

KAKAO

Vol.08

AI

2017.11

AI CODE

```
def similarities(self, srcs, dsts):
    sims = [self.w2v_similarity(word, ref_word)
             for word, ref_word in zip(srcs, dsts)]
    return self.geometric_mean(sims)

def clip_count(self, cand_d, ref_ds):
    count = 0
    for key, value in cand_d.items():
        key_max = 0
        words = key.strip().split()

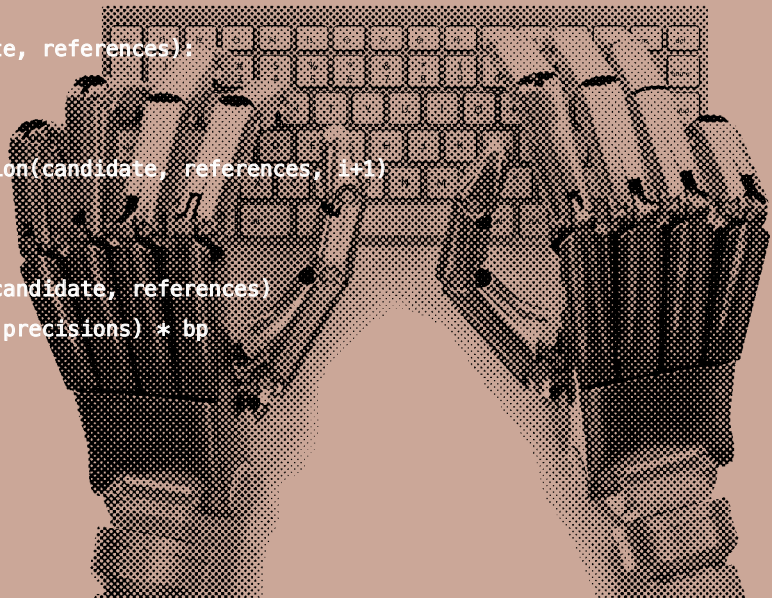
        for ref in ref_ds:
            for ref_key, ref_value in ref.items():
                ref_words = ref_key.strip().split()
                sim = self.similarities(words, ref_words)
                key_max = max(key_max, ref[ref_key]*sim)

        clipped_count = min(value, key_max)
        count += clipped_count
    return count
```

```
def advanced_BLEU(self, candidate, references):
    precisions = []
    for i in range(2):
        pr = self.ngram_precision(candidate, references, i+1)
        precisions.append(pr)

    bp = self.brevity_penalty(candidate, references)
    bleu = self.geometric_mean(precisions) * bp
    return bleu
```

kakao



카카오에서 매일 발행하는 시리포트입니다.

KAKAO AI REPORT

Vol. 08

발행일 | 2017년 11월 30일
발행처 | (주)카카오
발행인 | 카카오 정책지원파트
편집 | 김대원, 문정빈, 양원철, 양현서, 전수민
디자인 | 허진아
메일 | kakaoaireport@kakaocorp.com

COVER
카카오 AI 리포트의 표지에선 AI와 관련된 의미 있는 코드들을 매월 소개하고 있습니다.

Vol.08 코드 | 오형석 hulk.oh@kakaocorp.com
Parallel corpus 추출을 위한 alignment tool의 알고리즘 일부를 발췌한 것입니다. 카카오는 기존 규칙 기반 접근의 한계를 극복하기 위해 딥러닝 기반의 alignment 알고리즘을 사용하였고, BLEU에 word embedding 기법을 적용하여 ABLEU라는 더 개선된 측정법을 고안하여 사용했습니다. 이를 통해 양질의 parallel corpus를 대량으로 추출할 수 있었습니다.

contents	
preface	02
AI in Kakao	
기계번역기의 A to Z, 그리고 카카오의 기계번역	
김미훈 기계번역기의 역사와 발전	06
배재경 신경망 번역 모델의 진화 과정	10
오형석 카카오 번역기가 양질의 대규모 학습 데이터를 확보하는 방법	16
hot topic	
카카오미니와 제프리 힌튼 그리고 ICCV	
조희주 김수형 카카오미니는 어떻게 스무고개 정답을 맞출까	24
이수경 강중호 제프리 힌튼의 캡슐망을 풀이하다	30
이주영 노명철 ICCV 2017 참관기	34
AI & media art	
예술에 인공지능을 묻다	
송호준 예술이 AI를 바라보는 시선	40
최승준 "X의 목적은 통찰이지 Y가 아니다"	44
exercise	
슈퍼마리오 그리고 GAN : 두 번째 이야기	
송호연 강화학습으로 풀어보는 슈퍼마리오 part2.	56
유재준 Do you know GAN? (2/2)	64
information	
석/박사생을 위한 카카오의 상시 연구지원 프로그램	70
closing	72

카카오 AI 리포트

8호를 내며

8번째로 발행되는 카카오 AI 리포트에서 주목한 주제는 카카오의 기계번역기였습니다.

카카오의 번역기는 지난 9월 시범 적용된 이후, 소리없이 강한 모습으로 번역기 시장에

자리매김하고 있습니다. 이러한 이유로 카카오 번역기 개발 주역들에게 기계번역 기술의

역사, 최근 기계번역의 핵심 기술로 일컬어지는 신경망 번역 모델의 진화 과정, 그리고

카카오 번역기의 경쟁력을 만들어 낸 기술에 대한 소개 글을 부탁드립니다. 카카오

번역기에 관심이 있는 분이라면 양질의 대규모 학습 데이터 확보 기술을 소개한 오형석 님의

글에 흥미를 느끼실 것으로 예상됩니다.

번역기에 대한 세 편의 글을 보신 이후에는 AI 기술의 구현체인 카카오톡니 속

유쾌한 기능에 대한 소개글이 이어집니다. 많은 카카오톡니 이용자들이 신기해 하는 스무고개

게임 제작기입니다. 조희주 님과 김수형 님의 글은 소재 만큼이나 문체도 유쾌합니다.

카카오브레인의 이수경 님과 강종호 님은 최근 AI에서 가장 주목받는 논문 중 하나인 제프리

힌트 교수의 캡슐망(capsule network) 논문 내용과 의미를 설명해 주셨습니다.

AI는 산업 뿐만 아니라, 예술에도 영향을 미치고 있습니다. 두 분의 미디어

작가(최승준 작가와 송호준 작가)가 정리해 주신 글은 예술이 바라보는 AI에 대한 관점을

이해하고, 향후 AI와 시각 예술의 융화 방향을 이해하는데 요긴한 참고 자료가 될 것으로

생각됩니다. 조금은 긴 글이지만, 시간을 두고 읽어 보시길 권해 드립니다. 송호준 작가님은

인공지능 관련 공연과 발표를 위해 인도 뉴델리에서 머무르며 글을 보내오셨습니다.

강화학습을 슈퍼마리오 게임에 적용한 송호연 님의 두 번째 글, GAN의 개념을

설명한 유재준 님의 두 번째 글 또한 이번 호에서 만나보실 수 있습니다. 카카오 AI부문의

이주영 님과 노명철 님이 지난 10월 열린 국제컴퓨터비전학회(ICCV)를 다녀와서 쓰신

참관기도 실려있습니다.

아울러, 카카오가 AI를 연구하는 석사와 박사 과정 학생을 위해 마련한 연구

지원 프로그램 정보 역시 리포트 말미에 정리해서 넣었습니다. 이번 호가 귀한 시간을

내주신 독자 분들께 가치있는 경험이 되길 고대합니다.

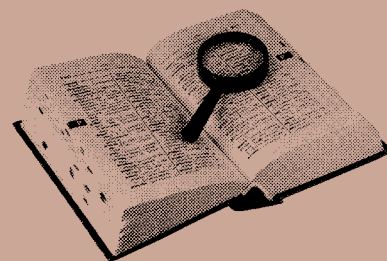
2017년 11월 30일

카카오 정책지원파트 드림

기계번역기의

A to Z, 그리고

카카오의



기계번역

AI in Kakao

김미훈 | 기계번역기의 역사와 발전

06

배재경 | 신경망 번역 모델의 진화 과정

10

오형석 | 카카오 번역기가 양질의 대규모 학습 데이터를 확보하는 방법

16

인공지능(AI) 기술이 우리 눈 앞에 구현되고 있는 대표적 형태 중 하나가 기계번역입니다. 이번 호에서는 기계번역이 발전되어 온 역사, 그리고 현재 기계번역 기술의 급성장을 가져온 신경망 번역 모델이 진화해 온 양상에 대해 설명한 글을 담았습니다. 카카오 번역기가 경쟁력을 높이기 위한 목적에서 양질의 대규모 학습 데이터 확보를 위해 사용한 기술에 대한 설명글도 준비했습니다.

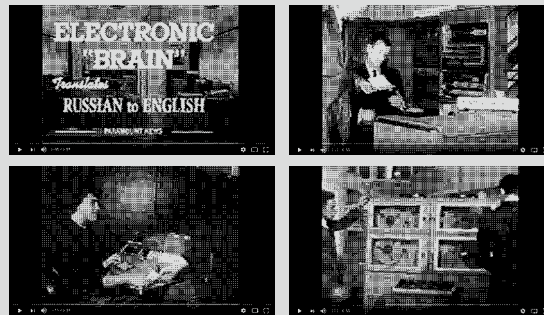
기계번역기의 역사와 발전

신경망 기반의 기계번역(neural machine translation, NMT)은 메이저급 기계번역 서비스에 속속 도입되고 있다. 딥러닝과 방대한 병렬 말뭉치를 핵심 기술로 삼고 있는 기계번역기는 번역 전공자들의 위기감까지 유발할 정도의 높은 성능을 자랑한다. 이러한 시스템의 등장은 지난 70여년 간의 '번역'이라는 작업(task)에 대한 깊은 이해와 고민, 여러 차례의 시행 착오가 없었다면 불가능했을 것이다. 기계번역기의 발전 연대기는 기술의 발전 양상에 따라 (1) 1960년대 중반, (2) 1960년대 중반에서 1990년대 중반, (3) 1990년대 중반 이후로 구분될 수 있다.

기계번역기의 발전 연대기

기계번역이라는 용어는 1949년 워렌 위버(Warren Weaver)의 번역(translation)*에 처음 언급 되었다. 이 논문은 당시 큰 주목을 받았고, 기계번역 연구의 촉매 역할을 한 것으로 평가된다. 1951년 MIT에서 기계번역 연구가 본격적으로 시작됐다. 이후, 일본, 러시아, 프랑스 등의 국가에서도 기계번역 관련 연구가 활발히 진행됐다. 다음의 유튜브 영상*2을 보면 1950년대 기계번역에 대한 기대 정도를 엿볼 수 있다.

[그림 1] 1950년대 기계번역



5년이면 기계번역을 정복할 수 있을 것이라는 기대와 달리, 1960년대 중반에 와서도 기계번역 성능은 기대에 미치지 못했다. 기대는 실망으로 바뀌었고, 기계번역에 대한 회의론이 일기 시작했다. 기계번역 연구도 이전처럼 활발하지 못했다. 관련 연구는 여러 대학 연구실에서만 주로 이루어졌다. 1970년대와 1980년대를 거치면서 여러 방법론들이 제시되었지만 일부 분야(domain)에서만 그 성능을 인정받는 수준이었다. 하지만 이 시대의 기계번역 연구는 관련 자연어처리 기초 연구 분야들(형태소 분석, 구문분석, 언어생성 등)의 발전에 많은 영향을 주었다.

[그림 2] 1970년대 기계번역



1990년 전후로 통계적 방법을 기계번역에 접근한 통계 기반 기계번역(statistical machine translation, SMT)은 혁신적인 변화를 일으켰다. 원문과 번역문이 함께 있는 병렬 말뭉치에 통계적으로

접근하는 방법론이다. 다수의 기업들이 이러한 방법으로 기계번역 개발에 적극 뛰어들기 시작했다.

이후 딥러닝 기반 방법론들이 이미지 처리, 음성인식, 자연어 처리 기반 기술로 사용되면서 기계번역도 좋은 성능으로 주목받게 되었다. 번역 관련 서비스들이 폭발적으로 늘어남에 따라 구글(Google), 마이크로소프트(Microsoft), 페이스북(Facebook) 뿐만 아니라 중국의 바이두(Baidu), 러시아의 안덱스(Yandex) 등 현재 각 나라의 주요 포털 업체들의 경우 모두 자체 플랫폼에서 번역 서비스를 제공하고 있다. 이외에도 시스트란(SYSTRAN)과 같은 번역 서비스 회사들까지 활발하게 사업을 펼치고 있는 중이다. 그래서 지금은 가히 기계번역기의 춘추전국시대라고 해도 과언이 아니다.

아울러 해외 여행의 급증, '직구' 등을 통한 해외 쇼핑 경험 확대, 비즈니스 교류 증가 등으로 통번역 수요가 높아지고 있다. 모바일 플랫폼 확장으로 인해 높아진 번역 서비스에 대한 접근성도 기계번역에 대한 필요를 키우고 있다. 이와 같은 번역 서비스에 대한 수요의 지속적 증가는 기계번역기를 둘러싼 치열한 경쟁의 형성에 한 몫을 하고 있다. 이같은 환경 속에서 주요 IT 업체들은 자사의 기계번역 기술을 바탕으로 사용자에게 글의 맥락(context)에 최적화된 번역 서비스를 제공하려고 노력하고 있다. 아울러 적용 폭 역시 확대 중에 있다. 시장조사 기관인 그랜드 뷰 리서치(Grand View Research)에 의하면, 기계번역 시장은 2022년 9억 8330만 달러 규모(1조 747억 원*)로 성장이 예상된다. 특히 방대한 양의 콘텐츠를 정확하고 빠르게 번역할 필요가 있는 전자, 자동차, 의료, 밀리터리 업계 등에서 기계번역의 높은 미래 가치가 점쳐지고 있다*4.

구글, MS, 바이두의 기계번역

현재 전 세계적으로 가장 많이 이용되고 있는 기계번역 서비스는 구글 번역이다. 구글 번역은 103개 언어의 번역을 지원하고 있는데, 이는 세계 최대 규모다. 구글 번역을 이용하는 사람은 전 세계 5억 명 이상이며, 매일 1400억 개 이상의 단어 번역이 이루어진다.

구글은 2006년 통계 기반 기계번역 서비스를 출시한 후, 다음 해 모든 번역 엔진을 SMT로 전환하였고, 2016년 10월 GNMT(Google's neural machine translation)를 발표하며 이를 서비스에 적용하기 시작했다. 또한 구글은 증강현실 애플리케이션 서비스 업체인 '퀘스트 비주얼(Quest Visual)'을 인수해 이미지 번역을 시작했다. 최근에는 40개 언어를 자동으로 번역해주는 구글 어시스턴트(Google Assistant)가 내장된 무선 헤드셋 '픽셀 버즈(Pixel Buds)'를 출시하며 하드웨어 분야로의 진출을 본격적으로 알리고 있다.

MS는 2016년 11월에 기계번역 시장 경쟁에 뛰어들었다.

글 | 김미훈 may.jin@kakaocorp.com

현재 14개월된 아기 엄마로서 인간지능의 발달과정과 인공지능간 차이의 분명함을 매일 경험하면서 인간을 모형으로 하고 있는 인공지능에게 '진정한 지능'을 만들려면 어떻게 해야 할까 항상 고민하고 있습니다. 밖에서 보여지는 인공지능 신화와 달리 아직은 할 일이 너무 많다고 생각하는 현업 종사자로서 매일매일 최선을 다할 뿐입니다.

기술문서 번역의 강점 및 엔터프라이즈(enterprise)^{*5} 서비스 경험을 기반으로 한, 발표자의 설명을 실시간으로 번역하는 프레젠테이션 번역, 스카이프(Skype)를 통한 실시간 통번역 서비스 등을 제공하며 기업 시장을 공략하고 있다.

중국의 구글이라 불리는 바이두는 정부의 적극적인 지지에 힘입어 최근 인공지능(artificial intelligence, AI) 분야에서 가파른 성장세를 보이고 있다. 지난 2년간 바이두가 AI 분야에 투자한 금액은 200억 위안(약 3조 6464억원)^{*6}에 육박한다. 바이두는 강력한 데이터베이스를 기반으로 현재 27개 언어에 대한 번역 서비스를 제공하고 있으며 일부 언어에 NMT를 적용하고 있다.

