<내가 궁금한 것, 내가 느낀 것, 상상하게 된 것.>

이예원

내가 궁금한 것

이번 파트를 읽고 계속 내 머리를 떠나지 않는 가장 주된 궁금증은 '적당히'와 '랜덤'이 가능한가, 라는 궁금증이다. 수업시간에 지나가듯 말씀하시기도 했지만, "완벽한 랜덤이란 존재할 수 없는 것일까?" 나는 "통밥"과 "무작위" 파트를 읽으며, 컴퓨터에게 '적당히' '아무거나'라는 명령을 어떻게 내릴 수 있는지가 가장 궁금했다. 고도로 정밀화된 시스템에 따라 '사용자에게 랜덤으로 보이도록' 조작되었다면, 그것을 무한대로 돌리면 결국에는 일정한 패턴을 가지게 될까? 강의를 듣고 나서 주변 사람들과 이 주제에 대해 가벼운 토론을 나누었는데, 어떤 사람의 주장에 의하면 '랜덤'이라는 것은 오히려 사람보다 기계가 더 정확하다고 한다. 그러니까, 기계에 의해 조작된 '랜덤'이 사람이 정말 직관에 의해 고르는 '랜덤'보다 훨씬 '무작위'하다는 것이다. 이는 통계에 의하면 사람이 무작위로 선택할 때에는 사람마다 무의식 중에 일정한 패턴을 따르게 된다는 것이 그 이유였다. 물론 '랜덤'의 정의를 '패턴 없음'이라고 한다면 그러한 주장이 가능할 수도 있다. 그렇다면 '랜덤'의 정의를 '패턴과는 상관 없는 직관'으로 정의한다면 어떻게 될까? 사실 랜덤보다도 더욱 신기한 것은 통밥의 원리이다. 직관적으로 얼추 맞을 것 같은 정답을 내놓는 것은 그 전까지의 케이스를 집대성한 확률을 따지는 매커니즘을 따르는 것일까? 하는 궁금증이 들었다.

또한 양자컴퓨터 파트를 공부하면서 '어디까지 자동일까?'라는 궁금증이 들었다. 양자 컴퓨터 말고도, 전체 파트에 걸쳐 종종 들었던 궁금증이다. 양자 탐색 알고리즘에서, 찾으려는 번호만 레이저를 쏴서 확률진폭을 반대 방향으로 뒤집는 장면, 그것은 어떻게 가능한 것인지 궁금했다. 번호를 찾기 위해서, 그 번호에 레이저를 쏜다는 것은 얼핏 순환 논증 같기도 하다. 문제는 이런식으로 '자동으로 해결되는' 분야가 종종 보인다는 것이다. 혹시 그 경계가 있는 것인지, 무슨 기준으로 갈리게 되는지 궁금증이 들었고, 탐색 해봐야 되겠다는 생각이 들었다.

마지막으로 컴퓨터 소프트웨어의 언어 파트, 자동 번역 부분을 보면서 현재 어마어마하게 발전하고 있는 AI와 관련 지어 궁금증이 생겼다. 현실 언어와 프로그래밍 언어의 가장 큰 차이점은 바로 프로그래밍 언어의 경우 한 단어에 이중의미가 존재하는 경우가 극소수라는 것이다. 이것은 프로그래밍 언어들이 자동번역을 할 수 있는 이유가 되기도 한다. 그러나 이 방식의 단점은, 현실 언어에서 표현할 수 있는 그 섬세한 영역, 중의성을 통한 의미 전달 등 여러 문학적이고 수사적인 표현방식을 구현할 수 없다는 것이다. 그런데 요즘 화제가 되고 있는 AI들 중에는, 소설을 써서 일본 문학상 예선을 통과한 인공지능이 있다고 한다. 어떤 방식으로 이것이 가능해지는 것일 것일까? 문학만은 컴퓨터가 절대 대체할 수 없는 영역이라고 생각했던 나에게 그 뉴스는 정말 충격적이었다. 이것에 대해서는 관련 자료나 논문을 통하여 조만간 더 알아보려고 하고 있다.

내가 느낀 것, 내가 상상하게 된 것.

