Digital Discovery

ROYAL SOCIETY OF CHEMISTRY

PERSPECTIVE

View Article Online
View Journal | View Issue

Cite this: *Digital Discovery*, 2022, **1**, 79

Natural language processing models that automate programming will transform chemistry research and teaching†

Glen M. Hocky *[a] and Andrew D. White *[b]

Received 2nd September 2021
Accepted 25th January 2022

DOI: 10.1039/d1dd00009h

rsc.li/digitaldiscovery

Natural language processing models have emerged that can generate useable software and automate a number of programming tasks with high fidelity. These tools have yet to have an impact on the chemistry community. Yet, our initial testing demonstrates that this form of artificial intelligence is poised to transform chemistry and chemical engineering research. Here, we review developments that brought us to this point, examine applications in chemistry, and give our perspective on how this may fundamentally alter research and teaching.

In 2021, Chen *et al.* released a new natural language processing (NLP) model called Codex that can generate code from natural language prompts.[1] Interest has been broadly focused on its application to software engineering. We, somewhat sarcastically, asked it to "compute the dissociation curve of H2 using pyscf"[2] and the result is shown in Fig. 1. It generated correct code and even plotted it (see ESI† for further details). Some may scoff at the artificial intelligence (AI) selected method (Hartree–Fock) and basis set (STO-3G). Thus, we asked it to "use the most accurate method" as a continuation of our "conversation" and it switched to CCSD in a large basis. AI models that can connect natural language to programming will have significant consequences for the field of chemistry—here we outline a brief history of these models and our perspective on where these models will take us.

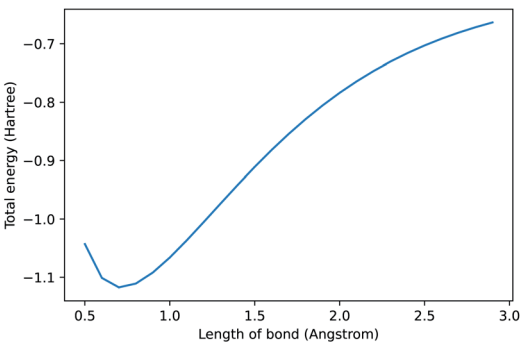


Fig. 1 Prompt: compute the dissociation curve of H2 using the pyscf library. See ESI† for code. Note, when repeating this prompt, the method, labels, and range can change due to under-specification of the request.

Recent developments

There has been a flurry of advances in the topic of "autocomplete" style language models that can generate text given a prompt. These language models are deep neural networks with a specific architecture called transformers.[3,4] These models are trained on text that has words hidden,[5] and have the task of filling in missing text.[4,6,7] This is called "pre-training," because these models were not intended to fill in missing words, but rather be used on downstream tasks like classifying sentiment in text or categorizing text.[4] Surprisingly, it was found that these models could generate a long seemingly real passage of text simply from a short initial fragment of text called a prompt.[4,8] These prompts can be to answer a question, summarize a story, or make an analogy—all with the same model. This was interesting, especially because the quality was beyond previous text generation methods like recurrent neural networks or hidden Markov models.[9] After increasing model size and the training corpus, the next generation of language models were able to answer novel prompts beyond standard question-and-answer or writing summaries.[10] For example, given three worked out examples of extracting compound names from a sentence, the GPT-3 model could do the same for any new sentence. We show the utility of this for parsing chemistry literature in Fig. 2 using text from ref. [11] (see ESI† for full details). This result is remarkable because it requires no additional training, just the input prompt–literally a training size of 3. Not so long ago, this was considered a difficult problem even when using thousands of training examples.[12] A caveat to these large language models (LLMs) is that they have a limited understanding of the text

[a]*Department of Chemistry, New York University, USA. E-mail: hockyg@nyu.edu*
[b]*Department of Chemical Engineering, University of Rochester, USA. E-mail: andrew.white@rochester.edu*

