이끄는 인간, 그리고 그 등을 밀어주는 인공지능

문성민

인공지능은 현재 다양한 문제를 풀 수 있을 정도로 발전했지만, 아직 그 한계 또한 명확합니다. 그 한계 중 하나는 인공지능의 구조로부터 비롯되는 핵심적인 문제로, 인간과 같이 논리적으로 사고할 수 없다는 사실입니다. 프로그래밍의 영역에서 역시 이 문제는 치명적이며, 아직 그 한계는 해결되지 않았습니다. 그러나 인공지능이 현재 해낼 수 있는 기능은 이미 충분히 강력하며, 부수적인 문제가 해결된다면 인공지능은 그 사용 방식에 따라 프로그래밍, 더 나아가 다른 수많은 분야에서도 효과적으로 쓰일 수 있을 것입니다. 그 과정에서 우리는 창조의 영역에, 인공지능은 모방의 영역에 머물며 우리와 인공지능이 각각 부족하거나 하기 어려운 영역을 보완하는 방식 또한 하나의 방법입니다.

인공지능이 현재 어떤 일을 해낼 수 있는지, 그리고 앞으로 얼마나 더 다양한 문제를 해결할 수 있을지는 많은 사람의 관심을 받는 주제입니다. 프로그래머, 컴퓨터 과학자뿐만 아니라 저처럼 다른 분야에서, 더 나아가 인간의 창의성을 요구하는 "연구"를 직업으로 삼고자 하는 사람들에게도 이는 중요한 문제입니다.

예전에 구글 딥마인드에서 개발한 인공지능이 "complex geometry problem"(수학 용어로는 복소수 기하학 문제)를 풀었다는 기사를 봤습니다. 복소수 기하학은 제 전공 분야인 수학에서 활발한 연구가 이루어지는 주제입니다. 이곳에 벌써 인공지능이 닿았음에 기대 반, 걱정 반으로 기사를 들여다보았지만 알고 보니 고등학교 수준의 "복잡한 도형 문제"를 풀었다는 것이었습니다(영어 단어 complex는 일상용어로 "복잡한"이란 뜻도 가집니다. 즉 "복잡한 기하학 문제"). 비록 제게는 이 사실이 아쉽게 느껴졌지만, 오히려 이런 측면이 현재의 인공지능이 어떤 위치에 있는지를 보여주고 있다고 생각합니다. 아직 어떤 사람도 해내지 못했던 새로운 일을 인공지능이 해내지는 못합니다. 그렇지만 누군가는 할 수 있으나 누구에게는 어렵거나 번거롭게 느껴지는 일이라면 인공지능은 그것을 해내어 후자의 사람들에게 제공할 수 있습니다.

인공지능은 여러 개의 함수와 다양한 특성(매개변수)의 값의 조합이며, 이를 효과적으로 만들기 위해서는 각 함수와 특성 하나하나를 변화시켰을 때, 전체 구조가 우리가 모방하고자 하는 목표(함수)에 얼마나 가까워지는지를 측정할 수 있어야 합니다. 이러한 방법을 적용할 수 없는 한계 중 하나는 논리입니다. 인공지능이 우리가 모방할 목표에 가까운 정도를 숫자와 같은 지표로 나타낼 수 있다면 효과적인 "측정"이 되겠지만 "예", "아니오"로만 결과가 나뉘는 논리는 이러한 효과적인 측정을 해내기 어렵기 때문입니다. 이 한계는 여러 분야에서 드러나며, 그중 컴퓨터 과학과 프로그래밍에서도 치명적입니다.

물론 이미 인공지능은 스스로 프로그래밍을 할 수 있습니다. 하지만 이는 인공지능이 학습했던 자료, 즉 사람들이 직접 작성했던 프로그램들을 모방하는 것입니다. 스스로 논리적인 사고를 통해 코드 한 줄 한 줄이 어떠한 기능을 하는지를 관찰하고 직접 조합하여 프로그램을 작성하는 것과 거리가 멀다는 것입니다. 때문 인공지능은 사람이 이전에 구현한 적 없는 새로운 기능, 혹은 창의적인 알고리즘을 스스로 만들어내지 못합니다. 그러나 다시 돌아와서, 인공지능은 누군가 개발한 프로그램을 효과적으로 학습해 언제든 스스로 다시 구현할 수 있습니다. 이것만으로도 인공지능의 기능은 굉장한 강점을 가집니다.

