to sort documents by scientific impact, a set of pairwise comparisons were generated, where for each comparison the LLM is asked to select which publication is higher impact. The LLM was provided with each document's text—including title, abstract, and initial portion of main text (up to the context limit of the model)—but not provided with ancillary information such as journal name. Comparison pairs were selected randomly, biased so that every document is involved in at least one pairwise comparison. From this set of 818 comparisons (out of a total possible $176^2 = 30{,}976$), a ranked list of documents was generated through a straightforward sorting procedure; namely, iteratively considering pairs of documents, and swapping their order if the swap reduces (or does not change) the total number of misordered pairs (*i.e.* pairs where a higher-impact paper is incorrectly sorted lower in the list). This procedure does not resolve to zero misordered pairs, since the LLM pairwise comparisons are not guaranteed to form a perfectly consistent set. Viewed as a directed graph, we indeed identify cycles. Nevertheless, the ordering is found to be meaningful, as it roughly correlates to the impact factor of the journal the work was published in, implying that the LLM is capturing some aspects that humans use to predict impact (Fig. S1 and S2 ESI†).

The LLM was evaluated on a classification task, where each document was manually assigned to one of 6 thematic categories. The LLM was then asked to classify each document into one of those categories. The task is inherently ambiguous, since some categories are subsets of others (e.g. self-assembly and photo-responsive papers are special cases of the more general materials category), while other publications touch on multiple topics (*e.g.* some papers involve applying machine-learning to scattering datasets). Despite this challenge, the LLM identifies the same category as the human in the majority of cases (accuracy 81–99%).

4.5 Image querying

Figures from publications were identified from the Grobid XML files, and extracted from the PDF documents into images using the PyMuPDF library. References to figures, along with captions, were stored in the MySQL database. Raw images were similarly added to the database, without caption information. Image embeddings were computed using the contrastive language-image pre-training (CLIP)[64] method, specifically the ViT-B/32 pre-trained model provided by the PyTorch[84] deep learning environment (vectors are length 512). Bulk calculation of embeddings requires $\approx$66 ms per image. The list of embeddings were stored in the MySQL database. Image similarity was computed using multiple measures: Euclidian distance in the CLIP space, which measures the distance between the meaning of the images; cosine similarity of the embedding vectors, which measures the thematic similarity; and the raw (non-normalized) dot product, which measures a form of projected relatedness. Since CLIP embeddings are not normalized, the cosine similarity and Euclidian distances are not equivalent; nevertheless in practice they are found to return highly similar results, since the image corpus is relatively clustered in the overall CLIP space.

Data availability

Source code for chatbot and associated tools is available at **https://github.com/CFN-softbio/SciBot**.

Author contributions

KGY developed the concepts, authored the software, conducted the research, and wrote the manuscript. A LLM chatbot (ChatGPT4) was used as a digital assistant to expedite code development.

Conflicts of interest

There are no conflicts to declare.

Acknowledgements

This research was carried out by the Center for Functional Nanomaterials, which is a U.S. DOE Office of Science Facility, at Brookhaven National Laboratory under Contract No. DE-SC0012704. We thank Dr Kevin F. Casey for fruitful discussion. We acknowledge the important contributions of the open source software community.