† Electronic supplementary information (ESI) available. See DOI: 10.1039/d1dd00009h

```
Sentence:  In brief, fluorescent dyes dissolved
in tetrahydrofuran (THF) were added to particles
stabilized with Pluronic F108 (MilliporeSigma)
to a final concentration of 30% v/v THF, then
diluted by a factor of five before washing the
particles via multiple sedimentation and
resuspension cycles to set them in pure water.
Chemical Entities: Pluronic F108,
tetrahydrofuran, THF
```

Fig. 2 Example of chemical entity recognition after training on three examples with GPT-3. Prompt including a direct quote of text from ref. [11] is in monospace and response is bolded and red. Note that this misses the connection between tetrahydrofuran and THF, and does not associate water with a chemical entity.

which they parse or generate; for example, we find they can generate seemingly valid chemistry text but cannot answer simple questions about well known chemical trends.

After these new LLMs were developed, anyone could have state-of-the art performance on language tasks simply by constructing a few examples of their task. In the last few months, even the need for worked out examples can be removed. In some cases a simple 'imperative' sentence is enough.[13] For example, a variation on the name of this article was generated by asking an imperative-style model to "write an exciting title" given an earlier version of the abstract. The pace has been nothing short of remarkable, going from the transformer in 2017 to a near universal language model in 2020 to a model which can take instructions in 2021.

The largest and arguably most accurate model in this class is still the GPT-3 model from OpenAI.[10] GPT-3 is an enigma in the field of natural language models. It is democratizing because anyone can create a powerful language model in a few hundred characters that is deployable immediately. Yet its weights are a pseudo-trade secret, owned and licensed by OpenAI exclusively to Microsoft. Thus the only way to run it is *via* their website (or API). These kinds of models are known as Large Language Models. Any state-of-the-art language models should start with a LLM like GPT-3 or, for example, the freely available GPT-NEO.[14] GPT-3 has been trained on billions of tokens and no effort has yet to match its scale of training data and model size. It can be unsettling too because it has quite adeptly captured the racism, sexism, and bias in human writing and can be reflected in its responses.[15] Mitigating this is an ongoing effort.[16] Another interesting outcome is that "prompt engineering," literally learning to interface more clearly with an AI, is now a research topic.[17]

GPT-3 has yet to make a major impact on chemistry, likely because it was available starting only in 2021. We previously prepared a demo of voice-controlled molecular dynamics analysis using GPT-3 to convert natural language into commands.[18] Although an impressive example of voice controlled computational chemistry had been published using Amazon's Alexa,[19] we found in our work that GPT-3 could handle looser prompts such "wait, actually change that to be ribbons." It also took only about a dozen examples to teach GPT-3 how to do tasks like render a protein, change its representation, and select specific atoms using VMD's syntax.[20] This is a significant reduction in researcher effort to make such tools, only taking a few hours total between the two of us. Our program itself adds an element of accessibility for those who may have difficulty with a keyboard and mouse interface through this voice-controlled interface, and we could easily, and plan to, generalize this approach to other analysis software used in our groups.

Perhaps because programmers were the most excited about GPT-3, frequent usage examples involved the generation of code. And thus we reach the present, with OpenAI's release in August of a GPT-3 model tuned explicitly for this purpose, termed Codex.[1] Although automatic code generation in chemistry is not new (*e.g.* ref. [21–23]), we believe that the scope and natural language aspects mean that code-generating LLMs like Codex will have a broad impact on both the computational and experimental chemistry community. Furthermore, Codex is just the first capable model and progress will continue. Already in late 2021 there are models that surpass GPT-3 in language[24] and equal it but with 1/20th the number of parameters.[25]

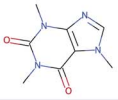
Over time, there has been a tremendous increase in the number of available software packages to perform computational chemistry tasks. These off-the-shelf tools can enable students to perform tasks in minutes which might have taken a large portion of their PhD to complete just ten years ago. Yet now, a large fraction of a researcher's time that used to be spent on repetitive coding tasks has been replaced by learning the interfaces to these numerous software packages; this task is currently done by a combination of searching documentation pages on the web, reading and following tutorial articles, or simply by trial and error. These new NLP models are able to eliminate intermediate steps and allow researchers to get on with their most important task, which is research! Some successful examples we have tried are shown in Fig. 3, with full details in the ESI.† While reading these examples, remember that the model does not have a database or access to a list of chemical concepts. All chemistry knowledge, like the SMILES string for caffeine in example A, is entirely contained in the learned floating point weights. Moreover, keep in mind that Codex may produce code that is apparently correct and even executes, but which does not follow best scientific practice for a particular type of computational task.

