Paper review of "Toolformer: Language Models Can Teach Themselves to Use Tools"

Alexandra Ioana Neagu
TU Delft
neagu.alexandra0206@gmail.com

Abstract

In this review, I carefully assessed a paper on natural language processing (NLP), ***Toolformer: Language Models Can Teach Themselves to Use Tools*** (Schick et al., 2024), focusing on its strengths, weaknesses, experimental evaluation, and reproducibility. The paper introduces Toolformer, a language model trained to utilize various external tools via API calls in a self-supervised manner. While the work demonstrates promising results and innovative methodology, there are limitations in its ability to handle chained tool usage and interactive tool engagement. The experimental evaluation addresses key research questions, highlighting Toolformer's effectiveness in improving model performance across different downstream tasks. However, the reproducibility of the work may vary depending on the accessibility and specificity of the data and parameters provided. Overall, the review provides valuable insights into the paper's contributions and areas for potential improvement.

1 What is this paper about, and what contributions to NLP does it make?

This paper introduces **Toolformer**, a large language model (LLM) that can autonomously decide when and how to use external tools through APIs to enhance its performance. Toolformer addresses several limitations of current LMs, such as their inability to access up-to-date information, perform precise calculations, and reduce hallucinations. The model learns to use tools like a calculator, Q&A system, search engine, translation system, and calendar in a self-supervised manner, requiring only a handful of demonstrations for each API.

The key innovation is enabling the model to autonomously decide which APIs to call, when to call them, what arguments to pass, and how to incorporate the results into its predictions. This approach maintains the generality of the LM and significantly improves zero-shot performance on various downstream tasks. The main contributions of the paper entail:

- **Autonomous tool use**: Toolformer represents a novel approach to integrating external tools into LMs, allowing them to autonomously decide when and how to use tools like calculators, search engines, and translation systems. This self-directed tool use enhances the model's ability to handle tasks that typically require specialized knowledge or up-to-date information.

- **Self-supervised learning**: The method relies on self-supervised learning, eliminating the need for extensive human annotations. The model generates and filters its own API calls based on a self-supervised loss function, which identifies calls that improve future token prediction. This reduces the cost and potential biases associated with human-generated annotations.

- **Improved performance**: Toolformer demonstrates significant improvements in zero-shot performance across a range of tasks. It outperforms the much larger GPT-3 model and other baselines, showcasing the effectiveness of the approach despite using a smaller model (based on GPT-J with 6.7B parameters).

- **Preservation of generality**: The approach ensures that the model does not lose its general LM capabilities. By applying the tool-use learning method to the same dataset used for pretraining, Toolformer retains its broad applicability while enhancing its performance on specific tasks.

Furthermore, the paper provides a thorough evaluation of Toolformer's performance on diverse downstream tasks. This includes tasks that require up-to-date information, precise calculations, and specialized knowledge, demonstrating the model's versatile and enhanced capabilities. This tool aims to address problems such as the inability of LMs to access current information and perform precise calculations, the tendency of LMs to hallucinate facts, challenges in understanding low-resource languages, the requirement for human annotations in existing approaches for tool integration, and task-specific limitations of current tool-use methods. They do so mainly by utilizing existing methodologies in innovative ways: the introduction of a self-supervised approach for tool-use learning, the use of in-context learning to generate and filter API call datasets, and fine-tuning the LM on useful API calls to enhance its performance. These contributions significantly advance the field of NLP by demonstrating a scalable and efficient method for integrating external tools into LMs, leading to improved performance without sacrificing the generality of the language model.

1.1 Paper summary

The *approach* section describes how to empower an LM with the capability to utilize various tools via API calls, enhancing its performance on a wide range of tasks. The process involves several steps: representing API calls as tuples, augmenting a dataset with API calls, executing API calls, filtering them based on their usefulness, finetuning the LM on the augmented dataset, and implementing inference. API calls are represented as tuples, consisting of the API name and corresponding input. The dataset is augmented with API calls using in-context learning, where the LM samples potential API calls, executes them and filters out those that improve future token prediction. After filtering, API calls are merged with the original inputs to create the augmented dataset for LM finetuning. During inference, the LM utilizes the inserted API calls by interrupting the decoding process to obtain responses and continuing decoding after incorporating the responses. This approach enables the LM to autonomously decide when and how to use different tools based on its own feedback, enhancing its performance across tasks.

In *Section 3*, the paper explores a range of tools aimed at addressing different limitations of traditional LMs. The criteria for selecting these tools are twofold: they must be representable with text sequences for both inputs and outputs, and a few demonstrations of their usage must be obtainable. The paper introduces five tools: a question-answering system, a Wikipedia search engine, a calculator, a machine translation system, and a calendar API. The question-answering system, based on Atlas, focuses on factual queries; the calculator performs basic arithmetic operations; the Wikipedia search engine retrieves snippets for comprehensive information; the machine translation system translates phrases into English from various languages; and the calendar API provides temporal context. These tools collectively aim to enhance the LM's capabilities across different domains, offering solutions to various shortcomings.