Notes and references

1 J. Qiu, Q. Wu, G. Ding, Y. Xu and S. Feng, *EURASIP J. Adv. Signal Process.*, 2016, **2016**, 67.
2 M. Jovanovic and M. Campbell, *Computer*, 2022, **55**, 107–112.
3 R. Gozalo-Brizuela and E. C. Garrido-Merchan, ChatGPT is Not All You Need. A State of the Art Review of large Generative AI Models, *arxiv*, 2023, Preprint, arXiv:2301.04655, DOI: **10.48550/arXiv.2301.04655**.
4 A. Ramesh, P. Dhariwal, A. Nichol, C. Chu and M. Chen, Hierarchical Text-Conditional Image Generation With CLIP Latents, *arxiv*, 2022, Preprint, arXiv:2204.06125, DOI: **10.48550/arXiv.2204.06125**.
5 R. Rombach, A. Blattmann, D. Lorenz, P. Esser and B. Ommer, High-Resolution Image Synthesis With Latent Diffusion Models, *arxiv*, 2021, Preprint, arXiv:2112.10752, DOI: **10.48550/arXiv.2112.10752**.
6 J. Oppenlaender, *Proceedings of the 25th International Academic Mindtrek Conference*, New York, NY, USA, 2022, pp. 192–202.
7 T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever and D. Amodei, *Advances in Neural Information Processing Systems*, 2020, pp. 1877–1901.
8 A. Radford, K. Narasimhan, T. Salimans and I. Sutskever, *Improving Language Understanding by Generative Pre-Training, OpenAI Technical Report*, 2018.
9 A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser and I. Polosukhin, Attention is All You Need, *arxiv*, 2017, Preprint, arXiv:1706.03762, DOI: **10.48550/arXiv.1706.03762**.
10 R. Bommasani, D. A. Hudson, E. Adeli, R. Altman, S. Arora, S. von Arx, M. S. Bernstein, J. Bohg, A. Bosselut, E. Brunskill, E. Brynjolfsson, S. Buch, D. Card, R. Castellon, N. Chatterji, A. Chen, K. Creel, J. Q. Davis, D. Demszky, C. Donahue, M. Doumbouya, E. Durmus, S. Ermon, J. Etchemendy, K. Ethayarajh, L. Fei-Fei, C. Finn, T. Gale, L. Gillespie, K. Goel, N. Goodman, S. Grossman, N. Guha, T. Hashimoto, P. Henderson, J. Hewitt, D. E. Ho, J. Hong, K. Hsu, J. Huang, T. Icard, S. Jain, D. Jurafsky, P. Kalluri, S. Karamcheti, G. Keeling, F. Khani, O. Khattab, P. W. Koh, M. Krass, R. Krishna, R. Kuditipudi, A. Kumar, F. Ladhak, M. Lee, T. Lee, J. Leskovec, I. Levent, X. L. Li, X. Li, T. Ma, A. Malik, C. D. Manning, S. Mirchandani,

E. Mitchell, Z. Munyikwa, S. Nair, A. Narayan, D. Narayanan, B. Newman, A. Nie, J. C. Niebles, H. Nilforoshan, J. Nyarko, G. Ogut, L. Orr, I. Papadimitriou, J. S. Park, C. Piech, E. Portelance, C. Potts, A. Raghunathan, R. Reich, H. Ren, F. Rong, Y. Roohani, C. Ruiz, J. Ryan, C. Ré, D. Sadigh, S. Sagawa, K. Santhanam, A. Shih, K. Srinivasan, A. Tamkin, R. Taori, A. W. Thomas, F. Tramèr, R. E. Wang, W. Wang, B. Wu, J. Wu, Y. Wu, S. M. Xie, M. Yasunaga, J. You, M. Zaharia, M. Zhang, T. Zhang, X. Zhang, Y. Zhang, L. Zheng, K. Zhou and P. Liang, On the Opportunities and Risks of Foundation Models, *arxiv*, 2021, Preprint, arXiv:2108.07258, DOI: **10.48550/arXiv.2108.07258**.

11 K. Li, A. K. Hopkins, D. Bau, F. Viégas, H. Pfister and M. Wattenberg, Emergent World Representations: Exploring a Sequence Model Trained on a Synthetic Task, *arxiv*, 2023, Preprint, arXiv:2210.13382, DOI: **10.48550/arXiv.2210.13382**.

12 E. Akyürek, D. Schuurmans, J. Andreas, T. Ma and D. Zhou, What Learning Algorithm is in-Context Learning? Investigations with Linear Models, *arxiv*, 2023, Preprint, arXiv:2211.15661, DOI: **10.48550/arXiv.2211.15661**.

13 M. Kosinski, Theory of Mind May Have Spontaneously Emerged in Large Language Models, *arxiv*, 2023, Preprint, arXiv:2302.02083, DOI: **10.48550/arXiv.2302.02083**.

14 D. Ganguli, D. Hernandez, L. Lovitt, A. Askell, Y. Bai, A. Chen, T. Conerly, N. Dassarma, D. Drain, N. Elhage, S. E. Showk, S. Fort, Z. Hatfield-Dodds, T. Henighan, S. Johnston, A. Jones, N. Joseph, J. Kernian, S. Kravec, B. Mann, N. Nanda, K. Ndousse, C. Olsson, D. Amodei, T. Brown, J. Kaplan, S. McCandlish, C. Olah, D. Amodei and J. Clark, *2022 ACM Conference on Fairness, Accountability, and Transparency*, 2022.