Immediate impact on research and education

Scientific software

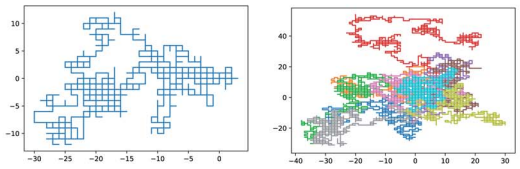
Many scientific programming tasks, whether for data generation or data analysis, are tedious and often repetitive over the course of a long research project. Codex can successfully complete a wide range of useful scientific programming tasks in seconds with natural language instructions, greatly reducing time to completion of many common tasks. These could include writing a function to convert between two different file formats, producing well formatted plots with properly labeled axes, converting LATEX equations into a function, implementing standard algorithms such histogramming, adding comments to code, and converting code from one programming

A 1. Load the caffeine molecule from SMILES
2. Draw it



B
```
def gauss(smiles):
    """This function prepares a Gaussian input file from the
given smiles using rdkit to prepare initial coordintes by
    1. Converting SMILES to molecule
    2. Preparing 3D coordinates
    3. Writing out Gaussian input file"""
    # Convert SMILES to molecule
    mol = Chem.MolFromSmiles(smiles)
    # Generate 3D coordinates
    mol = Chem.AddHs(mol)
    AllChem.EmbedMolecule(mol)
    AllChem.MMFFOptimizeMolecule(mol)
    # Write out Gaussian input file
    file_name = smiles + '.com'
    with open(file_name, 'w') as f:
        f.write('%chk=' + smiles + '.chk\n')
        f.write('%nprocshared=4\n')
        f.write('%mem=4GB\n')
        f.write('#p opt b3lyp/6-31g(d) empiricaldisper-
sion=gd3\n\n')
        f.write(smiles + '\n\n')
        f.write('0 1\n')
        for atom in mol.GetAtoms():
            pos = mol.GetConformer().GetAtomPosition(a-
tom.GetIdx())
            f.write('{:<4} {:>10.5f} {:>10.5f} {:>10.5f}\n'.format(a-
tom.GetSymbol(), pos.x, pos.y, pos.z))
        f.write('\n\n')
    return file_name
```

C - This script generates the behavior of a 2D random walk and plots trajectories
- Rewrite this into a function



D
```
# this function downloads a pdb file from the web and
saves it to the current directory
def download_pdb(pdb_id):
    pdb_file = pdb_id + ".pdb"
    url = "https://files.rcsb.org/download/" + pdb_file
    response = urllib.request.urlopen(url)
    data = response.read()
    pdb_file = open(pdb_file, 'wb')
    pdb_file.write(data)
    pdb_file.close()
    return pdb_file

# download the structure of 1lys
pdb_id = "1lys"
pdb_file = download_pdb(pdb_id)

# now compute the radius of gyration of 1lys with mdtraj
# load the pdb file into mdtraj
traj = md.load(pdb_file.name)
# compute the radius of gyration
rg = md.compute_rg(traj)
print(rg)
```

Fig. 3 Example prompts and either resulting code (B and D), or final figures that emerged from running the resulting code (A and C) (full details in the ESI†). Examples are in Python because our prompts include characteristics of Python code comments, but Codex can work in nearly any programming language included in its corpus.

language to another.[1] We have even found that Codex is capable of performing some of these tasks using non-english prompts, which could help reduce barriers to accessing software libraries faced by non-native speakers—although result accuracy when using non-English prompts has not been fully explored. Codex is not always successful. However, the rapid pace of progress in this field shows that we should begin to think seriously about these tasks being solved.