The experiments conducted in *Section 4* aim to evaluate whether the proposed approach allows a model to effectively utilize external tools without additional supervision, determining when and how to employ them. Using a subset of CCNet as the LM dataset and GPT-J as the LM, the researchers define heuristics to identify texts where API calls are likely to be beneficial. They then augment the dataset with API calls using the described method in *Section 2*. The models are finetuned on this augmented dataset using different thresholds for filtering API calls. Baseline models include GPT-J without finetuning, GPT-J finetuned on the original dataset without API calls and Toolformer, the model finetuned on the augmented dataset. Additionally, the experiments consider larger models, including OPT (66B) and GPT-3 (175B), to assess scalability. Evaluation is conducted across various downstream tasks, including LAMA, mathematical reasoning, question answering, multilingual question answering, and temporal datasets. Results show that Toolformer consistently outperforms baseline models across different tasks, demonstrating its ability to effectively leverage external tools. Furthermore, the experiments show that finetuning with API calls does not degrade LM performance, and the model's ability to benefit from external tools improves with model size. Overall, the findings suggest that the proposed approach enables models to make effective use of external tools for a range of tasks, with performance scaling with model size.

In *Section 5*, the researchers analyze the impact of their modified decoding strategy, which introduces the option to generate the <*API*> token if it is one of the k most likely tokens instead of always generating the most likely token. They evaluate this strategy on the T-REx subset of LAMA and the WebQS dataset for different values of k, showing that increasing k leads to a higher percentage of examples where the model makes API calls. Additionally, they observe that for k=1, the model tends to make API calls for examples where it would perform particularly poorly without them, indicating a degree of calibration. However, this calibration diminishes for higher values of k.

Furthermore, the researchers conducted a qualitative analysis of the data quality of API calls generated using their approach for different APIs. They present examples of texts from CCNet augmented with API calls, along with the corresponding scores used as a filtering criterion, and assess whether the API calls made by the model are intuitively useful in the given context. They note that high values of the filtering criterion typically correspond to useful API calls, while low values indicate API calls that do not provide useful information for predicting future tokens. However, they also observe that some API calls that are not filtered out may contain noise but could still be useful in training the model to not blindly follow the results of each API call it makes.

The *related work section* discusses various approaches in LM pretraining, tool use, and bootstrapping techniques. Different methods augment LMs with additional textual information during pretraining. These may include metadata, HTML tags, Wikipedia markup, or related texts obtained from an information retrieval system. Toolformer differs by learning to explicitly request relevant information rather than relying on predetermined augmentation methods. Some approaches equip LMs with the ability to use external tools such as search engines, web browsers, calculators, translation systems, and Python interpreters. These methods either rely on large amounts of human supervision or work in a few-shot setup tailored towards specific tasks. In contrast, Toolformer, being self-supervised, learns how and when to use tools without specific prompts. The concept of using self-training and bootstrapping techniques to enhance models has been explored in various domains, including word sense disambiguation, relation extraction, parsing, sequence generation, few-shot text classification, retrieval, and reasoning. Toolformer utilizes a similar approach by training on its own predictions after applying a perplexity-based filtering step to improve model performance and reduce perplexity on future tokens.

The final sections discuss the limitations of Toolformer and conclude by summarizing its contributions. In terms of limitations, one significant constraint is Toolformer's inability to utilize tools in a chain, where the output of one tool becomes the input for another. This limitation arises from the independent generation of API calls for each tool, lacking examples of chained tool use in the finetuning dataset. Additionally, Toolformer currently lacks interactive tool use capabilities, particularly noticeable with tools like search engines, where the ability to browse through multiple results or refine search queries could be advantageous. Furthermore, Toolformer models exhibit sensitivity to the exact wording of their input when deciding whether to initiate an API call, consistent with the well-known sensitivity of LMs to prompts in both zero- and few-shot settings. The method's sample inefficiency is another concern, especially apparent with certain tools, as processing over a million documents may only yield a few thousand examples of useful API calls to the calculator API. One potential solution could involve the iterative application of Toolformer or similar bootstrapping approaches to address this inefficiency. Lastly, Toolformer's decision-making process for API calls does not consider the computational cost associated with each call, representing a potential area for improvement in future iterations of the model.

In conclusion, Toolformer represents a significant advancement by introducing a self-supervised learning approach for LMs to utilize various tools through simple API calls. Its finetuning process significantly enhances the zero-shot performance of GPT-J models, even surpassing larger GPT-3 models across a range of downstream tasks. Despite its current limitations, Toolformer lays a foundation for further research in leveraging external tools for language understanding and generation tasks.

2 What strengths does this paper have?

This paper demonstrates several strengths that contribute to its overall quality and potential impact. The paper introduces a novel approach called Toolformer, which enables LMs to utilize external tools through API calls in a self-supervised manner. This innovative methodology represents a significant advancement in the field of natural language understanding and generation. It provides comprehensive empirical results demonstrating the effectiveness of Toolformer. Through extensive experimentation, the authors show that models trained with Toolformer achieve improved zero-shot performance across various downstream tasks compared to traditional LMs.