15 J. Wei, Y. Tay, R. Bommasani, C. Raffel, B. Zoph, S. Borgeaud, D. Yogatama, M. Bosma, D. Zhou, D. Metzler, E. H. Chi, T. Hashimoto, O. Vinyals, P. Liang, J. Dean and W. Fedus, Emergent abilities of large language models, *arxiv*, 2022, Preprint, arXiv:2206.07682, DOI: **10.48550/arXiv.2206.07682**.

16 N. Nanda, L. Chan, T. Lieberum, J. Smith and J. Steinhardt, Progress Measures for Grokking *via* mechanistic Interpretability, *arxiv*, 2023, Preprint, arXiv:2301.05217, DOI: **10.48550/arXiv.2301.05217**.

17 S. Bubeck, V. Chandrasekaran, R. Eldan, J. Gehrke, E. Horvitz, E. Kamar, P. Lee, Y. T. Lee, Y. Li, S. Lundberg, H. Nori, H. Palangi, M. T. Ribeiro and Y. Zhang, Sparks of Artificial General Intelligence: Early Experiments with GPT-4, *arxiv*, 2023, Preprint, arXiv:2303.12712, DOI: **10.48550/arXiv.2303.12712**.

18 T. Webb, K. J. Holyoak and H. Lu, *Nat. Hum. Behav.*, 2023, (7), 1526–1541, DOI: **10.1038/s41562-023-01659-w**.

19 D. M. Ziegler, N. Stiennon, J. Wu, T. B. Brown, A. Radford, D. Amodei, P. Christiano and G. Irving, Fine-Tuning Language Models from Human Preferences, *arxiv*, 2020, Preprint, arXiv:1909.08593, DOI: **10.48550/arXiv.1909.08593**.

20 N. Shinn, F. Cassano, B. Labash, A. Gopinath, K. Narasimhan and S. Yao, Reflexion: Language Agents with Verbal Reinforcement Learning, *arxiv*, 2023, Preprint, arXiv:2303.11366, DOI: **10.48550/arXiv.2303.11366**.

21 H. Lightman, V. Kosaraju, Y. Burda, H. Edwards, B. Baker, T. Lee, J. Leike, J. Schulman, I. Sutskever and K. Cobbe, Let's Verify Step by Step, *arxiv*, 2023, Preprint, arXiv:2305.20050, DOI: **10.48550/arXiv.2305.20050**.

22 W. Xu, A. Banburski-Fahey and N. Jojic, Reprompting: Automated Chain-of-Thought Prompt Inference Through Gibbs Sampling, *arxiv*, 2023, Preprint, arXiv:2305.09993, DOI: **10.48550/arXiv.2305.09993**.

23 S. Yao, D. Yu, J. Zhao, I. Shafran, T. L. Griffiths, Y. Cao and K. Narasimhan, Tree of Thoughts: Deliberate Problem Solving with Large Language Models, *arxiv*, 2023, Preprint, arXiv:2305.10601, DOI: **10.48550/arXiv.2305.10601**.

24 S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan and Y. Cao, ReAct: synergizing Reasoning and Acting in Language Models, *arxiv*, 2023, Preprint, arXiv:2210.03629, DOI: **10.48550/arXiv.2210.03629**.

25 T. Schick, J. Dwivedi-Yu, R. Dessì, R. Raileanu, M. Lomeli, L. Zettlemoyer, N. Cancedda and T. Scialom, Toolformer: Language Models Can Teach Themselves to Use Tools, *arxiv*, 2023, Preprint, arXiv:2302.04761, DOI: **10.48550/arXiv.2302.04761**.

26 L. Gao, A. Madaan, S. Zhou, U. Alon, P. Liu, Y. Yang, J. Callan and G. Neubig, Program-Aided Language Models, *arxiv*, 2023, Preprint, arXiv:2211.10435, DOI: **10.48550/arXiv.2211.10435**.

27 Y. Liang, C. Wu, T. Song, W. Wu, Y. Xia, Y. Liu, Y. Ou, S. Lu, L. Ji, S. Mao, Y. Wang, L. Shou, M. Gong and N. Duan, TaskMatrixAI: Completing Tasks by Connecting Foundation Models With Millions of APIs, *arxiv*, 2023, Preprint, arXiv:2303.16434, DOI: **10.48550/arXiv.2303.16434**.