Will using code from Codex make chemists better or worse programmers? We think better. Codex removes the tedium of programming and lets chemists focus the high-level science enabled with programs. Furthermore, the process of creating a prompt string, mentally checking whether it seems reasonable, testing that code on a sample input, and then iterating by breaking down the prompt string into simpler tasks will result in better algorithmic thinking by chemists. The code generated, if not guaranteed to be correct, at least satisfies common software coding conventions with clear variable names, and typically employs relevant software libraries to simplify complex tasks. We ourselves have learned about a number of existing chemistry software libraries that we would not have discovered otherwise through our iterative prompt creation. Note though that Codex does not need to have *a priori* knowledge of how to use your software of interest; API usage can be suggested as part of the prompt similar to how the task is defined in Fig. 2.

Classroom settings

We and many of our colleagues around the world have begun introducing programming assignments as a component of our courses (especially in physical chemistry);[26] this has dual pedagogical purposes of reinforcing the physical meaning underlying the equations we scribble on the board, and teaching our students a skill that is useful both for research and on the job market. One of us has even written a book on deep learning in chemistry and materials science based around this concept.[27] But will code generation models result in poor academic honesty, especially when standard problems can be solved in a matter of seconds (Fig. 3)? Realistically we have few methods to police our students' behavior in terms of collaborating on programming assignments or copying from web resources. We rely, at least in part, on their integrity. We should rethink how these assignments are structured. Firstly, we currently limit the difficulty of programming assignments to align with the median programming experience of a student in our course. Perhaps now we can move towards more difficult and compound assignments. Secondly, we can move towards thinking of these assignments as a laboratory exercise, where important concepts can be explored using the software rather than concentrating on the process of programming itself. Lastly, our coursework and expectations should match the realities of what our students will face in their education and

careers. They will always have access to web resources and, now, tools like Codex. We should embrace the fact that we no longer need to spend hours emphasizing the details of syntax, and instead focus on higher level programming concepts and on translating ideas from chemistry into algorithms.

Ongoing challenges

Access and price

Currently, access to advanced models from OpenAI and tools like GitHub copilot are limited to users accepted into an early tester program. Pricing from the GPT-3 model by OpenAI indicates a per-query cost that is directly proportional to the length of the input prompt, typically on the order of 1–3 cents per query. This model may of course change, but it is reasonable to expect that Codex will not be free until either there are competing open-source models or the hardware required for inference drops in price. Depending on this cost structure, these commercial NLP models may be inaccessible to the academic community, or to all but the best funded research groups and universities. For example, a group might need to run hundreds of thousands of queries to parse through academic literature and tens of thousands for students in a medium size course, and these would certainly be cost prohibitive. Models developed by the open source community currently lag commercial ones in performance, but are freely useable, and will likely be the solution taken up in many areas of academia. However, even these models require access to significant computational resources to store and execute the models locally, and so we encourage the deployment of these models by researchers who have such computational resources in a way in which they can be equitably available.

Correctness

Code generation models do not guarantee correctness. Codex typically generates correct code at about a 30% rate on a single solution on standard problems, but improves to above 50% if multiple solutions are tried.[1] In practice, we find that mistakes occur when a complex algorithm is requested with little clarity. Iterating by breaking a prompt into pieces, chaining together prompts into a dialogue, and giving additional clues like a function signature or imports usually yields a solution. The code generated rarely has syntax mistakes, but we find it fails in obvious ways (such as failing to import a library, or expecting a different data type to be returned by a function). Over-reliance on AI-generated code without careful verification could result in a loss of trust in scientific software and the analysis performed in published works. However, this is already an issue in scientific programming and strategies to assess correctness of code apply equally to human and AI-generated code. Interestingly, Codex can generate unit tests for code, although it is not clear that this strategy can identify its own mistakes.