국내의 기계번역 시장 역시 IT 기업들이 개발을 주도하고 있다. 카카오는 지난 10월 통합 인공지능 플랫폼, 카카오(아이)의 번역 엔진을 적용한 기계번역 서비스 '번역 베타(beta)' 서비스를 선보였다. 한국어, 영어 이외의 추가적인 언어 서비스를 위해 준비 중이다. 네이버는 '파파고(papago)'라는 서비스를 운영 중이다. 2016년 8월부터 운영 중인 파파고에는 NMT 기술이 적용됐다.

높은 관심과는 별개로, 기계번역 역시 딥러닝 연구의 빙하기와 마찬가지로 인기가 없던 암흑기가 있었다. 인기 없는 연구임에도 불구하고 이 문제를 해결하고자 시간과 심혈을 기울였던 모든 이와 연구를 지원해준 관련 정부 부처 연구비 집행 담당자분들의 안목이 있었기에 오늘 이러한 번역기를 만들 수 있다고 생각되기에 그 모두에게 감사드린다.

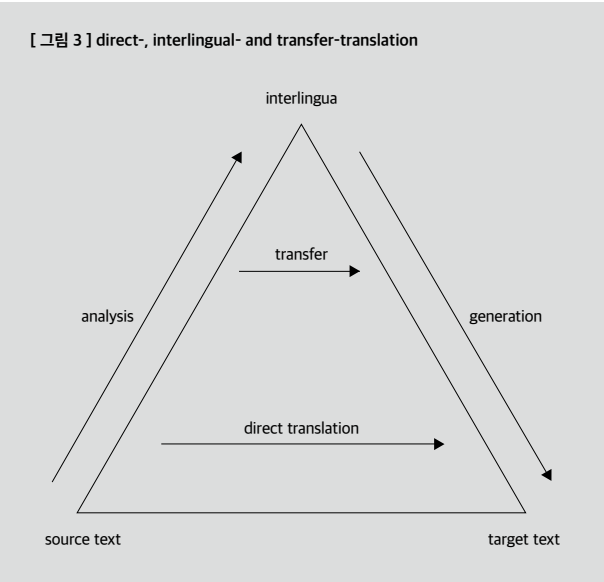
번역 기술의 개괄적 구조

번역 기술을 [그림 3]과 같은 형태로 표현해볼 수 있다. 번역하려는 언어(source language)와 번역언어(target language)를 삼각형 아래 두 꼭지로 표현할때, 번역 과정을 크게 3가지 경로에 따라 direct, interlingual 그리고 transfer방식으로 표현할 수 있다.

첫번째 경로가 보여주는 것은 소스 언어(source language)에서 타겟 언어(target language)로 direct-translation이라고도 하는데, word-to-word 번역 방식이다. 초창기에는 이 방식이 한국어와 일본어처럼 서로 유사한 언어를 번역하는 기계에 사용되기도 했다. 이 방법론의 장점은 간단함에 있지만, 단점은 토폴로지(topology)가 다른 언어쌍의 번역을 진행하기에는 부족하다는 것이다.

Source-target 언어 쌍의 차이를 극복하기 위한 방법으로 학자들은 source 문장들의 의미를 추상화하여 원래 언어와 독립된 매체 또는 구조로 표현하고, 이 추상화된 의미를 다시 target 언어로 생성하는 방식으로 접근했다. Source 언어 문장을 interlingua로 표현하고 다음으로 interlingua에서 target언어 문장을 생성 할 경우, 우리는 이를 interlingual 방식으로 부른다.

반면 transfer는 direct-translation과 interlingual-translation의 중간 정도 abstraction으로 생각할 수 있는데, 예로 syntactic transfer에서는 source 단 문장들을 (의존)구문분석^{*7}을 통하여 구문트리로 표현하고, 구문트리는 변환과정을 거쳐서 target 번역문을 만드는 과정이다.



1990년대 부터 말뭉치를 이용한 통계적 접근 방식이 자연어처리 등 여러 task에 적용되면서, 병렬 말뭉치를 이용한 통계 기반 기계번역 방법론들이 통용되기 시작했다. 통계 기반 기계번역은 translation model(이하, TM)과 language model(이하, LM)로 이루어져 있다. TM에서는 source와 target 문장 사이에 각 단어나 구가 어떻게 번역으로 매치되는지를, LM에서는 번역문의 각 단어들이 얼마나 문장다운 문장을 만드는지 수치화하여 보여준다. 이 두개 요소가 함께 번역하려는 문장에 딱 알맞는 번역문을 만드는 것을 제어한다.

NMT는 SMT의 범주로 볼 수 있는데, 신경망을 기반으로 TM이나 LM의 수치들을 얻는다는 점이 SMT와의 차이점이다. 최초 NMT^{*8} 모델의 중심은 LM이었고 그 이후 제안된 모델^{*9}에는 attention이라는 mechanism으로 source와 target문장의 대응된 단어들이나 구들을 매치해준다. 즉 기존 통계 기반 기계번역의 TM부분을 더욱 명확하게 모델링한 방법론이다. 현존하는 고성능 기계번역 서비스는 거의 대부분 이 모델을 기초로 하고 있고 이러한 NMT 모델은 전통적인 통계 기반 기계번역의 모델링 방법과 큰틀에서 다르지 않으므로 현재 NMT는 기존 연구들의 연속이라고 생각한다.

최근에는 attention mechanism을 더욱 적극적으로 사용하는 연구가 활발히 이루어지고 있다. 기존의 방식이 source 문장 구성 요소와 이에 대응되는 target 문장 구성 요소 사이의 관계를 찾는 것에 한정되는 Inter-attention 방식이었다면, 최근 모델은 독립적으로 source, target 각 문장의 요소 내부에서 관계를 찾는 Intra-attention 까지 활용한다. 이 방식은 각 구성 성분들이 담고 있는 ambiguity의 해소에 도움을 주어 결국 번역 성능 향상으로 이어지고 점점 전통적인 신경망 모델들을 대체하고 있다^{*10}.

NMT에 관련해서는 따로 지면을 할애하여 설명하겠지만,

기계번역기 모델링으로 sequence-to-sequence 모델링 방법은 자연어처리를 비롯한 여러 작업(task) 뿐만 아니라, 시퀀스(sequence)를 다루는 문제들을 해결하는 한 예시의 역할을 하고 있어, 그 의미가 더욱 크다.

^{*1} 참고 | <http://www.mt-archive.info/Weaver-1949.pdf> ^{*2} 참고 | <https://youtu.be/K-HfpsHPmw> ^{*3} 참고 | 1달러=1,093원 기준 ^{*4} 참고 | <http://www.grandviewresearch.com/industry-analysis/machine-translation-market> ^{*5} 참고 | 기업 내부의 일상적 활동 수행이 이뤄질 수 있도록 하는 기업 네트워크 시스템 ^{*6} 참고 | 1위안=168.23원 기준 ^{*7} 참고 | https://ko.wikipedia.org/wiki/구문_분석 ^{*8} 논문 | Sutskever, I. et al. (2014). Sequence to Sequence Learning with Neural Networks, NIPS ^{*9} 논문 | Bahdanau, D. et al. (2014). Neural machine translation by jointly learning to align and translate, ICLR ^{*10} 논문 | Vswani, A. et al. (2017). Attention Is All You Need, doi : arXiv:1706.03762

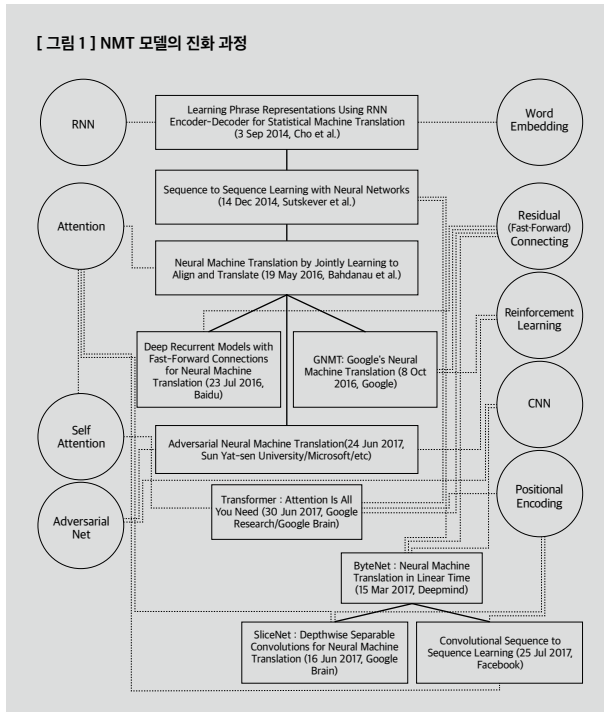
신경망 번역 모델의 진화 과정

통계 기반 번역기가 end-to-end 방식의 신경망 기반 기계번역(neural machine translation, NMT)으로 바뀌고 실제 서비스에 적용되기 시작한 것은 불과 1년 전의 일이다. 인공지능(artificial intelligence, AI)은 'AI winter' 시기를 견뎌낸 후, GPU 성능의 성장과 함께 꽃을 피웠다. 다양한 문제에 적합한 뉴럴 네트워크들이 나왔고, 단순한 뉴럴 네트워크들이 다양한 방식으로 연결되어 더 크고 복잡한 구조를 만들어 내고 있다.

네트워크가 더 크고 복잡해지는 이유는 각 단위 뉴럴 네트워크들이 어려운 문제를 해결하는데 상보적으로 작용하기 때문이다. 이는 마치 생명체가 세포들이 모여서 기관을 이루고, 기관들이 모여 온전한 개체로 완성되는 것과 유사하다. NMT도 가장 전형적인 복합 구조를 가지는 뉴럴 네트워크 중 하나이며 이미지넷(ImageNet)의 영상 인식 기술과 유사하게 매우 짧은 시기에 다양한 진화 단계를 겪어왔다. 물론 지금도 계속 진화중이다. 이번 글에서는 번역에 사용되는 모델의 진화 과정을 통해 각 모델의 핵심 구조와 아이디어들이 어떻게 발전해 왔는지 살펴보도록 하겠다.

NMT 모델의 진화 과정

지금까지 이루어진 NMT 연구 결과를 한 장의 도표로 표현하는 것은 쉽지 않다. 그렇지만 주요 기반 뉴럴 네트워크 및 모델들 간의 상관 관계를 시간 축으로 그려 보면 많은 정보를 얻을 수 있다.

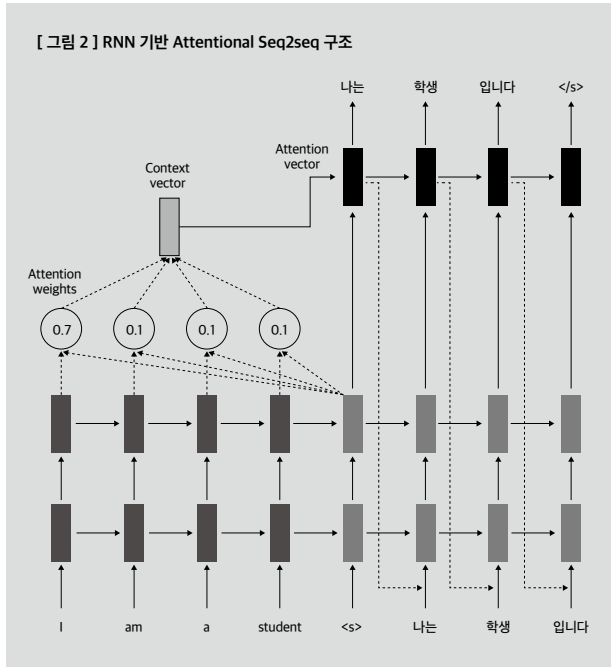


[그림 1] 에서 원은 주요 기반 모듈이고, 사각형은 NMT 모델이다. 각 모델은 기존 모델을 토대로 만들어지기도 하고 어느 정도 독립적으로 생성되기도 하는데, 그림에서 굵은 실선은 강한 영향 관계 또는 기존 모델을 토대로 했다는 의미이고 얇은 점선은 약한 영향 관계 또는 내부 모듈로 사용된 경우를 표현하고 있다.

첫번째 NMT 모델은 2014년 12월에 발표된 'Sequence to sequence learning with neural networks' 이다¹. 하지만 이 모델은 인코더-디코더(encoder-decoder)모델을 토대로 확장된 것이기 때문에 'Learning phrase representations using RNN Encoder-Decoder for statistical machine translation' 을 NMT의 시발점이라고 봐도 될듯 하다². Encoder-decoder 모델은 단위 정보(word 또는 token)의 시퀀스(sequence)를 입력값으로 받아서 고정 길이 vector representation을 생성한 후 이를 이용하여 또 다른 단위 정보의 sequence를 생성하는 모델이다. Sequence를 주로 다루기 때문에 최근에는 encoder-decoder 대신 sequence-sequence(seq2seq) 라는 용어를 많이 쓰고 있다. Sutskever의 모델에서 encoder와 decoder 각각은 RNN(recurrent neural network)으로 구현되며 단위 정보는 word embedding 을 통해 continuous value로 변환되어 사용된다.

Seq2seq 모델을 NMT에서만 사용하는 것은 아니다.

Sequence 형태로 표현될 수 있는 정보를 다루는 어떤 곳이든 사용 가능한데, 예를 들어, 문서 요약, QA(Question Answering), Dialog 등이 모두 포함된다. 따라서 NMT의 구조를 파악하는 것은 자연어를 다루는 많은 문제들, 특히 문맥(context) 정보를 파악해야 하는 과제를 풀어나가는 가장 좋은 출발점이라고 볼 수도 있다.



가장 단순한 형태의 seq2seq 모델은 성능이 그렇게 만족스럽지는 못했고 뉴럴 네트워크의 가능성을 확인한 정도였다. 기존의 가장 좋은 통계 기반 기계번역(statistical machine translation, SMT)과 경쟁할 만한 성과를 보여준 모델은 'Neural machine translation by jointly learning to align and translate' 이라 할 수 있다. 이 모델은 attention net을 활용하는 좀 더 복잡한 decoder를 사용한다³.

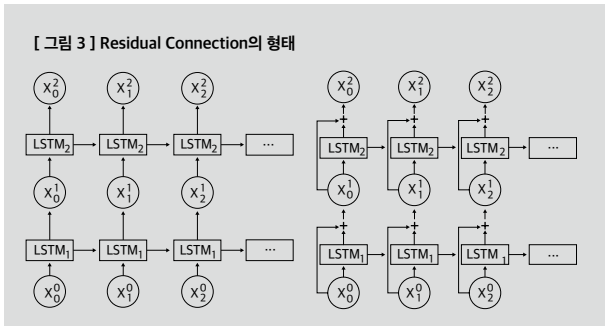
Attentional decoder는 decoding 시 매 time-step 별로 새로 생성될 토큰을 결정할 때 source sequence에서 가장 가까운 관계의 token을 결정한 후 이 정보를 활용하는 구조이다. 마치 두가지 언어를 구사할 수 있는 사람이 번역을 할 때 단어 별로 원문과 번역문을 매칭해 가면서 번역하는 것과 유사하다.

Attention을 도입함으로써 encoder의 결과를 고정 길이 벡터(vector)에 담아야 하는 문제도 해소되었다. 짧은 문장에 비해 긴 문장의 경우 더 많은 정보가 함축될 수밖에 없는데, 길이에 상관없이 고정 길이 벡터를 사용하는 것은 불합리하다는 것이다. 이 모델에서는 encoder의 매 time-step 시에 생성되는 벡터가 attention에 사용되므로 sequence 길이에 비례하여 더 많은 정보가 활용된다. 논문에서는 장문 번역의 성능이 높아진 결과를 attention 도입의 효과로 서술하고 있다.

Encoder, decoder의 각 RNN은 동일 구조가 반복적으로 쌓인 구조(multi-RNN)인데, 이렇게 할 경우 단일 layer에 비하여 좀 더 복잡하고 다양한 특징(feature)을 추출할 수 있다. RNN의 각 셀은 LSTM(long short-term memory) 또는 GRU(gated recurrent unit)를 사용한다. 논문에서는 추가적인 아이디어로 bidirectional RNN encoder를 제안하고 있는데, 이는 양방향의 이력(history) 정보를 모두 활용하여 놓치는 정보를 최소화하려는 의도이다. [그림 2]는 이 모델의 구조를 보여준다. 이 그림에서는 복잡도를 줄이기 위하여 unidirectional RNN을 가정하였다.