28 Y. Shen, K. Song, X. Tan, D. Li, W. Lu and Y. Zhuang: Solving AI Tasks with ChatGPT and Its Friends in Hugging Face, *arxiv*, 2023, Preprint, arXiv:2303.17580, DOI: **10.48550/arXiv.2303.17580**.

29 T. Cai, X. Wang, T. Ma, X. Chen and D. Zhou, Large Language Models as Tool Makers, *arxiv*, 2023, Preprint, arXiv:2305.17126, DOI: **10.48550/arXiv.2305.17126**.

30 B. Peng, M. Galley, P. He, H. Cheng, Y. Xie, Y. Hu, Q. Huang, L. Liden, Z. Yu, W. Chen and J. Gao, Check Your Facts and Try Again: Improving Large Language Models with External Knowledge and Automated Feedback, *arxiv*, 2023, Preprint, arXiv:2302.12813, DOI: **10.48550/arXiv.2302.12813**.

31 B. Xu, Z. Peng, B. Lei, S. Mukherjee, Y. Liu and D. Xu, ReWOO: Decoupling Reasoning from Observations for Efficient Augmented Language Models, *arxiv*, 2023, Preprint, arXiv:2305.18323, DOI: **10.48550/arXiv.2305.18323**.

32 C.-Y. Hsieh, S.-A. Chen, C.-L. Li, Y. Fujii, A. Ratner, C.-Y. Lee, R. Krishna and T. Pfister, Tool Documentation Enables Zero-Shot Tool-Usage with Large Language Models, *arxiv*, 2023, Preprint, arXiv:2308.00675, DOI: **10.48550/arXiv.2308.00675**.

33 G. Wang, Y. Xie, Y. Jiang, A. Mandlekar, C. Xiao, Y. Zhu, L. Fan and A. Anandkumar: An Open-Ended Embodied Agent With Large Language Models, *arxiv*, 2023, Preprint, arXiv:2305.16291, DOI: **10.48550/arXiv.2305.16291**.