프로그래밍을 할 때 직관적으로 간단하고 쉬운 아이디어도 실제로 구현하고 프로그램을 작성하는 것은 꽤 많은 시간을 들여야 할 수 있습니다. 또한, 그 과정에서 본인의 실수 때문에 프로그램 안에 내재한 문제도 무시할 수 없습니다. 이를 검사하기 위해 테스트 케이스를 작성하고 확인하는 것도 많은 시간을 소요합니다. 하지만 이 과정은 우리가 만들고자 하는 프로그램의 전반적인 구조를 알고 있는 상태에서 진행되며, 이 실질적인 구현은 인공지능이 해결하기에 적절한 문제입니다. 만들고자 하는 프로그램의 구조를, 이미 잘 알려진 알고리즘의 단위로 분해하여 인공지능에 입력했을 때를 생각해봅시다. 인공지능은 다수 사람에게서 얻은 자료를 모방해 각 단위의 알고리즘들을 효율적으로 구현할 수 있고, 결과적으로 전체 프로그램을 빠르게 작성할 수 있습니다.

잠시 원론적인 이야기로 넘어가 보겠습니다. 프로그래밍의 목표는 주어진 입력에 대해 어떤 출력을 하는 기능을 만드는 것으로 볼 수 있습니다. 더 현실적으로는 그러한 기능을 하는 프로그램을 작성하는 것입니다. 하지만 이는 단순히 어떤 목표 프로그램이 있고, 그 프로그램을 완벽히 똑같이 작성하는 것과는 분명히 다릅니다. 우리가 목표로 하는 것은 프로그램의 기능입니다. 프로그램의 세세한 구성이 달라질 수 있고 그 안에 내재한 핵심 원리도 달라질 수 있더라도, 결국에 원하던 기능을 구현한다면 통상적으로 목표에 맞습니다. 이러한 특징은 프로그램의 실질적인 작성에서 융통성을 부여하며, 앞에서처럼 인공지능을 이용하는 것은 결국 우리가 택할 효율적인 프로그래밍의 방법 중 하나입니다.

돌아보면, 프로그래밍에서 우리가 할 수 있는 것과 부족한 것, 그리고 인공지능이 할 수 있는 것과 할 수 없는 것은 명확합니다. 우리는 논리적인 사고를 통해 프로그램과 그 동작을 머릿속에 그릴 수 있지만, 그것을 현실에 구현하고 표현하는 데에 많은 자원, 이를테면 많은 시간이 필요합니다. 반면에 인공지능은 사람의 논리적 사고를 모방하는 것을 아직 넘어설 수 없고 새로운 것을 만들 수 없습니다. 그러나 이미 누군가가 구현했던 기록과 프로그램에 대해서는 통상적인 사람에 비해 효과적, 효율적으로 모방하고 적은 자원으로 그것을 다시 구현할 수 있습니다.

이러한 관점에서 당분간 인공지능이 프로그래밍에 이용될 방식은 비교적 명확합니다. 우리는 프로그램에 내재한 논리를 해석하고 그 전반적인 구조를 효과적으로 설계할 수 있습니다. 그리고 인공지능은 그 설계를 실질적, 세부적으로 구현하는 것에서 사람이 직접 구현하는 기존 방식보다 효율적인 방향을 제공합니다. 효율적이라는 것은 곧, 개발자의 직접적인 구현에 소비될 자원이 줄어들 것이라는 뜻입니다. 더 구체적으로는 이미 잘 알려진 알고리즘의 반복적인 구현, 비전공자가 알기 어려운 프로그래밍 언어의 비직관적이고 기술적인 측면, 그 외 다양한 상황과 요소들을 인공지능이 해결해낼 수 있다는 것입니다. 결국에는 우리가 설계를 주도하고 인공지능은 이를 따라 효과적으로 설계를 구현하며 사람들이 본인의 발상력에 집중할 수 있고 생산성을 극대화할 수 있도록, 자리 잡을 수 있을 것입니다.

마지막으로, 더 나아가 이런 접근은 프로그래밍이라는 한 분야뿐만 아니라 그 외의 다양한 영역들, 심지어 연구에도 인공지능의 효과적인 기능이 자리 잡을 수 있게 할 것으로 생각합니다. 이전에 누군가가 연구했던 것, 발명했던 것에 대한 정보를 재사용하고 배우는 과정에서 인공지능은 프로그래밍에서의 경우처럼 새롭고 효과적인 방향을 제공할 것입니다. 그리고 이를 기반으로 우리는 오로지 자신의 발상과 창조에 집중할 수 있을 것입니다. 오늘 저 또한 이 글을 쓰는 과정에서 문장을 다듬고 어감을 고려해 새로운 단어를 탐색하는 과정에서 여러 번 ChatGPT를 이용했습니다. 덕분에 글을 어떤 내용으로 쓸 것인지 구상하는 데에 집중하며 적은 시간 동안 글을 완성할 수 있었으며, 동시에 더 나은 글이 되었다고도 생각합니다. 이런 순기능을 제가 앞으로 더 공부하고, 나아가 연구를 하며 많이 경험할 수 있기를 기대합니다.