'아, 이제부턴 정말 현실이구나,' 이번 블록의 내용을 읽었을 때 내가 느낀 첫 감상이었다. 가장 가시적으로 보이는 차이점은 이전의 역사, 이론파트와는 다르게 낯선 것들이 등장했다는 점이다. 신기한 회로도가 등장하고, P와 NP라는 용어가 등장하고, C언어, ML, 기계어, … 그리고 람다가 등장했다. 튜링의 아이디어 속 컴퓨터의 연원을 찾아 떠났던 역사 이야기와는 완전히 다른 결이었다. 또 한가지 본질적으로 이전 파트와 달랐던 점은 내용이 훨씬 구체화되고, 현실로 성큼 다가왔다는 점이다. 이제는 실제 튜링 머신을 어떻게 현실로 구현할 것인가를 구체적으로 고민하고, 소프트웨어(튜링머신)가 어떤 방식으로 구성되는지 알고리즘과 언어로 나누어 꼼꼼히 살펴본다.

일단은 조금 두려워서 주춤했다가, 쉬운 설명에 깜짝 놀랐다. 처음에는 이렇게 낯선 용어들이 불쑥 불쑥 등장하고, 너무나 갑작스럽게 튜링 머신을 0과 1만으로 현실에 구현해 버리려 하는 마법 같은 시도에 겁이 났었다. 그러나 막상 책을 읽으며, 찬찬히 내용을 따라가면서, 특히나 어려운 내용마다 빠지지 않고 첨언되어 있는 적절한 비유를 따라가면서는 생각했던 것만큼 이해 불가능한 개념이 아니라는 사실에 약간 놀랐다. 용어에 관해 수업 중에도 자주 언급하셨던 만큼, 최대한 언어 때문에 장벽을 느끼는 일이 없도록 쉬운 말만 골라 내용을 꾸렸고, 비유도 가장 적절하게 들어맞는 대상을 찾기 위해 얼마나 신중하게 골라 내었는지가 물씬 느껴졌다. 제대로 된 배경지식도 없는 내가 아무런 어려움 없이 용어의 뜻과 비유를 이해하는 것이 자칫 별 생각 없이 넘겨버리면 아무 것도 아닌 것 같지만, 한 번 더 생각해보면 그 자체가 기적이기 때문이다. 역시 전공은 속일 수 없는 지, 구체적인 내용보다도 내게 먼저 와 닿았던 것은 이런 식의 내용 구성, 스토리텔링 방식, 그리고 중간중간 감초처럼 첨부된 절묘한 시들이었다. 한편으로는 소프트웨어의 언어 파트에서 나오는 '번역사슬' 부분, 그 곳에서 설명하는 '원시 언어'와 '상위 언어'의 간격이 일견 이 책과도 통한다는 생각이 들었다. 비전공생들에게는 너무나 어려운 상위 언어 '컴퓨터 과학'과 일상어의 간극, 그 사이를 쉬운 말과 재미있는 비유로써 번역 사슬을 만들어 채우고 있는 것이다. 그 중에서도 백미는, 단연 람다 계산법의 눈사람 비유법이라고 생각한다. 이렇게 평이하고 간단한 방식으로 람다를 이해할 수 있다는 것에 감사함을 느꼈다.

본격적으로 내용을 탐구하면서 가장 감명받았던 부분은 "속내용을 감추며 차곡차곡 쌓기"이다. 이 지혜는 감히 '컴퓨터'의 처음부터 끝 까지를 관통하는 이해의 실마리라고 하고 싶다. 내가 컴퓨터 과학을 접하면서 가장 많이 느꼈던 고민은 늘 이런 종류의 것이었다. '여기서 내가 이 명령어를 입력했을 때 왜 이런 결과가 도출되는 걸까?' '어떤 방식으로? 어떻게 전기가 흘러서 컴퓨터가 이 말을 알아듣고 일을 수행하는가?' 그 고민은 늘 중간 중간 멈춰 서서 별 소득도 없이 나를 방황하게 만들었다. 그러나 이 지혜는 그러한 고민을 일시에 해결해준다. '층위가 다름'을 일러주기 때문이다. '내가 명령어를 입력'하는 차원과, '전기가 흘러 컴퓨터가 이 일을 수행'하는 차원은 분명히 다르다. 이 지혜는 컴퓨터 그 자체 뿐만 아니라 기계어와 상위언어의 차이에도 적용되는 보편적인 지혜이다. 상위언어를 쓰는 인간은 기계어를 모두 알고 있을 필요가 없다. 01010과 01101이 어떻게 다른 지 외우고 있을 필요가 없는 것이다. 얼마 없는 코딩 경험이지만, 코딩을 할 때마다 '내가 뭣도 모르면서 피상적으로, 남들이 하라는 대로 명령어만 두들기고 있는 것 아닐까'하는 회의에 빠져 있던 나에게 큰 가르침이 되었다.