34 G. Li, H. A. A. K. Hammoud, H. Itani, D. Khizbullin and B. Ghanem: Communicative Agents for “Mind” Exploration of Large Scale Language Model Society, *arxiv*, 2023, Preprint, arXiv:2303.17760, DOI: **10.48550/arXiv.2303.17760**.
35 D. A. Boiko, R. MacKnight and G. Gomes, Emergent autonomous scientific research capabilities of large language models, *arxiv*, 2023, Preprint, arXiv:2304.05332, DOI: **10.48550/arXiv.2304.05332**.
36 R. Taylor, M. Kardas, G. Cucurull, T. Scialom, A. Hartshorn, E. Saravia, A. Poulton, V. Kerkez and R. Stojnic, Galactica: A Large Language Model for Science, *arxiv*, 2022, Preprint, arXiv:2211.09085, DOI: **10.48550/arXiv.2211.09085**.
37 E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang and W. Chen: Low-Rank Adaptation of Large Language Models, *arxiv*, 2021, Preprint, arXiv:2106.09685, DOI: **10.48550/arXiv.2106.09685**.
38 T. Dettmers, A. Pagnoni, A. Holtzman and L. Zettlemoyer, QLoRA: Efficient Finetuning of Quantized LLMs, *arxiv*, 2023, Preprint, arXiv:2305.14314, DOI: **10.48550/arXiv.2305.14314**.
39 H. Djirdeh, *Customizing an OpenAI Chatbot With Embeddings*, Medium, 2023, **https://blog.bitsrc.io/customizing-an-openai-chatbot-with-embeddings-fdc9ec859bbb**, accessed 03 March 2023.
40 OpenAI, *Question Answering using Embeddings-Based Search*, Github, 2023, **https://github.com/openai/openai-cookbook/blob/main/examples/Question_answering_using_embeddings.ipynb**, accessed 2023-05-08.
41 I. Councill, C. L. Giles and M.-Y. Kan, *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco, 2008.
42 C. Ramakrishnan, A. Patnia, E. Hovy and G. A. Burns, *Source Code Biol. Med.*, 2012, **7**, 7.
43 D. Tkaczyk, P. Szostek, M. Fedoryszak, P. J. Dendek and Ł. Bolikowski, *Int. J. Document Anal. Recognit.*, 2015, **18**, 317–335.
44 M. Singh, B. Barua, P. Palod, M. Garg, S. Satapathy, S. Bushi, K. Ayush, K. Sai Rohith, T. Gamidi, P. Goyal and A. Mukherjee, *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, Osaka, Japan, 2016, pp. 3390–3400.
45 *GROBID*, 2008–2023, **https://github.com/kermitt2/grobid**.
46 C. G. Stahl, S. R. Young, D. Herrmannova, R. M. Patton and J. C. Wells, DeepPDF: A Deep Learning Approach to Extracting Text from PDFs, *7th International Workshop on Mining Scientific Publications*, 2018, **https://www.osti.gov/biblio/1460210**.
47 L. van der Maaten and G. Hinton, *J. Mach. Learn. Res.*, 2008, **9**, 2579–2605.
48 T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, Y. Iwasawa, Large Language Models are Zero-Shot Reasoners, *arxiv*, 2023, Preprint, arXiv:2205.11916, DOI: **10.48550/arXiv.2205.11916**.
49 I. Beltagy, M. E. Peters and A. Cohan, Longformer: The Long-Document Transformer, *arxiv*, 2020, Preprint, arXiv:2004.05150, DOI: **10.48550/arXiv.2004.05150**.
50 J. Ding, S. Ma, L. Dong, X. Zhang, S. Huang, W. Wang, N. Zheng and F. Wei, LongNet: Scaling Transformers to 1,000,000,000 tokens, *arxiv*, 2023, Preprint, arXiv:2307.02486, DOI: **10.48550/arXiv.2307.02486**.
51 S. Tworkowski, K. Staniszewski, M. Pacek, Y. Wu, H. Michalewski and P. Miłoś, Focused Transformer: Contrastive Training for Context Scaling, *arxiv*, 2023, Preprint, arXiv:2307.03170, DOI: **10.48550/arXiv.2307.03170**.
52 P. Clark, I. Cowhey, O. Etzioni, T. Khot, A. Sabharwal, C. Schoenick and O. Tafjord, Think You Have Solved Question Answering? Try ARC, the AI2 Reasoning Challenge, *arxiv*, 2018, Preprint, arXiv:1803.05457, DOI: **10.48550/arXiv.1803.05457**.
53 R. Zellers, A. Holtzman, Y. Bisk, A. Farhadi and Y. Choi, HellaSwag: Can A Machine Really Finish Your Sentence?, *arxiv*, 2019, Preprint, arXiv:1905.07830, DOI: **10.48550/arXiv.1905.07830**.
54 D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. Song and J. Steinhardt, Measuring massive multitask language understanding, *arxiv*, 2021, Preprint, arXiv:2009.03300, DOI: **10.48550/arXiv.2009.03300**.
55 S. Lin, J. Hilton and O. Evans, TruthfulQA: measuring how models mimic human falsehoods, *arxiv*, 2022, Preprint, arXiv:2109.07958, DOI: **10.48550/arXiv.2109.07958**.
56 Z. Qin, R. Jagerman, K. Hui, H. Zhuang, J. Wu, J. Shen, T. Liu, J. Liu, D. Metzler, X. Wang and M. Bendersky, Large Language Models are Effective Text Rankers With Pairwise Ranking Prompting, *arxiv*, 2023, Preprint, arXiv:2306.17563, DOI: **10.48550/arXiv.2306.17563**.
57 P. W. Majewski, A. Rahman, C. T. Black and K. G. Yager, *Nat. Commun.*, 2015, **6**, 7448.
58 A. Stein, G. Wright, K. G. Yager, G. S. Doerk and C. T. Black, *Nat. Commun.*, 2016, **7**, 12366.
59 A. Rahman, P. W. Majewski, G. Doerk, C. T. Black and K. G. Yager, *Nat. Commun.*, 2016, **7**, 13988.
60 Y.-H. Lin, K. G. Yager, B. Stewart and R. Verduzco, *Soft Matter*, 2014, **10**, 3817–3825.
61 K. A. Smith, Y.-H. Lin, J. W. Mok, K. G. Yager, J. Strzalka, W. Nie, A. D. Mohite and R. Verduzco, *Macromolecules*, 2015, **48**, 8346–8353.
62 K. G. Yager, O. M. Tanchak, C. Godbout, H. Fritzsche and C. J. Barrett, *Macromolecules*, 2006, **39**, 9311–9319.
63 K. G. Yager and C. J. Barrett, *J. Photochem. Photobiol., A*, 2006, **182**, 250–261.
64 A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger and I. Sutskever, Learning Transferable Visual Models From Natural Language Supervision, *arxiv*, 2021, Preprint, arXiv:2103.00020, DOI: **10.48550/arXiv.2103.00020**.
65 J. Haase and P. H. P. Hanel, Artificial Muses: Generative Artificial Intelligence Chatbots Have Risen to Human-Level Creativity, *arxiv*, 2023, Preprint, arXiv:2303.12003, DOI: **10.48550/arXiv.2303.12003**.
66 K. Girotra, L. Meincke, C. Terwiesch and K. T. Ulrich, *Ideas are Dimes a Dozen: Large Language Models for Idea Generation in Innovation*, SSRN, 2023.