드즈미트리 바다나우(Dzmitry Bahdanau)의 논문 이전에도 여러 기업에서 NMT 연구가 활발히 이루어졌지만, 이 논문을 계기로 좀 더 적극적으로 바뀌었다. 특히 구글(Google)과 바이두(Baidu)의 물밑 경쟁은 눈여겨 볼만하다. 먼저 바이두에서 추가적인 아이디어를 통해 번역 성능을 높인 논문(Deep recurrent models with fast-forward connections for neural machine translation)을 발표했다⁴. 핵심적인 내용은 fast-forward connection(구글에서는 residual connection으로 명명)의 도입인데 기존 encoder/decoder에서 Multi-RNN을 사용할 때 layer가 3~4개 이상인 경우에 학습이 잘 안되던 문제를, fast-forward connection을 통해 8개 layer 이상도 학습이 가능하도록 만든 것이다. Layer를 깊게 가져 가려는 이유는 더 풍부한 특징을 추출하여 성능을 높이기 위함인데, 너무 깊을 경우 기울기 값이 소실되는 문제(gradient vanishing)로 학습이 안되는 경우가 있었다. Fast-forward connection을 통해 이 문제를 해소하였고, 그 결과 드디어 번역 성능이 SMT를 능가하게 되었다. Fast-forward connection은 복잡한 뉴럴 네트워크가 아니라 n번째 layer의 입력이 n+1 번째 layer의 입력에 같이 들어가도록 추가 connection을 하나 두는 방식인데 CNN에서 먼저 사용되어 효과를 본 것을 RNN에도 유사하게 적용한 것이다.

아래 그림을 보면 단순 RNN과 fast-forward(residual) connection 이 있는 RNN의 차이를 확인할 수 있다.



구글이 NMT 핵심 연구 분야에서 앞서가고 있었음에도 불구하고, 바이두가 한발 먼저 최고 성능의 번역 모델을 발표한 상황이

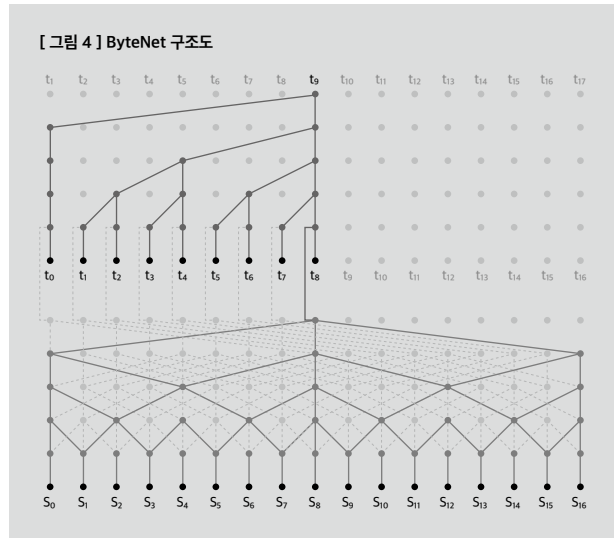
되었고, 이에 대한 대응으로 구글은 빠르게 실제 서비스를 런칭하는 방향으로 전략을 수정하였다. 논문으로 실험 결과를 발표하는 것과 실제 서비스화 하는 것에는 큰 차이가 있는데, 학습 시간도 이슈지만 응답 속도 및 처리량에 대한 고려를 하기 위해서는 많은 추가 작업이 필요하기 때문이다.

구글은 당시 유효하다고 판단했던 여러가지 기반 뉴럴 네트워크를 NMT 모델에 적용하였고, 특히 빠른 학습 속도를 위하여 하드웨어에 최적화된 model parallelism을 구현하였다. 빠른 응답 속도를 위해 양자화(quantization)를 도입하였고, 번역 품질을 극대화 하기 위해 길이 정규화(length normalization), coverage penalty 등의 몇가지 예측 알고리즘(prediction algorithm)을 도입하였다. 전략은 성공했고 필자도 당시 GNMT 논문을 처음 접했을 때 공학적인 측면에서 감탄하지 않을 수 없었다. 구글에서 발표한 'Google's neural machine translation system'이라는 논문은 'Bridging the gap between human and machine translation'이라는 부제를 가지고 있는데 그만큼 성능에 자신이 있었기 때문이었을 것이다⁵.

하지만 어느 정도 시간이 지난 후 필자는 GNMT 모델의 구조가 당시 하드웨어 스펙에 맞추느라 다소 부자연스러운 점이 있다고 생각했고, 이 구조가 많은 알고리즘을 조합해 낸 결과이기 때문에 장기적인 개선 작업 또한 쉽지는 않을 것이라고 판단했다. 아니나 다를까 구글은 RNN 기반이 아닌 새로운 구조의 모델을 조만간 발표하게 된다. 이들에 대해서는 잠시 후에 다룰 예정이다. 사실 그 사이 네이버가 구글보다 먼저 NMT를 번역 서비스에 적용했는데 발빠른 행보가 참으로 돋보였다. 그렇지만 모델이 공개되지 않아 그 구조를 파악할 수 없었고 글자 수 200자 제한을 꽤 오랜 기간 유지한 것도 아쉬운 부분이었다. 네이버, 구글 이외에도 여러 업체에서 NMT 기반 번역 서비스를 시작하게 되면서 번역은 비전(vision) 분야와 함께 딥러닝의 주요 화두가 되었고 2017년 초부터 또다른 연구 성과들이 경쟁적으로 공개되었다. 그 중 가장 주목할 만한 것은 CNN 기반의 모델인 ByteNet이다⁶.

딥마인드에서 개발한 이 모델은 논문의 제목이 'Neural machine translation in linear time'인 것에서 알수 있듯이 학습 시간을 선형 시간(linear time)에 가능하게 하는 것이 목적이다. 지금까지의 NMT 모델은 RNN 기반의 attentional seq2seq를 거의 정석처럼 사용했는데 attention net 때문에 학습 시간이 quadratic time(source sequence size * target sequence size)을 가지게 된다. 반면, ByteNet에서는 attention net을 사용하지 않으므로 linear time($c * \text{source sequence size} + c * \text{target sequence size}$)에 학습이 가능하다(여기서 c 는 constant value). ByteNet은 [그림 4]와 같이 CNN을 사용하여 encoder 위에 decoder가

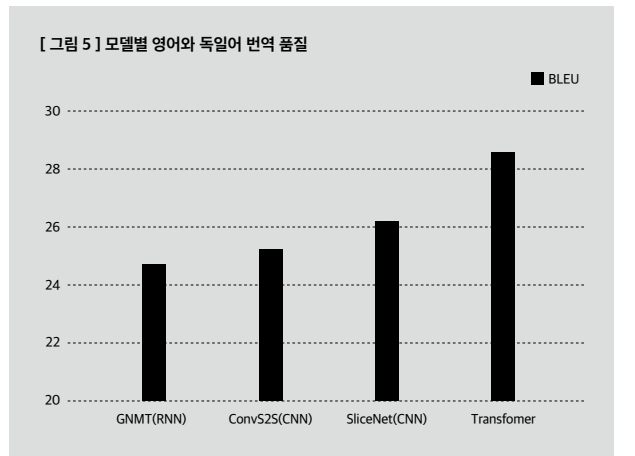
스택처럼 쌓이는 네트워크 구조를 만들고 dynamic unfolding이라는 기법을 통해 가변 길이 sequence를 생성해 낸다.



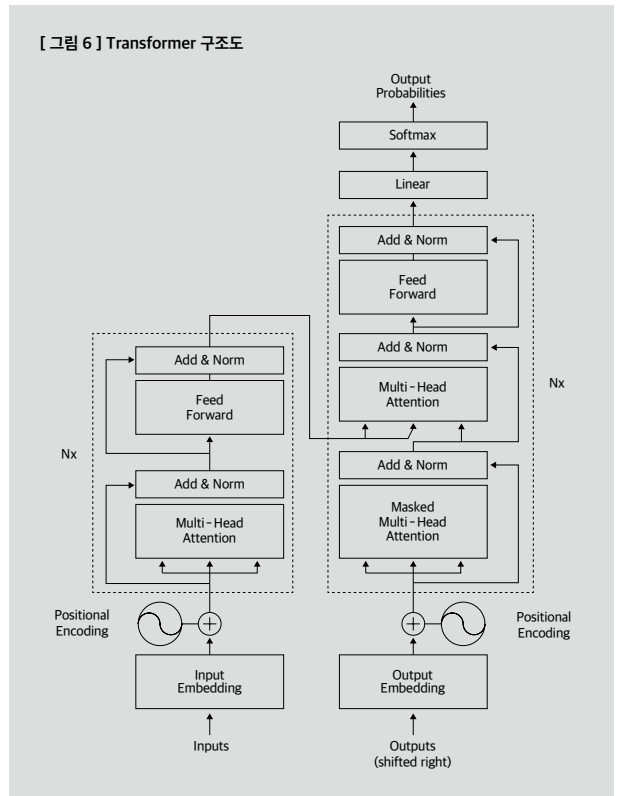
RNN에서는 필수적인 time-step과 step 간 정보의 기억(memorization)이 필요없게 되는데, 구조적인 특성상 병렬화의 여지가 훨씬 크고 멀리 떨어진 단위 정보 사이의 관계 특성(feature)을 더 잘 찾아낼 수 있다. 이 모델은 성능에 있어 기존 RNN 모델과 비교할 수준은 아니었지만, character to character 번역(단위 정보로 워드나 토큰이 아니라 character를 사용)에서는 최고의 성능을 보여주었다.

ByteNet 처럼 CNN 기반의 NMT 모델도 짧은 기간 동안 연구가 활발히 이루어져 RNN 기반 모델의 성능을 능가하는 모델이 나오기 시작했다. 그 중 두드러진 두가지 모델은 페이스북에서 공개한 Convolutional Sequence to Sequence Learning과 구글브레인에서 공개한 SliceNet이다^{7,8}. 두 모델 모두 convolution net을 사용하고 positional encoding 과 attention net을 적용하였다. SliceNet이 약간 먼저 나오긴 했지만 convolution net의 구조와 attention net의 적용 형태가 약간 다를 뿐 서로 상당히 유사하다. 결국 attention net이 RNN에 적용되어 극적인 성능 향상이 이루어진 것처럼 CNN 방식에서도 유사한 과정이 진행되었다고 볼수 있다. CNN에서는 time-step이 없으므로 단위 정보의 위치 정보를 표현하기 위한 다른 방법이 필요한데 이를 위해 positional encoding을 사용한다.

[그림 5]는 주요 모델들의 성능을 비교한 것이다⁹. 그동안 자연어 텍스트 처리에는 RNN이 적합하다는 관점이 우세했지만 이를 뒤엎는 결과가 나온 것을 확인할 수 있다. 필자는 이 결과를 보고 RNN의 시대가 벌써 저무는것 아닌가 하고 생각했는데 과연 그럴지는 두고 볼 일이다.



비슷한 시기에 RNN, CNN 기반 모델 이외에 압도적인 성능을 보여준 모델이 하나 더 공개되었다. Transformer라 불리는 이 모델은 부제가 'Attention is all you need' 인데 RNN, CNN 모두 필요 없다는 말이다¹⁰. 뉴럴 네트워크는 attention net과 normalization, feed-forward net의 반복적인 구조로 이루어진 매우 매우 단순한 형태이다. 대신 attention net이 기존에 decoder의 sequence와 encoder의 sequence 간에 align을 맞춰주는 용도로 사용되었다면, 여기서는 추가로 encoder/decoder 각 layer의 입력 정보를 함축하는데 사용되는 방식으로 확장되었다. 이 때문에 Transformer에서 추가로 적용된 attention 방식을 self-attention이라고 부르기도 한다. [그림 6]은 Transformer의 구조를 보여준다.



전형적인 encoder-decoder 모델의 동작 방식을 다시 되새겨 보자. Encoder가 input 정보를 vector representation으로 함축하고 decoder에서 이 정보를 바탕으로 최종 output을 생성하는데 지금까지의 모델은 input 정보의 함축을 위해 RNN이나 CNN을 사용했던 반면, Transformer에서는 단위 정보 각각의 상관 관계를 attention net 구조로 풀어내면서 정보를 함축한다. 이러한 구조만으로도 feature 정보를 충분히 잘 추출해내어 RNN과 CNN 보다 오히려 더 나은 성능을 보여준 점은 새로운 발전이 고정 관념을 깨는 것으로부터 출발하다는 좋은 실례를 보여주는 것이라 할 수 있다.

결국 RNN/CNN에 이어 self-attention 이라는 기반 뉴럴 네트워크가 가세하면서, encoder-decoder라는 큰 구조를 제외하면 이에 대한 구현체들은 얼마든지 다양한 방식으로 결정될 수 있다는 생각이 더 자연스럽게 이루어지게 되었다. Seq2seq는 RNN2RNN, RNN2CNN, CNN2CNN 뿐만 아니라 Any2Any로 고민될 수 있다. 사실 개발자 입장에서는 선택의 폭이 넓어진 것이 썩 달갑지 만은 않다. 특정 문제에 어떤 뉴럴 네트워크를 사용하는 것이 적합한지에 대한 명확한 근거가 없는 경우가 대부분이고 따라서 대부분 실험적으로 접근할 수밖에 없는데, 실험은 수많은 hyperparameter 최적화 과정뿐만 아니라 기반 뉴럴 네트워크의 다양한 조합들이 모두 고려되어야하기 때문에 부담이 가중된다고 볼 수 있다. 그렇지만 좀 더 큰 틀에서 본다면 특징 추출을 위한 가장 적합한 구조의 뉴럴 네트워크들이 다양하게 나오고 많은 연구자들에 의해 이들의 장단점, 특징들이 파악되어 감에 따라 AI가 완전한 black box에서 어느 정도 투명하고 제어 가능한 모습을 가지게 될 것이란 기대를 해본다. 그리고 추가로 AutoML이라는 분야의 연구도 활발히 진행되고 있는데 이는 뉴럴 네트워크 선택과 hyperparameter 튜닝의 자동화에 대한 연구로 AI 개발자들의 가려운 곳을 많이 긁어줄 수 있을 것으로 기대된다.

어떻든 self-attention net의 성능에 대해서는 어느 정도 증명이 되었다고 볼 수 있고, 그렇다면 이를 능가하는 새로운 기반 뉴럴 네트워크가 또 나올 것인지를 예측해 보는 것도 흥미로운 것 같아서 개인적인 의견을 달아 본다.

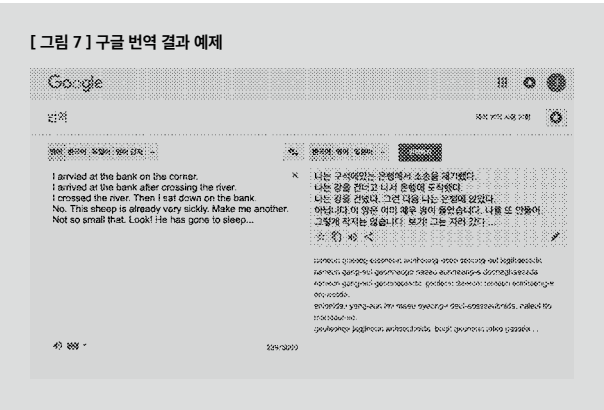
어떤 지도 학습(supervised learning) 모델(모델 A)의 power를 측정하기 위한 간단한 접근 방식 중 하나는 샘플을 충분히 많이 확보하여 기존 모델(모델 B)에 적용한 후 그 결과로 만들어진 데이터를 학습셋으로 이용하여 모델 A를 다시 학습하고 그 성능이 모델 B와 유사한지 아닌지 보는 것이다. 만일 성능이 서로 유사하다면 모델 A는 모델 B 보다는 약하지 않다고 판단할 수 있다. 그리고 위의 과정을 역으로 진행했을 때 성능이 유사하지 않다면 모델 A가 모델 B보다 나은 성능을 가진다고 말할 수 있을 것이다.