Because the accuracy of Codex depends strongly on how the prompts are phrased, it remains unclear how accurate it can be for chemistry problems. We are currently developing a database of chemistry and chemical engineering examples that can be used to systematically evaluate LLM performance in these and related domains. A second question remains as to whether the code produced is scientifically correct (and best practice when multiple solutions exist) for a given task, which will still require expert human knowledge to verify for now. We also note that in practice some of the correctness is ensured by default settings of chemistry packages employed in the Codex solution, just as they might be with human generated code.

Fairness/bias

As discussed in the Codex paper,[1] there are a number of possible issues related to fairness and bias which could accrue over time. The use of AI generated code, and then the updated training of that AI on the new code, could lead to a focus on a narrow range of packages, methods, or programming languages. For example, Python is already pushing out other programming languages in computational chemistry and this could increase due to the performance of Codex in Python over languages like Fortran or Julia. Another example we noticed is the preference of Codex to generate code using certain popular software libraries, which could lead to consolidation of use. For example, a single point energy calculation shown in the ESI† selects the package Psi4 if the model is not prompted to use a particular software.

Outlook

There are many exciting ways in which AI techniques are being integrated into chemistry research [ref. [28–30]]. Bench chemists have expressed the fear that automation will reduce the need for synthetic hands in the lab.[31] Now it looks like these NLP models could reduce the need for computational chemists even sooner. We disagree in both cases. Better tools have not reduced the need for scientists over time, but rather expanded the complexity of problems that can be tackled by a single scientist or a team in a given amount of time. Despite the challenges in the previous section, we foresee the use of NLP models in chemistry increasing accessibility of software tools, and greatly increasing the scope of what a single research group can accomplish.

Data availability

All prompts and multiple response are presented in the ESI.† Code was executed in Python 3.8. Access to OpenAI Codex and GPT-3 is governed by OpenAI and not the authors.

Conflicts of interest

There are no conflicts to declare.

Acknowledgements

Research reported in this work was supported by the National Institute of General Medical Sciences of the National Institutes of Health under award number R35GM137966 (to ADW) and R35GM138312 (to GMH). We thank John D. Chodera for

a helpful discussion on Twitter of how some code examples could be seemingly correct while producing poor or incorrect answers if it is not checked that a proper version of an algorithm is employed.