다음으로는 "통밥" 개념을 공부하며, '적당히'에 대해 생각해보게 되었다. NP문제를 현실적인 비용으로 풀고 싶을 때 쓰는 꼼수 같은 방법인 이 통밥은, 솔직히 말하면 완벽을 포기하는 것이다. 나는 여기에서 두 가지 생각을 가지게 되었는데, 첫 번째는 컴퓨터가 '적당히'라는 명령을 받아들일 수 있을까?라는 생각이었다. 이것은 궁금한 점에 기록하였다. 두 번째 들었던 생각은, 통밥이라는 문제해결방식이 지금 현실적으로 어쩔 수 없다고는 하지만, 별로 마음에 들지 않는다는 생각이었다. 우선 주어진 틀 안에서는 흠 난 데 없이 완벽하고, 칼로 무 자르듯 답을 내 놓아야 하는 '기계' 컴퓨터의 옥의 티가 아닌가 하는 생각이 들었다. 어디에서나 어떨 때나 완벽하고 싶지만 시간에 쫓겨, 또는 공간에, 금전에, 다양한 현실적인 제약에 갇혀 완벽함을 포기해버리는 절망스러운 모습이 연상되었다. 물론 현실에 타협하는 똑똑한 문제해결방식이라고, 실제로 우리가 건드릴 수 없는 영역의 문제로 닿게 해주는 발판이라고 긍정적으로 생각할 수도 있다. 실제로 그렇기도 하고, 그것을 부정하고 싶지는 않다. 다만 개인적으로는 "통밥"이라는 해결방식을 사용하더라도, 좀 더 많은 사람들이 "통밥"같은 해결방식은 '미봉책'이며, 분명히 개선되어야 하는 해결방식이라는 생각을 늘 가지고 있었으면 좋겠다. "통밥"이라는 방법의 편리함에 속아서 발전을 잊는 일은 일어나지 않아야 한다고 생각한다. 사실 이것은 스스로에게 하는 다짐이기도 하다.

마지막으로 소프트웨어의 언어 파트를 읽으면서, 프로그래밍 언어와 현실 언어의 공통점과 차이점을 생각했다. 프로그래밍 언어는 현실 언어와 달리 한 단어 단 하나의 의미만이 허락되는 원칙을 확실하게 고수한다. 이것이 프로그래밍 언어 간 자동번역이 이루어질 수 있는 이유이다. 그런데, 프로그래밍 언어들은 현실 언어와 같이 다양한 종류, 조금씩 다른 스타일을 가지고 있다. 그리고 그것은 다른 방식의, 다른 관점의 생각을 유도하기도 한다. 이것은 매우 매력적인 요소이다. 생각해보면 이렇게 다양한 방식의 프로그래밍 언어가 생긴 것은 아마도, 분야마다 특화되어야 하는 지점이 달라서, 더 편하게 작업하기 위해 만들어진 것일 테다. 그런데 이렇게 필요에 의해 만들어진 다양한 언어들이, 전혀 의도하지는 않았지만 역으로 사고를 유도하게 되는 점이 신선하게 느껴졌다. 어쩌면 이것도 튜링 소년의 논문에서 나왔던 컴퓨터의 원형처럼, 생각치 못한 원석이 아닐까 하는 생각이 들었다. 언어마다 사고의 차이가 있고 따라서 같은 문제도 다른 관점에서 볼 수 있다는 것은 무한한 발전의 가능성을 내포한다고 생각한다. 사람들마다 생각이 달라서 토론하고, 싸워도 보고, 합의점을 찾으면서 진리의 일부가 어렴풋이 드러나는 것처럼 한 문제를 여러 언어들이 여러 관점에서 토론하고, 관찰하면서 훨씬 많은 정보가 얻어질 수 있다. 컴퓨터 과학의 발전 가능성을 단적으로 보여주는 예시인 것 같아 감명 깊었다.

이번 파트는 저번 파트와는 달리 훨씬 난이도가 어렵고, 구체적인 문제들이 많다 보니 수업을 들을 때도 책을 읽을 때도 훨씬 많은 에너지를 쏟았다. 그러나 훨씬 많은 에너지를 쏟은 만큼, 컴퓨터에 대해, 특히 소프트웨어라는 튜링머신에 대해 한 발짝 더 다가간 것 같아 보람차다. 벌써 결론을 제외하고는 마지막 하나의 장만을 남겨두고 있다. 아직 배울 것이 산더미라는 것은 알지만, 이때까지 엄두도 못 냈던 컴퓨터과학을 체계적으로 시작할 수 있었던 것이 처음부터 지금까지 늘 신기하고, 감사하다.

감사합니다!