Transformer는 거의 복사기 수준으로 기존 모델을 모사해 낸다. 이런 측면에서 Transformer는 충분히 강력하다고 판단되고 이를 월등히 능가하는 모델은 쉽게 나오기 힘들 것이라는 예상을 해본다. 모델 간의 경쟁 과정은 거의 수렴 단계로 보이며 따라서 번역 관련해서는 전혀 새로운 형태의 강자가 나타나기 보다는 기존 뉴럴 네트워크를 토대로 더 넓은 문맥을 다루는 모델로 진화해 나가지 않을까 생각된다.

결론으로 넘어가기 전에 'Adversarial neural machine translation'에 대해서도 살짝 언급해야 할 것 같다¹⁾. CNN 기반 모델들의 연구가 활발히 이루어지고 있는 동안 전혀 다른 학습 방식을 사용하는 접근도 이루어졌다. 지금도 여전히 활발한 연구가 이루어지고 있는 GAN(Generative Adversarial Networks)의 접근 방식을 NMT에 유사하게 적용한 모델인데, 여기서는 사람의 번역과의 유사도를 극대화하는 기존 학습 방식 대신 NMT 모델과, 이와 사람의 번역을 구별해 내는 CNN 기반의 adversary net을 도입하여 둘 사이를 경쟁 관계로 두고 서로 발전해 나가는 모델이다. RNN 기반의 seq2seq 모델을 토대로 하긴 했지만, 새로운 학습 방법을 통해 기존 모델의 성능을 개선한 것이다.

이처럼 딥러닝의 많은 기본 아이디어들은 그 사용성이 일반적인 경우가 많다. 어떤 아이디어가 유효하다면 그 쓰임새가 특정 모델에 국한되지 않는다는 의미이다. 지금까지 살펴 보았던 attention net, residual connection, positional encoding 등도 모두 그러한 예이다.

NMT가 짧은 기간 동안 큰 발전을 이루어 왔지만 아직 갈 길이 멀다. 예를 들어 다음 구글의 번역 결과를 살펴보자.



[그림 7]의 예에서 'bank'라는 단어가 'river'와 같이 쓰일 때는 강둑이 더 적합하다. 또한 'Make me another.' 처럼 대명사가 들어간 문장을 제대로 번역해 내기도 힘들며, 구어체에서 많이 나타나는 짧은 어구나 문장들은 앞뒤 문맥을 더 넓게 봐야 정확한 번역이 이뤄질 수 있다.

카카오에서도 완전히 새로운 형태의 뉴럴 네트워크를

연구하기보다는 기존에 잘 동작하는 모델을 기반으로 문체나 더 넓은 문맥에 초점을 맞춰 모델을 연구 중이다. 해당 문장들을 카카오에서 실험 중인 모델로 다시 번역하면 다음과 같은 결과가 나온다.

I arrived at the bank on the corner.	나는 모퉁이에 있는 은행에 도착했다.
I arrived at the bank after crossing the river.	나는 강을 건너 후 강둑에 도착했다.
I crossed the river.	강을 건너 강둑에 앉았다.
Then I sat down on the bank.	
No. This sheep is already very sickly. Make me another.	아니, 이 양은 이미 병약해. 다른 양으로 만들어 줘.
Not so small that.	그렇게 작긴 않아요,
Look! He has gone to sleep...	보세요! 그는 잠들었어요.

짧은 예제라서 성급한 판단은 이르지만 가능성은 확인할 수 있다. 첫 두 문장의 'bank'의 번역은 RNN에 비하여 attention 기반 모델이 문맥을 더 잘 활용하는 예이고, 나머지 번역은 카카오에서 연구 중인 larger context 모델로 더 적절한 번역문을 만들어 낸 예제이다.

마치며

기계 번역이 인간의 수준을 따라잡기는 쉽지 않을 것으로 보인다. 언어는 수천 년간 독립적으로 형성된 문화를 반영하므로 언어간 1:1 매칭이 되지 않는 번역 규칙이 수없이 존재한다. 따라서 정확한 번역을 위해서는 문화를 이해해야 하고 이와 함께 역사/경제/과학/예술 등의 도메인 지식이 있어야 적절한 번역문을 생성해 낼 수 있다. 결국 사람은 단순히 텍스트 정보로만 번역하는 것이 아니라 수많은 추가 정보를 토대로 논리적인 유추의 과정을 거치면서 번역을 하게되는 것이다.

다행히 언어는 각 언어 별로 공통적인 규칙이 매우 많기 때문에 현재 기술로도 놀라운 성과를 내고 있다. 하지만 궁극의 번역 기술은 general AI 영역에 속한다고 볼 수 있다. 따라서 수년 내에 완벽한 번역을 해내는 AI 기술을 기대하는 것은 무리다. General AI도 언젠가는 탄생할 것이고 지구적 진화 과정의 시간 관점으로는 참나에 해당하겠지만 기껏해야 한 세기를 살수 있는 인간의 관점에서는 긴 시간일 거라 추측된다. 그렇지만 언젠가 될지 모르는 이 시점에 대한 두려움과 기대감이 교차하기도 한다.

¹⁾ 논문 | Cho, K. et al. (2014). Learning Phrase Representations Using RNN Encoder-Decoder for Statistical Machine Translation, doi : arXiv:1406.1078. ²⁾ 논문 | Sutskever, I. et al. (2014). Sequence to Sequence Learning with Neural Networks, doi : arXiv:1409.3215. ³⁾ 논문 | Bahdanau, D. et al. (2016). Neural Machine Translation by Jointly Learning to Align and Translate, doi : arXiv:1409.0473. ⁴⁾ 논문 | Zhou, J. et al. (2016). Deep Recurrent Models with Fast-Forward Connections for Neural Machine Translation, doi : arXiv:1606.04199. ⁵⁾ 논문 | Wu, Y. et al. (2016). Google's Neural Machine Translation: Bridging the Gap between Human and Machine Translation, doi : arXiv:1609.08144. ⁶⁾ 논문 | Kalchbrenner, N. et al. (2017). Neural Machine Translation in Linear Time, doi : arXiv:1706.03059, 2017. ⁷⁾ 논문 | Kaiser, L. et al. (2017). Depthwise Separable Convolutions for Neural Machine Translation, doi : arXiv:1706.03059. ⁸⁾ 논문 | Gehring, J. et al. (2017). Convolutional Sequence to Sequence Learning, doi : arXiv:1609.08144, 2017. ⁹⁾ 참고 | <https://research.googleblog.com/2017/08/transformer-novel-neural-network.html> - Google Research Blog. ¹⁰⁾ 논문 | Vaswani, A. et al. (2017). Attention Is All You Need, doi : arXiv:1706.03762, 2017. ¹¹⁾ 논문 | Wu, L. et al. (2017). Adversarial Neural Machine Translation, doi : arXiv:1704.06933.

카카오 번역기가 양질의 대규모 학습 데이터를 확보하는 방법

2017년 10월, 카카오가 번역 서비스를 출시했다. 신경망 번역 기술이 적용됐고, 블라인드 테스트 결과에 근거하면, "경쟁력이 높다"는 결과가 나왔다. 이 높은 성능을 위해서는 좋은 모델이 전제되어야 하지만, 학습 데이터 역시 매우 중요하다. 이 글에서는 카카오의 번역 서비스가 양질의 대규모 학습 데이터(병렬 말뭉치)를 확보하기 위하여 사용한 기술 중 Ableualign 툴을 소개하려 한다.

본론에 앞서 우선 BLEU가 어떤 의미인지 이해해야 한다. 이 개념을 간단히 설명하면, 원문에 대한 사람의 번역 결과와 번역기의 결과가 얼마나 유사한지를 수치화 한 것이다. 수치화 하는 방식은 문장을 구성하는 단어들을 n-gram으로 만들고 그 n-gram들이 얼마나 서로 매칭되는 지를 보는 것이다¹⁾.

n-gram은 문장을 어떤 단위로 쪼갰을 때 인접한 n개를 모은 것을 의미한다. 단위는 여러가지가 가능한데 띄어쓰기를 단위로 하여 3-gram을 구해보자.
예) input: "I am a boy"
3-gram: "I am a", "am a boy"

이런 식으로 문장을 쪼개는 이유는, 문장간의 유사도를 측정할 때, 문장의 일부로 매칭을 해보기 위함으로, 본 글에서는 유사도를 sliding window방식으로 비교하기 위한 것이라 생각해도 된다.

이 방식은 문법 구조, 유의어 등을 보지 않기 때문에 한계를 가지긴 하지만 다른 더 나은 방법론이 아직 없기 때문에 번역의 품질 평가에도 여전히 많이 사용되고 있다.

데이터 확보에 왜 BLEU를 쓰는가 라는 의문이 들 수 있을 것이다. 병렬 말뭉치(parallel corpus)의 raw data는 대부분 글(article) 단위이고 문장(sentence) 학습을 하는 신경망 기반 기계번역(neural machine translation, NMT) 모델을 위해서는 문장 단위로 정렬을 해주는 툴이 필수적이다. 카카오에서는 정렬(alignment)을 위하여 초기에 규칙 기반 툴을 만들어서 사용했는데 한계가 있음을 경험했다. 규칙을 프로그래밍 하기가 너무 힘들었고 도메인이 바뀌면 기존 규칙이 적합하지 않는 문제가 발생하였다. 이처럼 규칙은 복잡하지만 사람이 하면 쉬운 문제의 경우 해결할 수 있는 적절한 방법이 있는데, 바로 딥러닝을 적용하는 것이다. 결국 이미 딥러닝을 통해 학습된 번역기를 활용하면 쉽게 align을 할수 있을 것이란 결론에 도달하였고 Bleualign을 사용하게 된 것이다. Ableualign은 Bleualign을 개선한 버전인데 비교적 간단한 아이디어로 많은 효과를 보았다. 이제 본론으로 들어가 보자.

Bleualign

Bleualign²⁾은 BLEU score를 이용해서 두 말뭉치들을 정렬하는 툴이다. 번역기의 학습을 위해서는 병렬 말뭉치가 필요한데, Bleualign에 두 가지 언어로 된 글을 던지면 그 안의 문장 단위 번역 쌍을 뽑아준다. 카카오 번역의 모델을 학습시킬 때 쓸 데이터를 만들기 위해 Bleualign을 사용하였다. 이 툴의 동작 방법을 간단히 살펴보자.

(1) 다음과 같은 한국어 영어로 된 글이 있다고 하자. (문장 단위로 분리가 되어 있다고 가정한다)

[그림 1] 한국어, 영어 글

한국어	영어
지난 여름은 너무 더웠어. 이제 반팔티를 입어도 될 것 같아. ... 여기에 뭔가 있었는데... 에어컨이 고장난 것 같아	I'm Hulk! I will destroy everything!!! It was so hot last summer. I think I can wear short sleeves now. ... Shall we turn on the air-conditioner? I think the air conditioner is out of order.

같은 글에 대한 문장들이므로, 비슷한 문장들이 많이 존재하나 무조건 1:1 매칭이 되지는 않는다.

(2) 이 중 하나를 한영번역기(또는 영한번역기)를 통해 번역한다.

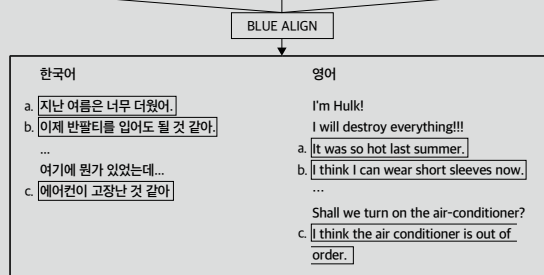
[그림 2] 한영번역기를 이용한 번역

한국어	기계 번역 결과
지난 여름은 너무 더웠어. 이제 반팔티를 입어도 될 것 같아. ... 여기에 뭔가 있었는데... 에어컨이 고장난 것 같아	It was so hot last summer. I think I can now wear short sleeves. ... There was something here... I think the air conditioner is broken.

(3) 이제 한국어, 영어, 기계 번역 결과를 동시에 입력값으로 Bluealign에 넣어주면 문장 단위의 병렬 말뭉치가 출력된다. a,b,c 박스는 각각 같은 문장의 다른 언어로 된 표현(representation)들이다.

[그림 3] Bluealign 출력 결과

한국어	기계 번역 결과	영어
지난 여름은 너무 더웠어. 이제 반팔티를 입어도 될 것 같아. ... 여기에 뭔가 있었는데... 에어컨이 고장난 것 같아	It was so hot last summer. I think I can now wear short sleeves. ... There was something here... I think the air conditioner is broken.	I'm Hulk! I will destroy everything!!! It was so hot last summer. I think I can wear short sleeves now. ... Shall we turn on the air-conditioner? I think the air conditioner is out of order.



3번 단계에서 기계번역 결과와 영어 말뭉치를 통해서 같은 문장을 찾게되는데, 이 때, 비슷한 문장을 찾기위해 BLEU score를 사용한다. 학습 데이터를 얻기위해 Bleualign tool을 잘 사용하였는데, BLEU를 개선시키면 더 많은 병렬 말뭉치를 얻을 수 있지 않을까 하는 생각이 들었고, 그 결과 나온 것이 점수 함수(score function)를 ABLEU로 대체한 Ableualign이다. 이제 BLEU에 대해서 알아보고, 개선을 시켜보도록 한다.

글 | 오형석 hulk.oh@kakaocorp.com

새로 만드는 것은 무엇이든 흥미가 있어서 다양한 것들을 제작하다가 프로그래머가 되었습니다. 현재는 카카오의 AI부문에서 연구 및 개발을 하고 있으며, 이 분야 내에서 새로운 것을 창조해보려 노력 중입니다.

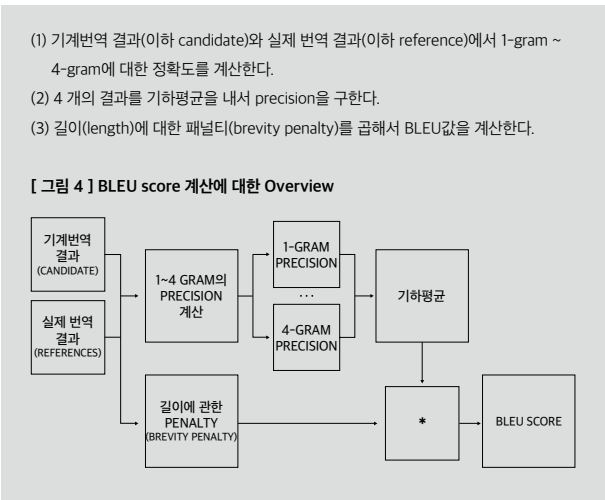
BLEU

기계 번역의 품질을 평가하는 측정 기준(evaluation metric) 중의 하나로서, 간단하고 빠른 계산으로 인간이 생각하는 품질과 높은 상관관계(correlation)를 가지고 있어, 현재까지도 가장 널리 쓰이는 측정 기준이다.

BLEU의 입력값(input)은 '기계 번역 결과'와 '여러 개의 실제 번역 결과'이며, 출력값(output)은 '0~1 사이의 값'이다. 출력값이 1에 가까울 수록 번역 결과가 좋은 것을 의미한다. 이 문서에서는 조금 더 쉽게 설명하기 위해 실제 번역 결과가 하나만 존재한다고 가정할 것이다.

1) BLEU Overview

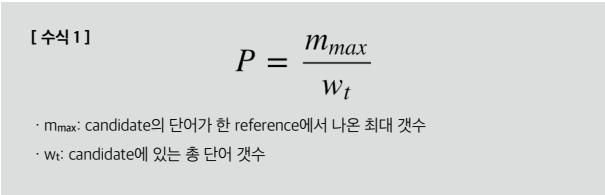
BLEU를 간략하게 나타내보면 다음 그림과 같다.



각 방법에 대해서 자세히 알아보자.

2) n-gram의 정확도 구하기

쉽게 설명하기 위해 예시를 들어서 띄어쓰기를 단위로 한 1-gram의 정확도를 계산하겠다. Candidate로 '나는 밥을 먹었다', reference로 '나는 밥을 버렸다'가 들어왔다고 가정하자. 여기서 candidate의 1-gram은 '나는', '밥을', '먹었다'이며, 2-gram은 '나는 밥을'과 '밥을 먹었다' 이다. 1-gram의 정확도(precision)는 [수식 1]로 구해진다.