References

1 M. Chen, J. Tworek, H. Jun, Q. Yuan, H. Ponde, J. Kaplan, H. Edwards, Y. Burda, N. Joseph and G. Brockman, *et al.*, *Evaluating large language models trained on code*, arXiv:2107.03374, 2021.
2 Q. Sun, T. C. Berkelbach, N. S. Blunt, G. H. Booth, S. Guo, Z. Li, J. Liu, J. D. McClain, E. R. Sayfutyarova, S. Sharma, *et al.*, Pyscf: the python-based simulations of chemistry framework, *Wiley Interdiscip. Rev.: Comput. Mol. Sci.*, 2018, **8**, e1340.
3 A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser and I. Polosukhin, Attention is all you need, in *Advances in neural information processing systems*, 2017, pp. 5998–6008.
4 J. Devlin, M.-W. Chang, K. Lee and K. Toutanova, *Bert: Pre-training of deep bidirectional transformers for language understanding*, arXiv:1810.04805, 2018.
5 More generally, "tokens" are masked.
6 W. L. Taylor, "cloze procedure": A new tool for measuring readability, *Journalism quarterly*, 1953, **30**, 415.
7 A. M. Dai and Q. V. Le, Semi-supervised sequence learning, *Advances in Neural Information Processing Systems*, 2015, **28**, 3079.
8 A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, *et al.*, Language models are unsupervised multitask learners, *OpenAI blog*, 2019, **1**, 9.
9 I. Sutskever, J. Martens, and G. E. Hinton, Generating text with recurrent neural networks, in *ICML*, 2011.
10 T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry and A. Askell, *et al.*, *Language models are few-shot learners*, arXiv preprint arXiv:2005.14165, 2020.
11 T. Hueckel, G. M. Hocky, J. Palacci and S. Sacanna, Ionic solids from common colloids, *Nature*, 2020, **580**, 487.
12 M. Krallinger, F. Leitner, O. Rabal, M. Vazquez, J. Oyarzabal and A. Valencia, Chemdner: The drugs and chemical names extraction challenge, *J. Cheminf.*, 2015, **7**, 1.
13 Unpublished, but part of ongoing work known as davinci-instruct GPT-3 variant.
14 S. Black, L. Gao, P. Wang, C. Leahy and S. Biderman, *GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow*, 2021, If you use this software, please cite it using these metadata.
15 E. M. Bender, T. Gebru, A. McMillan-Major and S. Shmitchell, On the dangers of stochastic parrots: Can language models be too big?, in *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, 2021, pp. 610–623.
16 I. Solaiman and C. Dennison, *Process for adapting language models to society (palms) with values-targeted datasets*, arXiv preprint arXiv:2106.10328, 2021.
17 L. Reynolds and K. McDonell, Prompt programming for large language models: Beyond the few-shot paradigm, in *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*, 2021, pp. 1–7.
18 https://github.com/whitead/marvis.
19 U. Raucci, A. Valentini, E. Pieri, H. Weir, S. Seritan and T. J. Martínez, Voice-controlled quantum chemistry, *Nat. Comp. Sci.*, 2021, **1**, 42.
20 W. Humphrey, A. Dalke and K. Schulten, Vmd: visual molecular dynamics, *J. Mol. Graphics*, 1996, **14**, 33.
21 M. K. MacLeod and T. Shiozaki, Communication: Automatic code generation enables nuclear gradient computations for fully internally contracted multireference theory, *J. Chem. Phys.*, 2015, **142**, 051103.
22 J. Austin, A. Odena, M. Nye, M. Bosma, H. Michalewski, D. Dohan, E. Jiang, C. Cai, M. Terry and Q. Le, *et al.*, *Program synthesis with large language models*, arXiv preprint arXiv:2108.07732, 2021.
23 T. Zirwes, F. Zhang, J. A. Denev, P. Habisreuther and H. Bockhorn, Automated code generation for maximizing performance of detailed chemistry calculations in openfoam, in *High Performance Computing in Science and Engineering'17*, Springer, 2018, pp. 189–204.
24 J. W. Rae, S. Borgeaud, T. Cai, K. Millican, J. Hoffmann, F. Song, J. Aslanides, S. Henderson, R. Ring and S. Young, *et al.*, *Scaling language models: Methods, analysis & insights from training gopher*, arXiv preprint arXiv:2112.11446, 2021.
25 S. Borgeaud, A. Mensch, J. Hoffmann, T. Cai, E. Rutherford, K. Millican, G. v. d. Driessche, J.-B. Lespiau, B. Damoc and A. Clark, *et al.*, *Improving language models by retrieving from trillions of tokens*, arXiv preprint arXiv:2112.04426, 2021.
26 A. Ringer McDonald, Teaching programming across the chemistry curriculum: A revolution or a revival?, in *Teaching Programming across the Chemistry Curriculum*, ACS Publications, 2021, pp. 1–11.
27 A. D. White, *Deep Learning for Molecules and Materials*, 2021.
28 J. A. Keith, V. Vassilev-Galindo, B. Cheng, S. Chmiela, M. Gastegger, K.-R. Müller and A. Tkatchenko, Combining machine learning and computational chemistry for predictive insights into chemical systems, *Chem. Rev.*, 2021, **121**, 9816, DOI: 10.1021/acs.chemrev.1c00107pMID: 34232033.
29 N. Artrith, K. T. Butler, F.-X. Coudert, S. Han, O. Isayev, A. Jain and A. Walsh, Best practices in machine learning for chemistry, *Nat. Chem.*, 2021, **13**, 505.
30 R. Pollice, G. dos Passos Gomes, M. Aldeghi, R. J. Hickman, M. Krenn, C. Lavigne, M. Lindner-D'Addario, A. Nigam, C. T. Ser, Z. Yao, *et al.*, Data-driven strategies for accelerated materials design, *Acc. Chem. Res.*, 2021, **54**, 849.
31 Chemjobber, Will robots kill chemistry?, *Chem. Eng. News*, 2019, **97**(15), 25–25.