67 L. Boussioux, J. N. Lane, M. Zhang, V. Jacimovic and K. R. Lakhani, *Harvard Business School Technology & Operations Mgt. Unit Working Paper*, 2023.

68 A. R. Doshi and O. Hauser, *Generative Artificial Intelligence Enhances Creativity*, SSRN, 2023.

69 Pinecone, *Vector Database for Vector Search*, 2023, **https://www.pinecone.io/**, accessed 2023-06-09.

70 T. M. Project, *Milvus*, Github, 2023, **https://github.com/milvus-io/milvus**, accessed 2023-06-09.

71 Chroma, *Chroma*, Github, 2023, **https://github.com/chroma-core/chroma**, accessed 2023-06-09.

72 W.-L. Chiang, Z. Li, Z. Lin, Y. Sheng, Z. Wu, H. Zhang, L. Zheng, S. Zhuang, Y. Zhuang, J. E. Gonzalez, I. Stoica and E. P. Xing, *Vicuna: An Open-Source Chatbot Impressing GPT-4 with 90%* ChatGPT Quality*, 2023, **https://lmsys.org/blog/2023-03-30-vicuna/**.

73 localGPT, *localGPT*, Github, 2023, **https://github.com/PromtEngineer/localGPT**, accessed 2023-06-09.

74 Q. Wang, D. Downey, H. Ji and T. Hope, Learning to Generate Novel Scientific Directions with Contextualized Literature-based Discovery, *arXiv*, 2023, preprint, arXiv:2305.14259, DOI: **10.48550/arXiv.2305.14259**.

75 H. S. Stein and J. M. Gregoire, *Chem. Sci.*, 2019, **10**, 9640–9649.

76 E. Stach, B. DeCost, A. G. Kusne, J. Hattrick-Simpers, K. A. Brown, K. G. Reyes, J. Schrier, S. Billinge, T. Buonassisi, I. Foster, C. P. Gomes, J. M. Gregoire, A. Mehta, J. Montoya, E. Olivetti, C. Park, E. Rotenberg, S. K. Saikin, S. Smullin, V. Stanev and B. Maruyama, *Matter*, 2021, **4**, 2702–2726.

77 M. Abolhasani and E. Kumacheva, *Nat. Synth.*, 2023.

78 M. M. Noack, G. S. Doerk, R. Li, J. K. Streit, R. A. Vaia, K. G. Yager and M. Fukuto, *Sci. Rep.*, 2020, **10**, 17663.

79 K. G. Yager, P. W. Majewski, M. M. Noack and M. Fukuto, *Nanotechnology*, 2023, **34**, 322001.

80 G. S. Doerk, A. Stein, S. Bae, M. M. Noack, M. Fukuto and K. G. Yager, *Sci. Adv.*, 2023, **9**, eadd3687.

81 T. E. Oliphant, *Comput. Sci. Eng.*, 2007, **9**, 10–20.

82 G. Hinton and S. Roweis, *Proceedings of the 15th International Conference on Neural Information Processing Systems*, Cambridge, MA, USA, 2002, pp. 857–864.

83 J. D. Hunter, *Comput. Sci. Eng.*, 2007, **9**, 90–95.

84 A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai and S. Chintala, in *PyTorch: An Imperative Style, High-Performance Deep Learning Library*, Curran Associates Inc., Red Hook, NY, USA, 2019.