즉 위의 예제에서 각각은 다음과 같은 값을 갖게되어 1-gram precision은 2/3가 된다.

'나는'의 m_{max}=1/ '밥을'의 m_{max}=1/ '먹었다'의 m_{max}=0

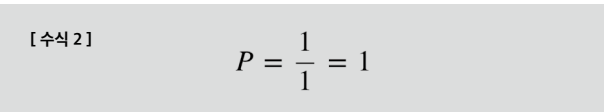
왜 굳이 reference에서 나온 최대 갯수로 clipping을 하는지에 대해서 궁금할 것이다. 이는 번역기 모델이 반복적으로 단어를 뱉어내는 경향이 있는데, 정확도를 계산할 때 반복된 단어들을 전부 넣어버리면 점수가 올라가므로 이를 방지하기 위해서다.

[표 1] 은 BLEU: a method for automatic evaluation of machine translation³에서 발췌한 예제이다. [표 1]의 경우 max 값으로 clipping 하지 않으면 7/7로 정확도가 1이 돼버린다.

[표 1] 번역은 안좋은데 unigram precision이 잘 나오는 경우

Candidate	the	the	the	the	the	the	the
Reference1	the	cat	is	on	the	mat	
Reference2	there	is	a	cat	on	the	mat

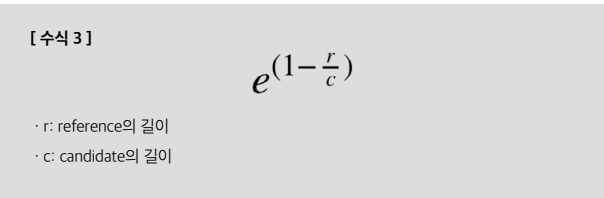
또한 위처럼 정확도를 계산하면 짧은 번역문을 선호하는 문제가 또 있다. 예를 들어, candidate로 '나는'이 나왔다고 하면,



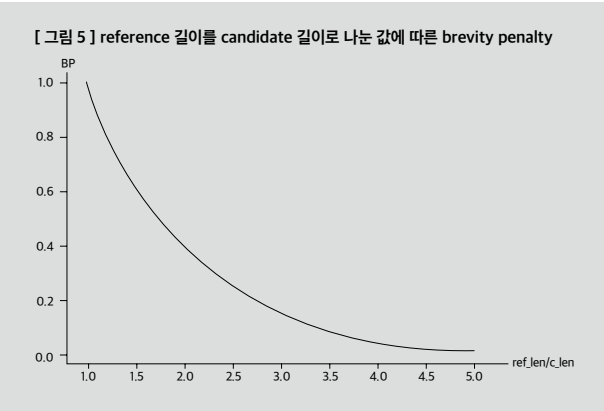
[수식2] 와 같이 된다. 따라서 candidate와 reference의 길이를 비교하여 패널티를 주어 보정한다.

3) Brevity penalty

문장의 길이를 이용해 penalty를 준다. 다음 [수식 3]을 precision에 곱해줘서 페널티를 줄 수 있다.



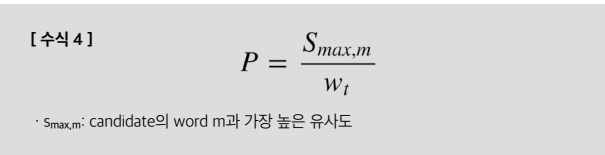
[그림 5] 는 reference 길이를 candidate 길이로 나눈 값에 따른 brevity penalty이다. Candidate가 reference 길이의 절반만 되더라도 정확도가 반 이상으로 떨어지게 됨을 알 수 있다.



ABLEU

1) Idea

우리는 BLEU에서 n-gram precision을 계산하는데 정확히 매칭되는 것만 카운팅을 하고있는 것을 주목했다. '이쁘다' 와 '예쁘다' 는 분명히 비슷한 뜻인데 0,1로만 구분하여 집계(discrete counting)해서 매칭이 되지 않는 결과가 발생했다. 만약 비슷한 단어에 대해서 0과 1 사이의 어떤 유사도(similarity)를 연속적(continuous)으로 부여할 수 있다면, 이것이 훨씬 좋은 결과를 낼 수 있을 것이라 생각했다. 유사도를 구할 수 있다고 가정하고 ABLEU는 BLEU수식에서 단 한개만 바꾸면 된다.

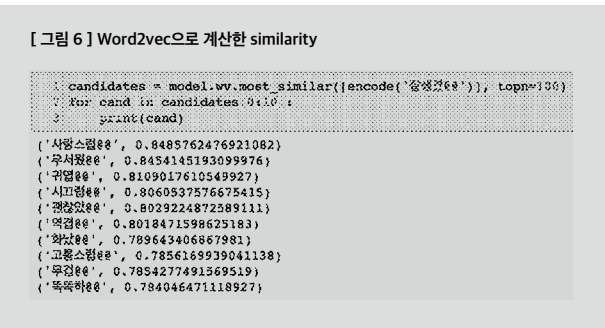


이제 우리가 할 일은 유사도를 계산하는 방법을 만드는 것이다.

2) Similarity

(1)Word2vec

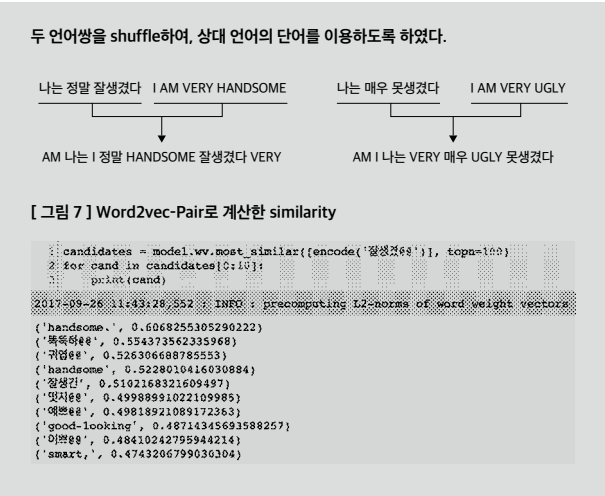
Word2vec은 단어를 고정된 길이의 벡터로 embedding하는데 쓰인다. 따라서 word2vec으로 단어들을 embedding하고 그것의 유사도를 쓰면 될 것이라 생각했다. 단 한가지 문제점은 word2vec은 단어의 위치에 따른 유사도가 크기 때문에 정반대의 단어도 유사도가 높게 나온다는 단점이 존재했다. 예를 들어 word2vec의 경우 '잘생겼'과 가장 비슷한 단어들 10개를 뽑으면 [그림 6] 과 같은 결과가 나온다. '역겹'과 '잘생겼'이 비슷하게 나온 것을 보면 문장 내의 위치(position)가 유사도에 미치는 영향이 큰 것을 알 수 있다.



(2)Word2vec-Pair

이를 방지하기 위해 만든 방법은 굉장히 단순하며 비슷한 단어의 유사도를 잘 뽑아냈다. 그 방법은 우리가 가지고있는 병렬 말뭉치를 썬어서 word2vec을 학습(training)시키는 것이다. 두 문장을 썬어버리면 영어 단어와 한글 단어가 word2vec의 입력값으로 같이 들어가며, 비슷한 단어의 유사도를 높이는데 서로를 이용할 것이라는 생각이었다. [그림 7]을 보면, word2vec-pair의 경우 뜻이 비슷한

단어가 더 잘 뽑히는 것을 알 수 있다. 두 언어에 대해서 model을 학습시키기 때문에, 영어 단어도 뽑히게 된다.



위의 word2vec-pair를 통해 나온 embedding으로 서로의 유사도를 구했고, 이를 사용하여 ABLEU를 만들 수 있었다.

결과

[표 2] Bleualign과 Ableualign의 수율 및 정확도

	Bleualign	Ableualign
수율(%)	35.64	47.1
정확도(%)	86	93.73

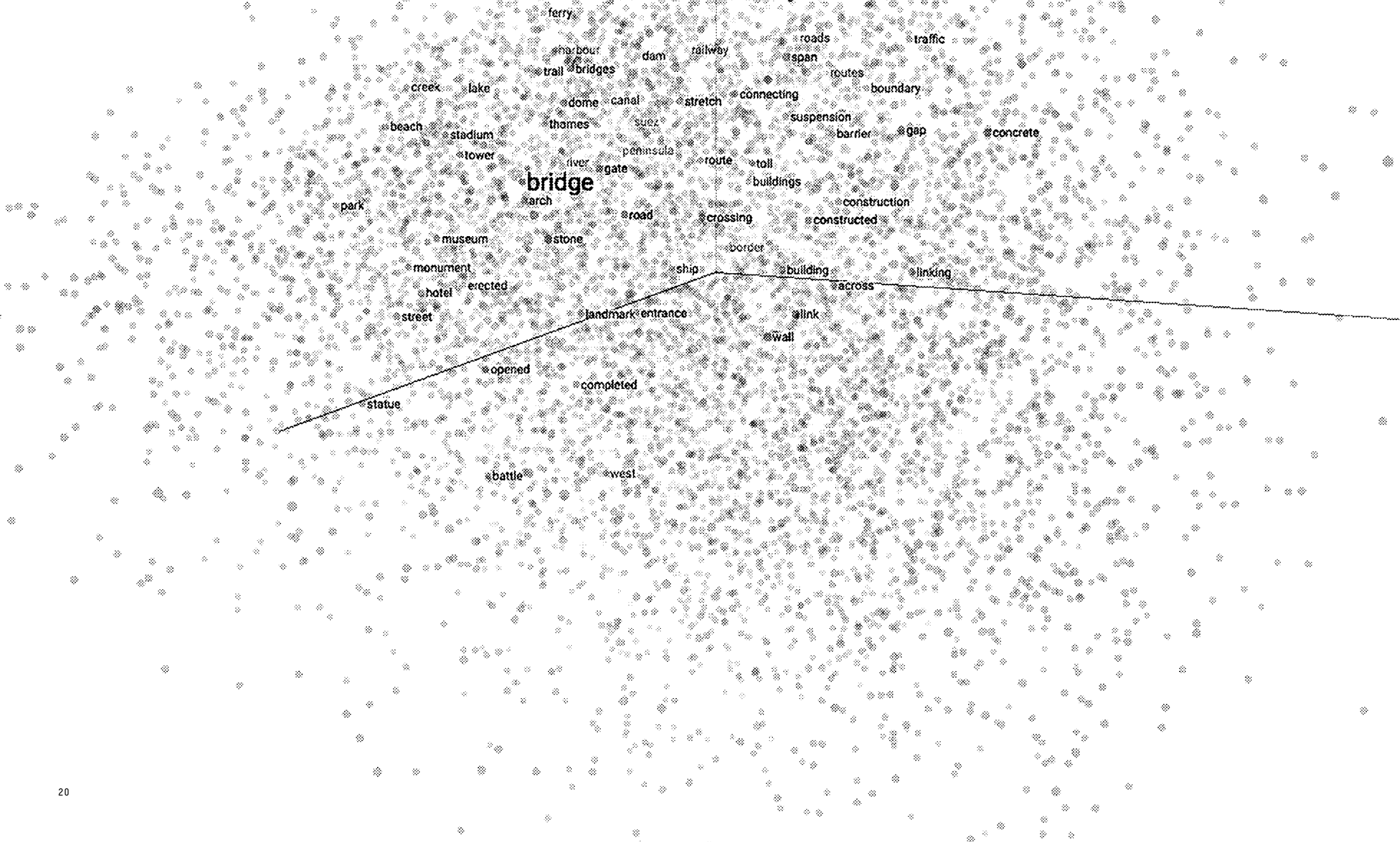
실험 결과 동일한 번역기를 사용할 때 Bleualign 대비 Ableuaign은 수율이 35.64%에서 47.10%로 약 11% 증가하였고 정확도도 86%에서 93.73%로 증가하였다(정확도는 sampling 하여 수동 평가로 측정하였다). 이 말은 ABLEU 적용으로 더 나은 품질의 데이터를 기존보다 30% 정도 더 추출했다는 말이다. Bluealign의 성능은 번역기의 성능에 의존적이기 때문에 더 나은 번역기를 사용할 경우 수율이 더 높아질 수 있다.

따라서 Ableualign을 사용해 번역기의 품질을 올리고, 이를 다시 Ableualign 번역 스텝에 사용하여 수율을 올리는 반복 작업이 가능했다. 실험이 진행되는 동안 카카오에서는 한번의 번역기 모델 개선이 더 이루어졌는데, 이 모델을 적용하여 최종적으로는 기존보다 58% 더 많은 병렬 말뭉치를 추출할 수 있었다.

결론

카카오는 번역기의 학습에 이용할 양질의 병렬 말뭉치를 얻기 위해 Bleualign을 사용하였다. Bleualign의 점수 측정 기준(score metric)을 더 좋게 바꾸면 더 질 좋은 데이터를 더 많이 얻을 수 있으리라 기대했으며 0, 1 로만 구분하여 집계하는 BLEU score의 discrete counting에 주목했다. 이를 연속적인(continuous) 유사도로 바꾸기 위해 word2vec을 언어쌍으로 학습하는 방법을 고안하였으며, 그 결과 Ableualign을 만들 수 있었다. 또한 Ableualign으로 더 많이 확보된 데이터를 통해 번역기의 품질을 높이고, 이 번역기를 다시 Ableualign의 번역 step에 넣어서 반복(iteration)을 통해 선순환 구조를 만드는 것이 가능했다.

[그림 8] Word embedding의 예시 - bridge와 가까운 단어들*4

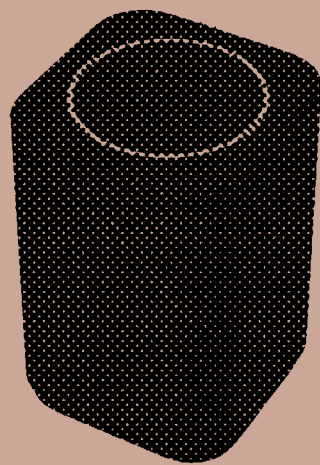


[표 3] 단어 'bridge'와 다른 단어 간의 거리	
단어	거리
bridges	0.408
road	0.554
tunnel	0.571
gate	0.590
canal	0.594
ferry	0.596
dam	0.613
tower	0.616
thames	0.633
railroad	0.634
harbour	0.634
link	0.648
arch	0.650
suspension	0.652
railway	0.656
roads	0.657
crossing	0.659
straits	0.660
bay	0.660
connecting	0.660
highway	0.662
river	0.663
gap	0.675
strait	0.676
tunnels	0.682
construction	0.686
avenue	0.687
boundary	0.688
erected	0.690
monument	0.693
peninsula	0.693
street	0.696
opened	0.699
connect	0.699
building	0.700
route	0.704
landmark	0.707
rail	0.709
connects	0.710
traffic	0.710
port	0.711
interstate	0.712
deck	0.714
span	0.715
routes	0.718

*1 참고 | 보통 3~4-gram 까지 진행된다. *2 참고 | <https://github.com/rsennrich/Bleualign> *3 논문 | Papineni, K., Roukos, S., Ward, T., Zhu, W.J., (2002). BLEU : a method for automatic evaluation of machine translation, O2 Proceedings of the 40th Annual Meeting on Association for Computational Linguistics,(pp. 311-318). *4 참고 | <http://projector.tensorflow.org>

카카오미니와 제프리 힌튼 그리고 ICCV 2017

hot topic	조희주 김수형 카카오미니는 어떻게 스무고개 정답을 맞출까	24
	이수경 강종호 제프리 힌튼의 캡슐망을 풀이하다	30
	이주영 노명철 ICCV 2017 참관기	34



카카오 AI 리포트는 AI 분야에서 크게 주목받는 세 가지 소재를 택했습니다. 카카오미니, 제프리 힌튼 교수의 캡슐망 논문, 그리고 국제컴퓨터비전학회(ICCV) 2017 참관기입니다. 카카오미니와 관련해서는 카카오미니 기능인 스무고개 게임의 제작기를 담았습니다. 카카오미니와 한 번이라도 스무고개 게임을 해보신 분은 열독하실 것으로 예상됩니다. 딥러닝의 대부인 제프리 힌튼이 공개한 캡슐망 논문 역시 딥러닝의 또 다른 시발점으로 평가받고 있습니다. 캡슐망이 과연 무엇이기에 이런 학계의 평가가 나오는 지에 대한 간명한 설명에 지난 10월 이탈리아 베니스에서 열린 ICCV 참관기가 이어집니다. 이 세 글만 봐도, 최신 AI 동향을 가늠하실 수 있을 겁니다.