Personally, I found the paper well-written and that it effectively communicates the methodology, experimental setup, and results. The sections are structured logically, making it easy for readers to follow the progression of the research and understand the key findings. The authors conduct a thorough analysis and discussion of the limitations of Toolformer, highlighting areas for future research and improvement. This critical examination adds depth to the paper and contributes to a better understanding of the model's capabilities and constraints.

Overall, this paper stands out for its innovative approach, rigorous experimentation, clear presentation, and insightful analysis, all of which contribute to its strengths and potential impact in the field of NLP.

3 What weaknesses does this paper have?

While the paper presents a compelling approach and offers valuable insights, there are some weaknesses that warrant consideration. The paper primarily focuses on demonstrating the effectiveness of Toolformer through empirical evaluation. However, the evaluation may be limited in scope, as it primarily assesses the model's performance on downstream tasks without delving deeply into the underlying mechanisms of tool utilization. A more thorough analysis of how Toolformer learns to use external tools and the specific contexts in which it succeeds or fails could provide deeper insights into its capabilities.

While the paper compares the performance of models trained with Toolformer against traditional LMs, it could benefit from a more extensive comparative analysis. For example, comparing Toolformer against other existing approaches for incorporating external knowledge or tools into language models would provide a more comprehensive understanding of its relative strengths and weaknesses.

The paper acknowledges certain limitations of Toolformer, such as its inability to use tools in a chain or in an interactive manner. However, it does not explore potential extensions or solutions to address these limitations in depth. Exploring strategies for enabling chained tool usage or interactive tool interaction could enhance the practical applicability and versatility of Toolformer.

While the paper outlines the general methodology of Toolformer, it may lack clarity in certain implementation details. Providing more detailed explanations or code snippets related to the training procedure, API call generation, or filtering criteria could facilitate a better understanding and reproducibility of the approach.

Lastly, the paper does not explicitly address potential ethical considerations associated with the use of external tools through Toolformer. Considering the ethical implications of accessing and utilizing external information sources, particularly in sensitive or controversial domains, is important for responsible AI research and deployment.

Overall, while the paper presents a promising approach for enhancing LMs with external tools, addressing these weaknesses could further strengthen its contribution to the field.

4 What are the research questions that are answered in the experimental evaluation?

In the experimental evaluation, the following research questions are addressed:

1. **RQ1: Effectiveness of Toolformer in Utilizing External Tools**: The experimental evaluation aims to assess the effectiveness of Toolformer in leveraging external tools, such as search engines, calculators, and translation systems, to improve the performance of LMs on downstream tasks. Key results include the demonstration that models trained with Toolformer outperform traditional LMs, even surpassing larger models, like GPT-3, across a range of tasks. This finding suggests that Toolformer effectively learns to utilize external tools to enhance its capabilities.

2. **RQ2: Impact of Sampled API Calls on Model Performance**: Another research question addressed in the experimental evaluation pertains to the impact of sampled API calls on model performance. The results indicate that filtering API calls based on their potential to improve zero-shot performance significantly enhances the performance of LMs finetuned with Toolformer. This insight highlights the importance of carefully selecting and filtering external information sources to optimize model performance.

Overall, the experimental evaluation provides valuable insights into the effectiveness and impact of Toolformer in leveraging external tools to enhance the capabilities of LMs.

5 Can you think of how you would improve the work presented in this paper?

To further improve the work presented in this paper, one potential avenue of exploration could be:

Research Question: How does incorporating a mechanism for the LM to learn from sequential API calls affect its performance and capabilities?

Justification: Current limitations of Toolformer include its inability to use tools in a chain, where the output of one tool serves as input for another. Allowing the LM to learn from sequential API calls could potentially enhance its ability to perform complex tasks that require multi-step processes. Enabling the LM to use tools in a sequential manner would align more closely with real-world scenarios where tasks often involve multiple steps and dependencies between different tools or information sources. Investigating the impact of sequential API calls on LM performance would provide insights into the effectiveness of such an approach and its potential to further enhance the capabilities of LMs in leveraging external tools.

6 How reproducible is the work?

Given the detailed methodology and appendix provided in the paper, researchers and practitioners would likely fall into category **3 out of 4** regarding reproducibility. While they could mostly reproduce the results described here, they might encounter some challenges due to the need to access certain proprietary data or tools. However, with appropriate substitutions or adjustments using publicly available data or tools, they should be able to replicate the core findings and verify the correctness of the results (Raff, 2019).

7 How confident are you in your assessment of this paper?

I would rate my confidence in the assessment of this paper as a **4 out of 5**. While I made a concerted effort to check the important points carefully and feel reasonably confident in my evaluation, there is always a chance that I might have missed some details or failed to fully grasp certain central points. However, I believe I have a good understanding of the general area and the key aspects of the paper's contributions.

References

Edward Raff. 2019. A step toward quantifying independently reproducible machine learning research. *Advances in Neural Information Processing Systems*, 32.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2024. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36.