카카오미니는 어떻게 스무고개 정답을 맞출까

올해 봄, 가수 김정치마가 낸 3집 앨범 TEAM BABY의 타이틀 곡 '나랑 아니면'은 "야 나랑 놀자 밤늦게까지 함께 손뼉 치면서" 라는 가사로 시작한다. 호모 루덴스(Homo Ludens). 이처럼 우리는 누구나 즐겁게 놀고 싶은 욕구를 가지고 있다. 어린 시절부터 지금까지 놀이기구를 타거나 블록 쌓기, 퍼즐 맞추기, PC게임, 모바일게임 등의 '놀이'를 끊임없이 해 오고 있는 것이다. 때문에 그러한 '놀이'가 새로운 시장을 만들어 나가고 있는 인공지능 스피커에 들어가는 것은 어찌보면 당연한 일이다.

AI스피커에 왜 게임이 필요할까?

우리가 꾸준히 놀이를 하는 이유를 생각해보면, 그를 통해서 '재미'를 얻기 때문이다. 카카오미니가 담을 수 있는 수많은 기능 중에서 게임은 놀이를 통한 재미 제공에 목적을 두고 있고, 교육용 게임에선 학습도 덤으로 얻을 수 있다.

우리는 유아기 때부터 이런저런 스킬을 얻고 있는데, 그 중 카톡 보내기보다 먼저 배우는 것이 말하기다'. '엄마', '아빠', '맘마'로 시작해서 발전해 나가지 않나. 외국어를 배울 때를 생각해보자. 가장 먼저 트이는 것은 듣기, 그 다음은 말하기, 그 다음이 바로 쓰기의 순서다.

'카카오'라는 브랜드는 스마트폰과 함께 카카오톡을 사용해온 대중들에게는 가깝고 친근한 이미지로 자리잡았을 것이다. 카톡 사용 집단의 범위가 넓다는 것을 고려해볼 때, 카톡보다 나중에 나왔지만 음성으로만 이루어지는 카카오미니를 완벽한 문장을 구사하는데 어려움이 있는 아이들부터 휴대전화를 잘 다룰 줄 모르는 분들도 사용할 수 있었으면 좋겠다. 예를 들면 카톡보단 아직 전화가 편한 우리 어머니와 같은 분들.

"나랑 놀자!"를 하기까지

정보만 전달하는 스피커도 좋지만 대화를 주고받고 같이 놀 수 있어야 한다고 생각한다. 우리들은 다른 사람들과의 관계에서 정보전달도 많이 하지만 함께 공감을 나누거나 놀면서 보다 가까워지기 때문이다.

예를 들어, 학교에 새롭게 입학한 에이미라는 친구가 있다. 기존에 알던 친구라곤 한명도 없는데, 아뿔싸 댄 생각을 하다가 선생님이 숙제 내주시는 걸 못 듣고 말았다.

정보전달 - 용기를 내어 옆자리의 라이언에게 "오늘 숙제가 뭐야?" 라고 물어보고, 라이언은 에이미에게 숙제가 뭐였는지 알려준다.
소통 - 그 다음 날, 에이미는 라이언과 인사를 하고 어제 숙제가 어려웠다는 내용의 대화를 한다.
재미 - 이후, 에이미와 라이언은 가까운 친구가 되어 같이 술래잡기도하고 얼음땡도 하는 사이가 되었다.

이렇게 가까워지는 단계를 보면 순서가 정보전달 → 소통 → 재미 의 순서가 있는데, AI스피커가 친구가 되려면 역시나 이런 순서를 따르는 것이 좋지 않을까? 그렇다면 카카오미니에 게임을 넣어보자.

WHY 스무고개?

그 많은 게임 중에 왜 스무고개를 넣었는가에 대해 묻는다면, 그 답은 첫 스타트를 끊기 '쉬워 보여서'였다. 답변의 종류가

'예/아니오(YES/NO)' 두 가지로만 이루어져 있어 음성 입력이 쉽고², 그에 따라 게임을 진행하다 보면 답이 쉽게 나올 것이라고 생각했기 때문이다.

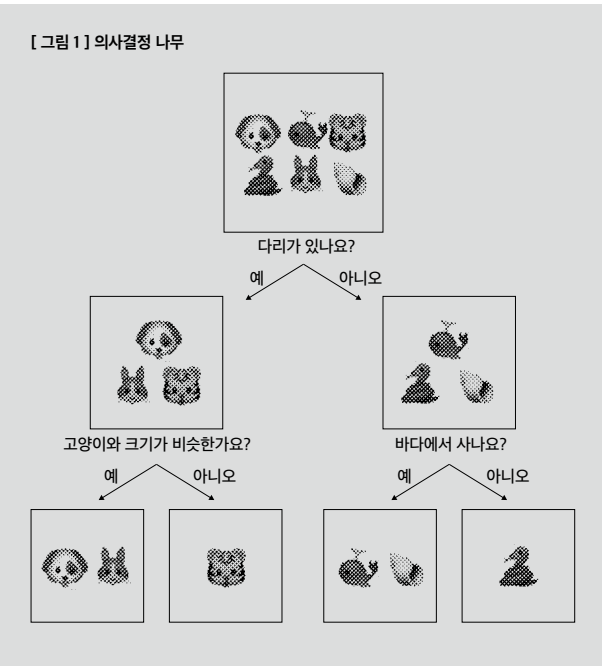
어떤 방법으로 문제를 맞출 수 있도록 해야 하나

스무고개 게임에서 알고리즘의 핵심은 '어떻게 하면 답을 잘 맞추게 할 수 있을것인가'이다. 더 정확히 말하면 '어떻게 하면 답을 더 빨리, 더 정확하게 맞추게 할 수 있을까'이다. 많이 쓰이는 방법을 생각해보면 가장 단순하게는 의사결정 나무(decision tree)부터 랜덤 포레스트(random forest), 나이브 베이즈(naive bayes), 뉴럴 네트워크(neural network) 등을 생각해 볼 수 있다.

의사결정 나무

처음엔 '예/아니오(YES/NO)'로만 답이 이루어지기 때문에 당연히 의사결정 나무(decision tree)면 될거라는 순진한 생각을 했다.

스무고개의 원리 자체는 이진검색(binary search)과 유사한 방식이다.



의사결정 나무에서도 가장 단순하게 생각해볼 수 있는 것은 고정된 트리가 딱 하나만 존재해서 질문의 순서가 정해져 있는 의사결정 나무이다. 예를 들어, 동물 스무고개를 제공하는 웹사이트인 애니멀게임³에서의 첫 번째 질문은 항상 "Can it fly?(날 수 있나요?)" 이다. 여기서 '예(YES)'를 누르면 "Does it have feathers?(깃털이 있나요?)" , '아니오(NO)'를 누르면 "Is it a very

글 | 조희주 amy.wine@kakaocorp.com

통계학과 졸업 후 머신러닝의 맛을 보다가 개발에 흥미가 생겨 능력자들 사이에서 학교다닐 때 안하던 공부중인 개발꿈나무 입니다.(라고 쓰고 우주먼지라고 읽습니다) 머신러닝을 적용한 프로덕트를 제 맘대로 만들어 내는게 단기적 꿈이고 세계평화와 인류의 행복을 위한 인공지능을 만드는 것이 장기적 꿈입니다.

글 | 김수형 cantabile.58@kakaocorp.com

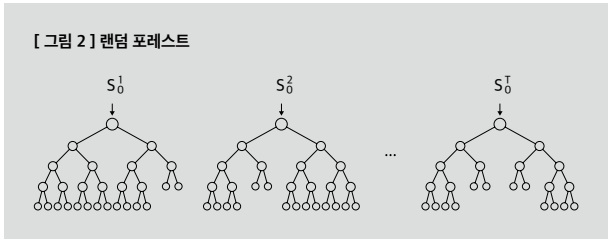
이창호를 능가하는 인공지능 바둑을 만들고 싶었으나 딥마인드에 선수를 빼앗겨 억울해하고 있는 늑은 개발자입니다. 한 때, 대학에서 학생들에게 수학과 프로그래밍을 가르쳐보기도 했지만 여전히 수학은 너무 낯설고 프로그래밍은 너무 어렵지만 합니다.

small animal?(덩치가 매우 작은가요?)"가 나온다. 세상 노잼… 이런 게임이라면 금방 싫증이 날 것이다.

질문 순서에 랜덤 요소를 반영하고 동적으로 트리를 새로 생성해 나간다면 이런 문제는 해결할 수 있을 것이다. 하지만 이상적인 경우엔 잘 동작해야만 하는 의사결정 나무 모델은 유저가 답변을 한번이라도 다르게 말하거나 인식이 잘못되면 돌아올 수 없는 강을 건너게 된다. 이후로는 어떤 질문을 아무리 많이 하더라도 답을 맞출 수 없다.

예를 들어, 위의 [그림1]에서 '호랑이'를 생각한 유저가 실수로 혹은 잘 모르고 "고양이와 크기가 비슷한가요?"라는 질문에 '아니오'가 아닌 '예'로 대답을 해버렸다면? 의사결정 나무에서 한번 뻘은 가지는 다시 뒤로 돌아오지 않기때문에 이제 유저의 답은 영영 나오지 않게 된다. 스무고개, 만만치 않다.

Random Forest⁴



그런 의사결정 나무의 단점을 극복하기 위해 의사결정 나무를 여러 개 만들어 숲(forest)을 이뤄서 경우의 수를 대폭 확장한다는 개념의 랜덤 포레스트(Random Forest)를 써볼 수 있다⁵.

하지만 랜덤 포레스트에서 데이터셋의 크기가 작은 경우엔 오버피팅(over-fitting)될 확률이 높다. 즉, 너무 구체적으로 디테일하게 학습돼 버리는 것이다. 또한 새로운 데이터가 반영될 때마다 숲(forest)을 다시 만들어 줘야(rebuild)해 풀고자 하는 문제에 비해 계산량이 너무 많아진다.

Neural Network

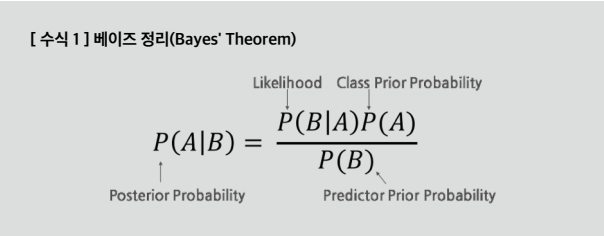
스무고개를 뉴럴 네트워크로 진행할 수도 있다. 로빈 부르게너(Robin Burgener)가 1988년 부터 뉴럴 네트워크를 이용해서 만든 스무고개는 웹⁶이나 애플리케이션에서 해볼 수 있고, 이 원리는 특허⁷로 등록되어 있다. 이 원리는 2006년 미국항공우주국(NASA) 주최의 콜로키움에서 발표⁸된 적도 있다. 아마도 현존하는 가장 크고 오래된 인공지능 스무고개일 것이다.

하지만 직접 들어가서 게임을 해보면 보이듯 선택지가 기존에 알던 '예/아니오(YES/NO)'가 아니고 문제에 따라 선택

대안이 바뀐다. 오랜 시간 많은 데이터를 쌓았겠지만 아쉽게도 우리가 원하는 품질과 방향이 아니었다. 작고 제한된 데이터셋(data set)이지만, 더 친근하고 재미있는 스무고개를 만들고 싶었기 때문이다.

베이즈 모델(Bayes' Model)

랜덤 포레스트의 오버피팅 문제를 걱정하지 않아도 되고 실시간으로 변하는 데이터를 적용시킬 수 있는 솔루션으로 나이브(naive; 꾸밈 없는, 자연스러운) 베이즈가 있다. 나이브 베이즈 분류(naive bayes classification)⁹라는 이름으로 머신러닝에서 많이 쓰이고 있는 방법으로, 상황에 따라 랜덤 포레스트보다 편리할 수 있다.



컨셉은 베이즈 정리를 꾸밈없이 담백하게 사용해서 확률을 계산하는 것이다. 베이즈 정리가 생소하다면 이것만 기억하면 된다¹⁰. "사후 확률(posterior probability)은 사전 확률(prior probability)과 우도(likelihood)의 곱이다." 즉, '일이 발생하기 전의 확률'과 '일이 발생할 가능성의 정도'를 곱하면 '일이 발생한 후의 확률'인 것이다.

카카오미니가 스무고개의 답을 맞추는 과정

카카오미니의 스무고개는 베이즈 정리를 기반으로 한 알고리즘을 사용하고 있다. 앞서 언급했듯이 베이즈 정리는 우도와 사전확률의 곱이고, 더 간단히 말하면 전체의 확률에서 그 사건이 일어날 확률이다. 아래의 예시를 보면 좀 더 구체적으로 이해가 될 것이다.

사후분포를 구하기 위한 준비물에는 P(A), P(B|A), P(B)가 필요하다 차례대로 준비해보자

먼저 P(A)는 각 동물에 대한 사전 확률이다. 게임의 플레이어가 얼마의 확률로 그 동물을 생각할지를 나타낸다. 초기 사전 확률은 임의로 지정했으며, 이는 데이터가 충분히 쌓이면 그 분포에 기반해 변할 것이다. 이 사전 확률의 총 합은 1보다 작다. 1이 아니라 1보다 작은 이유는 우리가 게임 속에 넣어 둔 동물이 사용자가 정답으로 생각할 수 있는 세상의 모든 동물을 완벽하게 넣은 것이 아니기 때문이다. 식으로 나타내면 [수식 2]와 같다.

【 수식 2 】 모든 사전 확률의 합을 1보다 작게한다.

$$\sum_i P(A_i) < 1$$

이 조건에서, 각 동물의 사전 확률을 지정해준다. 여기서 사전 확률의 총 합은 0.8(=80%)로 두자. 강아지나 토끼 등 비교적 친근한 동물은 선택될 확률이 높고, 소라나 뱀 등 친근하지 않은 동물은 선택될 확률이 낮다.

【 표 1 】 사전확률(Prior Probability)						
Prior	강아지	토끼	고래	호랑이	소라	뱀
P(A)(= 강아지, 토끼,...,뱀)	0.2	0.2	0.1	0.1	0.1	0.1

이제 우도인 P(B|A)를 준비할 차례다. A인 경우에 B일 확률로, A는 'A동물을 정답으로 생각하고 있는 경우', B는 'B질문에 대해 '예'라고 할 경우'로 본다. 나이브 베이즈 모델을 적용하기 위한 필수 조건으로 각각의 확률은 독립적(independent)이라고 가정한다.

【 표 2 】 우도(Likelihood)						
Likelihood	강아지	토끼	고래	호랑이	소라	뱀
다리가 있나요?	0.9	0.9	0.1	0.9	0.1	0.1
고양이와 크기가 비슷한가요?	0.8	0.9	0.1	0.2	0.1	0.2
바다에서 사나요?	0.1	0.1	0.9	0.1	0.9	0.2
알을 낳나요?	0.1	0.1	0.1	0.1	0.8	0.8
두 글자 인가요?	0.1	0.9	0.9	0.1	0.9	0.1

[표2]는 동물 스무고개에서 임의의 확률을 정한 표이다. 실 서비스에서 이 또한 [표1]과 같이 사용자 데이터로부터 갱신되어 확률이 변화하게 된다.

이제 준비가 끝났으니 사후분포를 구할 수 있다

'토끼'를 생각하고 있다고 가정하고 게임을 시뮬레이션 해보자. 질문의 순서 알고리즘은 배제하고 위의 다섯 문제를 순서대로 물어볼 것이다. 토끼의 사전 확률은 0.2이다. "다리가 있나요?"의 답은 '예'이므로 P(B|A) = 0.9를 곱한다. 식으로 나타내면 [수식 3]과 같다.

【수식 3】	
P(다리가 있다고 대답 토끼) = (0.2*0.9)/((0.2*0.9)+(0.2*0.9)+(0.1*0.1)+(0.1*0.9)+(0.1*0.1)+(0.1*0.1)) * 0.8	

문제에 대한 답이 "아니오"인 경우에는 1-P(B|A)로 계산을 한다. 이렇게 다섯번째 문제까지의 토끼에 대한 대답으로 모든 동물의 사후 확률을 정리해 소수점 셋째 자리까지 표현해보면 다음과 같다.

【 표 3 】 사후확률(Posterior Probability)							
Posterior	Answer	강아지	토끼	고래	호랑이	소라	뱀
다리가 있나요?	예	0.300	0.300	0.017	0.150	0.017	0.017
고양이와 크기가 비슷한가요?	예	0.351	0.395	0.002	0.044	0.002	0.005
바다에서 사나요?	아니오	0.353	0.398	0.000	0.044	0.000	0.004
알을 낳나요?	아니오	0.355	0.399	0.000	0.044	0.000	0.001
두 글자 인가요?	예	0.071	0.719	0.000	0.009	0.000	0.000

다섯 개의 질문을 한 끝에 사용자가 토끼를 생각하고 대답했을 확률이 약 72%로 다른 확률에 비해 월등히 높은 것을 볼 수 있다. 만약 "70% 이상이면 이 동물이 정답인지 물어보겠다"라고 설정해두었다면 다섯번째 질문 이후에 "토끼 인가요?"라는 질문이 나오고, '예'라는 대답이 돌아오면 게임이 끝나고 '아니오'가 돌아오면 토끼의 사후확률 값은 매우 적은 값, e로 치환되고 다음 질문들로 게임을 계속 진행한다.

"두글자인가요?"와 같이 답이 명백한 질문에도 잘못된 답을 할 가능성이 있어서 어떤 확률에도 극단적인 값인 0과 1은 발생하지 않는다. [표2]에 0.000으로 나타난 부분은 수가 매우 작아져서 소수점 셋째 자리로 잘랐을 때 뒤로 밀려나서이다.

다음 질문으로는 어떻게 좋을까

앞에서는 질문의 순서 알고리즘은 생각하지 않고 베이즈 정리만 써보았다. 데이터셋이 작을 때는 질문 순서를 랜덤으로 돌려도 20문제 안에 정답이 나올 확률이 크다. 하지만 데이터셋이 커지게 되면, 보다 효율적인 알고리즘을 필요로 한다.

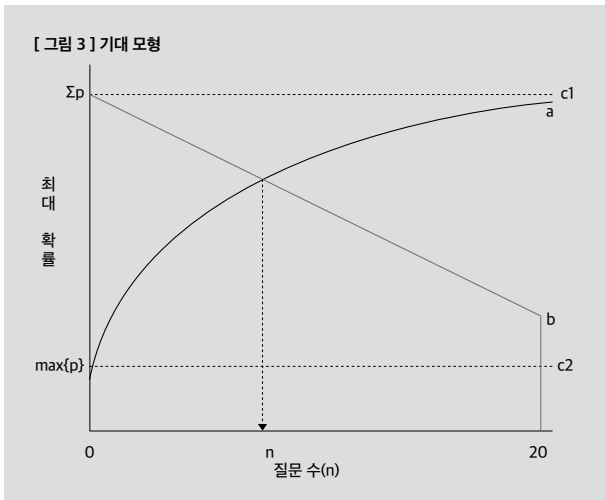
그렇다면 어떻게 다음 문제를 고를까. 스무고개 문제를 낼 때는 20개라는 한정된 질문만으로 답을 맞춰야한다. 보유하고 있는 질문셋에 대해 사용자가 '예/아니오(YES/NO)'로 대답한 이후의 각 정답셋에 대한 확률을 모두 계산한다. 그 중 우리가 선택하게 되는 질문은 각 질문에 대한 대답이 예(YES) 혹은 아니오(NO)일 경우의 사후 확률 중 낮은 값들을 비교해서 그 중 가장 큰 값을 고르는 것이다. 이를 식으로 나타내면 [수식 4]와 같다.

【 수식 4 】 Q는 질문셋, A는 정답셋을, y, n은 yes/no를 나타낸다.	
$\max_{k \in Q} \{ \min_{y, n} \{ \max_{i \in A} (P_i) \} \}$	

답을 제시하는 최적의 타이밍은

너무 빨리 답을 제시하면 정답률이 떨어지고, 늦게 제시하면 사용자가 지루함을 느끼게 된다. 따라서 질문의 수가 어느정도 일정하게 유지될 필요가 있다. 질문의 갯수가 적으면 보다 엄격한 기준을 제시하고, 많으면 느슨한 기준을 제시한다. 그리고 질문의 갯수는 특정값(예: 20)을 넘기지 않도록 한다.

기대하는 게임 모형과 앞으로의 과제



위의 스무고개 알고리즘 모델을 대략적으로 그려보면 이런 그림이 나온다. x, y축은 각각 게임에서 나온 질문의 수, 최대 사후 확률의 크기를 나타내며, a 곡선은 최대 확률곡선, b 선은 답을 제시할 최소 정답률을 나타낸다. a 곡선과 b선이 만나는 지점의 n번째(기댓값) 질문 이후 답을 제시하게 되는 것이다. 질문의 순서 알고리즘에 따라 a 곡선은 최대정답률에 빠르게 혹은 느리게 가까워진다. 즉, a 곡선이 b선과 만나는 n값이 작아지거나 커져서 평균 질문 수에 영향을 미치게 되는 것이다.

c1선의 위 부분은 사용자가 데이터에 없는 대상을 생각한 경우, c2선은 가장 빈번하게 선택되는 대상이 정답일 확률값을 나타내며, 앞의 예시에서는 각각 0.8, 0.2이다.

스무고개의 퀄리티를 높이려면

보유 정답 데이터셋을 늘려서 최대 정답률을 높이거나 질문셋의 질을 높이고 순서를 적절히 배치한 후 정답에 이르는 질문의 수를 줄이도록 한다.

마치며

아직도 완벽하진 않지만 초기 버전의 스무고개는 답을 너무 못 맞춘다는 원성이 자자했다. 준비된 정답이나 질문의 개수도 한정적이고 확률 테이블도 구체적인 유저 데이터가 없어 추정치를 사용하고 있기 때문이라면 너무 구차한 변명인가? 수학적인 모형과는 별개로 인공지능 스피커 안의 게임이라고 해도 결국 사람이 만드는 거고 사람이 사용하는 것이기 때문에 '휴머니즘'을 완전히 배제하기가 어렵다. 인공지능 스피커를 사용하다보니 인간미가 느껴지는 부분이 킬링 포인트(killing point)인 것 같은데 이는 정답률에 비례해서 높아지는 것은 아닐 것이다.

너무 똑똑해도 밉다. 카카오미니와 함께하면 언제라도 부담없이 즐길 수 있는 적당히 똑똑하고 적당히 허술한, 내 친구 같은 게임 및 다른 형태의 인공지능 콘텐츠를 기대해본다.

“데이터는 모든 알고리즘을 압도한다.”

“数据秒杀算法”

- 바이두 CEO 로빈 리(李彦宏, 리옌홍) -

^{*1} 참고 | California State University. “연령별 언어발달의 단계”. <http://www.calstatela.edu/academic/ccoe/programs/cats-korean/연령별-언어발달의-단계> ^{*2} 참고 | 실제로는 “두 글자입니까?” 라는 간단한 질문에 도, 사람들은 “예”와 “아니오” 이외에도 “물라”, “글쎄-”, “맞아!”, “비밀이야”, “안 가르쳐주지”, “잘 아네”, “3글자야” 등 다양하게 답하기는 한다. ^{*3} 참고 | <http://www.animalgame.com> ^{*4} 참고 | <https://www.techleer.com/articles/107-random-forest-supervised-classification-machine-learning-algorithm/> ^{*5} 참고 | <https://www.linkedin.com/pulse/significant-project-time-being-spent-data-preparation-wanjohi-ph-d/> ^{*6} 참고 | <http://www.20q.net> ^{*7} 참고 | <https://www.google.com/patents/US20060230008?dq=Artificial+neural+network+guessing+method+and+game> ^{*8} 참고 | <https://ecolloq.gsfc.nasa.gov/archive/2006-Spring/announce.burgener.html>, ^{*9} 참고 | https://ko.wikipedia.org/wiki/나이브_베이즈_분류 ^{*10} 참고 | https://ko.wikipedia.org/wiki/베이즈_추론

제프리 힌튼의 캡슐망을 풀이하다

딥러닝(deep learning)의 대부인 제프리 힌튼(Geoffrey Hinton) 교수가 최근 '캡슐망(capsule networks)'이라는 새로운 신경망과 훈련 알고리즘인 '캡슐 간 동적 라우팅(dynamic routing between capsules)'¹⁾을 논문을 통해 공개했다.

지난 1979년 힌튼 교수가 아이디어를 처음 고안한 지 무려 수십 년 만에 이를 실제로 구현해, 캡슐망이 CNN(convolutional neural networks)을 대체할 수도 있다는 점에서 이번 논문은 주목할 만하다. 물론 힌튼 교수가 오늘날 전 세계에서 널리 쓰이는 수많은 딥러닝 모델과 알고리즘을 개발해온 사람이라는 명성 또한 이번 논문에 대한 딥러닝 학계의 관심을 증폭시켰다²⁾.

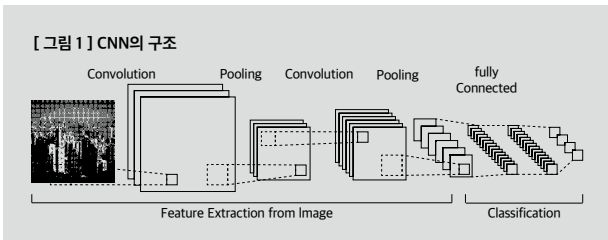
캡슐망이 고안된 배경과 그 구조를 이해하려면 우선 CNN을 살펴볼 필요가 있다. CNN이 가진 구조적 한계를 개선한 것이 바로 캡슐망이기 때문이다.

CNN이란

CNN은 데이터로부터 자동으로 특징(features)을 학습하는 대표적인 모델이다. 인간의 시각(vision) 정보 처리 방식을 흉내낸 것으로, 특히 이미지 인식과 분류에서 탁월한 성능을 낸다. 알파고의 승리도 CNN의 발전이 있었기에 가능했다. CNN의 메커니즘은 생각보다 간단하다. 입력과 가까운 층에서는 가장자리(edge), 곡선(curve)과 같은 저수준(low level) 특징을 학습한다. 점차 높은 층으로 올라 갈수록 질감(texture), 물체 일부분(object parts)과 같이 고수준(high level) 특징을 인식한다. 출력층에서는 물체의 종류를 인식하는 등 복잡한 추론을 수행한다.

CNN은 크게 세 가지 종류의 층으로 구성된다. 컨볼루션 층(convolution layer)은 이미지로부터 특징을 추출한다. 풀링 층(pooling layer)은 이미지에서 표본을 추출하는 방식으로 학습 속도를 높인다. FC 층(fully connected layer)은 최종적인 분류 작업을 담당한다.

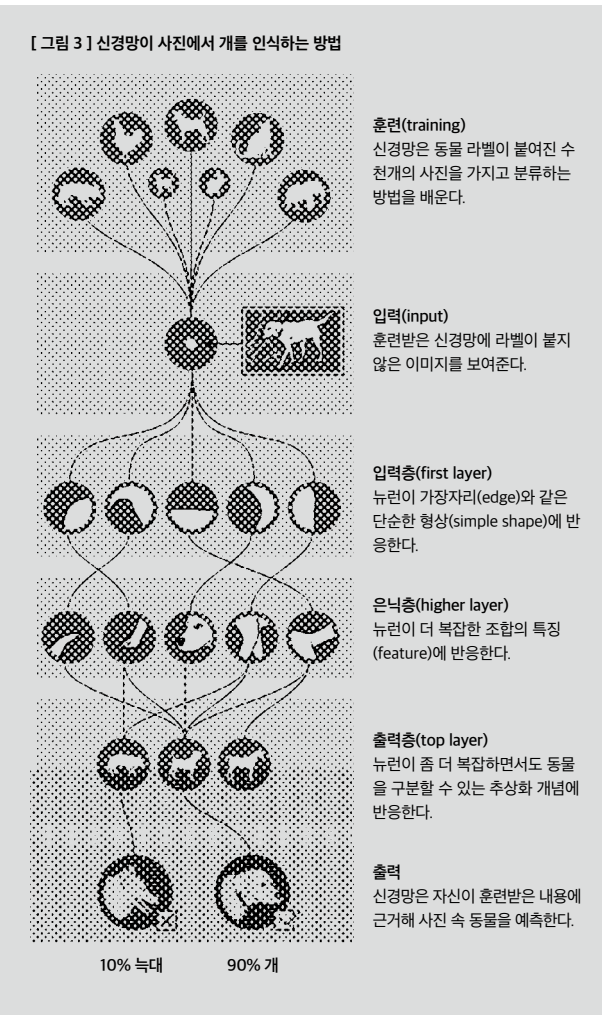
1) 컨볼루션 층



일반적인 신경망은 완전히 연결돼 있다. 모든 데이터가 하나의 신경층에서 다른 신경층으로 '전파(propagate)'된다는 의미다. 28×28처럼 작은 크기의 이미지³⁾의 경우 784(=28×28)개의 입력값만 처리하면 된다. 전체 이미지에서 특징을 학습하는 데 큰 무리가 없는 수준이다. 문제는 1024×768처럼 실제로 우리 실생활에서 사용되는 큰 이미지를 학습시키는 일이다. 예를 들어, 10⁴개의 입력값과 100개의 특징을 학습한다면 10⁶개의 가중치(weight)를 학습해야 한다. 학습 과정에서의 계산 또한 28×28 이미지와 비교해 약 100배 이상의 시간이 걸린다.



다행스러운 건 이미지가 정적(stationary)이라는 점이다. 이미지의 한 부분의 통계치가 다른 부분과 비슷하다는 의미로, 한 영역에서 학습한 특징은 다른 영역에서도 유사한 패턴을 찾는데 활용할 수 있다. 완전히 연결된 신경망 대신 CNN을 사용할 수 있는 이유다.



컨볼루션 층에서는 작은 이미지 영역인 패치(patch)를 큰 이미지 위에 빙빙 돌리면서(convolve) 각각 다른 특징 활성화(activation value)을 얻는다. 이 패치는 특징을 감지한다는 점에서 특징 추출기(feature detector) 또는 커널(kernel), 필터(filter)라고 부르기도 한다.

2) 풀링 층

캡슐망과 관련지어 볼 구성요소는 풀링 층⁵⁾이다. 풀링을 활용하면 학습 시간을 줄이면서도, 이미지 구성 요소의 위치 변화에 더 잘 대응할 수 있다. 일반적으로 사용되는 맥스 풀링(max pooling)은 주변 영역의 추론 결과값 중 최댓값만을 상위층으로 보낸다. 특징 탐색 영역(feature map)은 1/4로 축소되고, 위층에서의 특징 추출 및 추론에 대한 부담은 크게 줄어든다. 더불어 분류 작업에 유리한

글 | 이수경 samantha.lee@kakaobrain.com

2016년 3월 알파고와 이세돌 9단이 펼치는 세기의 대결을 두 눈으로 목도한 이후 인공지능을 제대로 공부해 봐야겠다고 결심했습니다. 인공지능 본진이자 연구소인 카카오브레인으로 걸어 들어온 이유입니다. 인공지능 기술과 이로 인해 바뀔 미래 사회를 다루는 글을 통해 사람들과 소통하고 싶습니다.

글 | 강종호 wiles.inno@kakaobrain.com

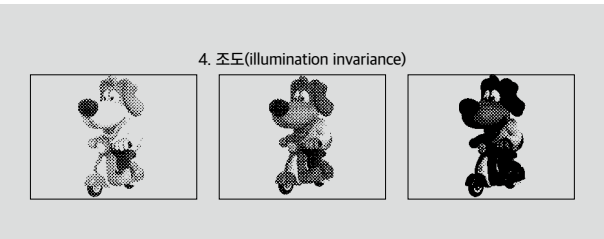
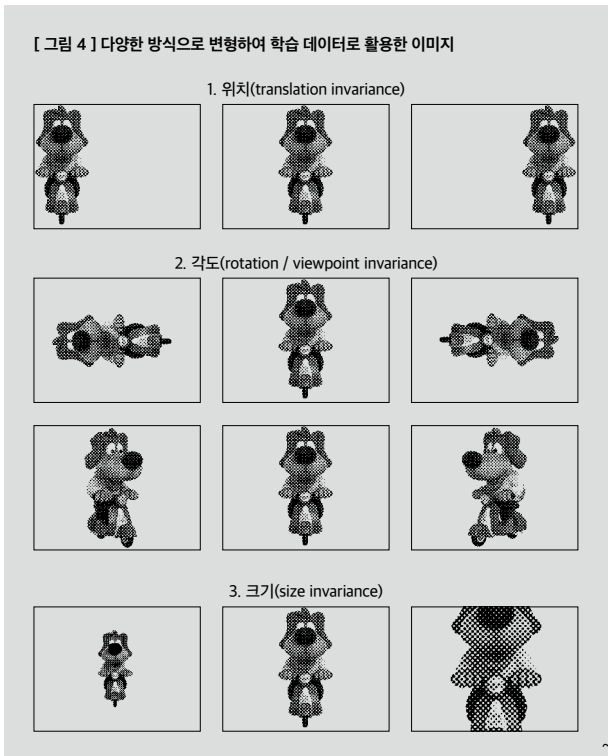
어릴 때부터 수학을 좋아했고, 학부에서 수학을 전공했습니다. 대학원에서 금융을 전공하고, 취미로 프로그래밍과 수학, 통계를 공부하다 알파고를 만났습니다. 거대한 변화의 시작임을 느끼고 그 흐름을 타고 싶어 카카오브레인에 들어왔습니다. 인공지능으로 더 행복한 세상을 만들고 싶습니다.

불변성질(invariance)을 얻을 수 있는 장점도 있다. 예를 들어, 얼굴의 방향이나 각도에 따라 특징이 서로 다르게 추출되면 상위층에서 이를 제대로 인식할 수 없는데, 맥스 풀링은 위치에 상관없이 눈, 코, 입을 인식할 수 있도록 해준다.

CNN이 가진 구조적 한계

탐색 속도를 높이고자 고안한 맥스 풀링은 아이러니하게도 CNN의 취약점으로 작용한다. 맥스 풀링을 거치면 이미지 구성 요소의 공간 관계에 관한 정보를 잃는다. 예로 들면, 사람 얼굴을 인식하는 CNN은 피카소의 작품 속(기형적인) 대상(object)을 사람으로 인식한다. 맥스 풀링을 통해 눈, 코, 입, 귀의 상대적인 위치, 방향과 상관없이 특징을 추출하기 때문이다.

종합하자면, CNN은 풀링을 통해 위치(translation)와 관계없이 객체를 동일하게 인식하지만, 방향(orientation)이나 비율(proportion)이 달라지면 서로 다른 객체로 인식한다. 아울러 물체를 바라보는 시점(viewpoint) 변화에 유독 취약하다. 이미지의 각도나 크기가 변형될 경우 해당 이미지를 제대로 인식하지 못한다는 의미다. 이런 이유로 힌트 교수도 오래 전부터 맥스 풀링에 관해 회의적인 모습을 보이며 이를 재앙(disaster)이라 표현하기도 했다. 이를 개선하고자 다양한 방식으로 변형한 이미지를 학습 데이터로 활용하는 방법(data augmentation)을 사용하지만, 대신 학습 시간이 증가하는 단점이 있다.



교란 샘플(adversarial example)은 CNN의 약점을 보여주는 또 다른 예다. 이 샘플은 사람의 판단에는 영향을 주진 않으나 머신러닝에서 오분류(misclassification)를 낳는다. '1픽셀로 딥 뉴럴 네트워크 속이기(One pixel attack for fooling deep neural networks)' 논문에서는 단 하나의 픽셀만 달라져도 대상을 제대로 인지하지 못한다는 결과를 보여주기도 했다.

신경망에 3D 세계를 구현하다

힌트 교수는 렌더링(rendering)에서 CNN의 한계를 극복할 아이디어를 얻었다. 렌더링은 매시(mesh)*6 객체를 포즈(pose, 위치와 방향) 정보를 활용해 시각적 이미지를 구성하는 것을 의미한다. 힌트 교수는 인간의 뇌가 바로 이 렌더링의 역과정(inverse graphics)을 통해 영상을 인식한다고 봤다. 즉, 눈으로 획득한 시각 정보를 계층적인 표현으로 해체(deconstruct)한 뒤, 사전에 습득한 지식과 매칭해 물체의 종류와 포즈 정보를 역추론한다는 설명이다. 이를 위해서는 신경망이 불변성질(invariance) 대신, 등가성질(equivariance)을 가져야 한다고 봤다. 이미지 속 물체가 적절히 변환(위치, 방향 등) 되면 추론 결과도 이에 상응해서 변해야 한다는 것이다.

그래서 힌트 교수는 CNN 층을 깊숙이 쌓는 대신, 캡슐을 계층적으로 쌓아 올린 캡슐망을 고안했다. 캡슐은 여러 신경망 층으로 구성된 단위 요소다. 그리고 중첩된 계층간 동적 라우팅(dynamic routing)하는 방식을 고안했다.

*scalar-out layer→vector-out layer
*max-pooling→dynamic routing

기존 신경망에서는 각각의 뉴런이 독립적으로 동작한다. 반면, 캡슐망에서는 뉴런들의 그룹인 캡슐이 단위 요소다. 뉴런의 출력값이 스칼라(scalar)이지만, 여러 뉴런으로 이루어진 캡슐의 출력값은 벡터가 된다. 그리고 이 벡터의 크기는 어떤 개체가 존재할 확률을, 벡터의 방향은 그 개체의 성질을 표현한다. 여기서 새롭게 고안한 비선형 함수인 '스퀴싱 함수(squashing function)'가 벡터 전체에 적용된다. 스퀴싱 함수를 통해 벡터의 크기는 1을 넘지 않게 된다. 개체가 존재할 확률을 효과적으로 나타내는 장치인 셈이다.

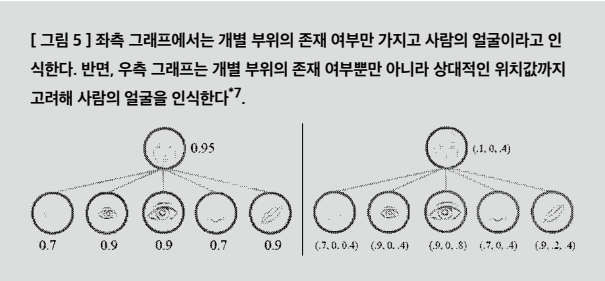
[수식 1] 스퀴싱 함수

$$V_j = \frac{\|s_j\|^2}{1 + \|s_j\|^2} \frac{s_j}{\|s_j\|}$$

일반적인 CNN에서는 층을 깊게 쌓아야만 물체의 특징을 인식할 수 있다. 반면 캡슐망은 층을 깊게 쌓지 않아도 된다. 캡슐이 출력하는 벡터 자체에 많은 정보가 담겨 있어서다. 아울러, 아래층 캡슐의 출력 벡터를 어떤 가중치로 사용할지는 맥스 풀링이 아닌 동적 라우팅 알고리즘이 결정한다.

캡슐망에서 캡슐의 모든 부모 캡슐은 캡슐의 출력을 예측한다. 예측의 가중합(weighted sum)은 스퀴싱 함수를 거쳐 캡슐의 출력이 된다. 그리고 이 예측 벡터와 자식 캡슐의 출력 간 '내적(scalar product)'이 큰 (즉 연관성이 큰) 부모 캡슐과 결합이 강화된다(가중치가 커진다). 학습이 잘 났을 때 정렬된 이미지가 들어온다면, 최종 출력 벡터의 방향은 이미지의 회전이나 크기 같은 정보를 포함한다(여기서 최종 출력 벡터는 출력층의 캡슐들이 출력하는 벡터 중에서 크기가 가장 큰 벡터를 의미한다). 그리고 이와 강하게 결합된 부모 캡슐의 예측 벡터들도 비슷한 방향을 가진다. 만약 눈, 코, 입의 크기와 각도가 제멋대로인 얼굴이 입력으로 들어온다면, 이러한 요소를 감지한 벡터들의 방향은 서로 어긋나게 된다. 방향이 어긋나면 자연스럽게 그 합의 크기는 감소한다. 이는 얼굴이라고 판단하는 확률이 낮아지는 셈이다. 단순히 객체를 구성하는 각 요소의 존재만으로 객체를 인지하는데 그치지 않고, 각 요소 간 상관관계까지 고려해 객체를 인지한다는 의미다.

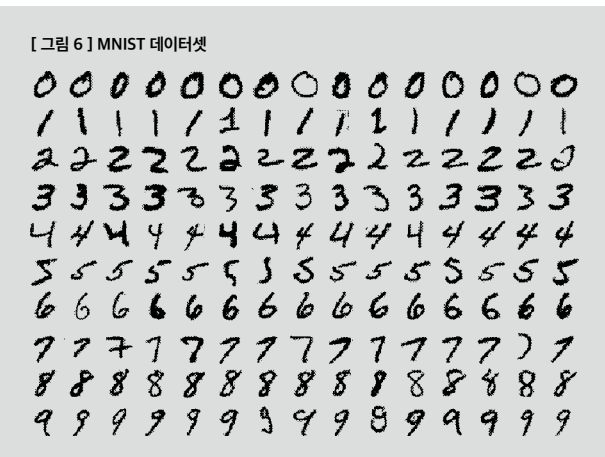
쉽게 이야기하면, [그림 5]의 왼쪽처럼 CNN이 눈과 코, 입이 있으면 얼굴이라고 인식하지만, [그림 5]의 오른쪽처럼 캡슐망은 두 눈이 인접해 있고 눈 사이 아래쪽에 코가 있으며, 눈 아래 입이 있으면 얼굴이라고 인식한다.



새롭게 고안된 캡슐망을 기반으로 MNIST*8 데이터셋을 훈련시켜 본 결과, 최신 CNN 대비 에러율을 45%까지 줄였다. 아울러 화이트박스(white box)*9 교란 공격(adversarial attack)에 대해서도 보다 효과적으로 저항했다.

물론 캡슐망을 현업에 적용하는 데는 다소 시간이 걸릴 것으로 보인다. 이번 논문이 최신의 딥러닝 신경망을 소개하고 있으나 그

성능 검증은 아직 끝나지 않았다. 우선, MNIST 데이터셋(28×28, 6만개 흑백 이미지)에서는 잘 동작하더라도 이미지넷(256×256 이상, 100만개 컬러 이미지)과 같은 매우 방대한 크기의 데이터셋에서도 비슷한 성능을 낼지는 미지수다. 아울러 학습에 걸린 시간 또한 기존 CNN보다 더 긴 시간이 걸렸다. 정확도에 큰 차이가 없다면 컴퓨팅 자원을 더 소모하는 캡슐망을 쓸 이유가 없다.



캡슐망은 이제서야 그 존재를 알렸다. 힌트 교수가 오랜 시간 가슴 속에 품어온 아이디어를 실제로 구현해내는 단계에 이르렀다. 물론 아직 더 많은 검증은 필요하나 희망적인 부분도 있다. 캡슐망은 기존 CNN보다 사람이 사물을 인식하는 방식에 더 가깝다는 이유에서다. 이를 통해 합리적인 결과를 내지 못하는 아이디어에 그치는 것이 아니라 딥러닝의 새 시대를 여는 시발점이 되기를 기대해본다.

*1 참고 | 이 논문의 제1저자는 사라 사보(Sara Sabour), 제2저자는 니콜라스 프로스트(Nicholas Frost)다. 논문 저자의 이름 순서와 내용, 인용을 분석해보자면 해당 연구는 구글브레인팀에서 진행하고 힌트 교수는 감수를 맡은 것으로 추측된다. 그로부터 8일 후 공개된 'EM 라우팅을 활용한 행렬 캡슐(Matrix Capsules with EM Routing)' 논문에는 익명의 저자들(anonymous authors)로 기재돼 있으나, 학계에서는 힌트 교수도 저자로 참여했을 것으로 추정하고 있다. *2 참고 | <http://v.sports.media.daum.net/v/20160308060204892> *3 참고 | 모든 이미지는 픽셀값의 행렬로 표시할 수 있다. 행렬 속 픽셀값의 범위는 0~255사이다. 그리고 R(Red), G(Green), B(Blue) 3종의 채널로 구성된다. *4 참고 | http://ml4a.github.io/ml4a/neural_networks/ *5 참고 | 이미지의 특정 영역을 요약한다는 의미에서 표본추출(sub-sampling) 또는 리사이징(resizing)이라 표현하기도 한다. 풀링에는 평균값을 계산하는 '에버리지 풀링 (average pooling)' 등 다양한 방법이 존재한다. 최근에는 실제로 더 나은 결과를 보여주는 '맥스 풀링(max pooling)'이 주로 쓰인다. *6 참고 | 3차원 그래픽에서 다면체를 구성하는 다각형(polygon)과 정점(vertex)들의 집합 *7 참고 | <https://jhui.github.io/2017/11/03/Dynamic-Routing-Between-Capsules/> *8 참고 | 필기체 숫자를 분류하는 머신러닝 모델을 평가하기 위한 표준화된 데이터. 18*18 픽셀 크기의 6만개 훈련 데이터와 1만개 테스트 데이터로 구성돼 있다. *9 참고 | 응용 프로그램의 내부 구조와 동작을 검사하는 소프트웨어 테스트 방식

ICCV 2017 참관기

세계에서 가장 오래된 국제 영화제가 열리는 곳이자, 가장

아름다운 관광 도시 중 하나로 꼽히는 이탈리아 베니스에서 ICCV

2017(국제컴퓨터비전학회, international conference on computer vision, ICCV)이 개최되었다.

ICCV는 CVPR(컴퓨터비전 및 패턴인식 학회, conference on computer vision and pattern recognition)과 함께 컴퓨터 비전

분야에서 가장 권위있는 학회로 꼽힌다. 1987년 영국 런던에서 첫

학회가 열렸고, 1~3년을 주기로 진행되다가 1999년부터 2년에 한 번씩

개최되고 있다. 전기전자공학 분야의 최대 기술 조직인 IEEE(전기 전자

기술자 협회, institute of electrical and electronics engineers)가

주최하는 학회로서, 발표 논문으로 채택되는 것은 매우 어렵다.

컴퓨터 비전의 가장 권위있는 학회, 2017 ICCV를 다녀오다

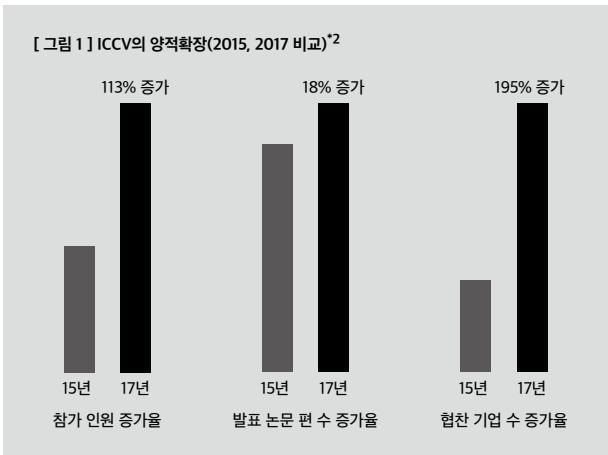
최근 개최된 CVPR¹과 유사하게 ICCV 역시 딥러닝의 인기에 힘입어

양적 확장이 눈에 띄었다. 수치적으로 비교해 보면, 2009년에서

2015년까지 1100~1400여명이 참석하던 것에 비해 이번에는 무려

3100명이 넘는 많은 사람이 참석을 했다. 그 외 투고되는 논문

편수나 협찬 기업의 수 등 많은 면에서 증가가 있었다.



국내 연구자들의 ICCV에 대한 기여는 큰 것으로 확인되고 있다.

한국은 무려 230여명이 참석해 4번째로 참석 인원이 많은

국가였다. 또한 구체적인 수치가 공개되지는 않았으나, 논문 제출

상위 연구기관에 KAIST(한국과학기술원)와 서울대학교가 이름을

올리는 등 양과 질 두 측면 모두에서 ICCV에 많은 기여를 하고

있음이 확인 가능했다. 이는 국내에서 컴퓨터 비전을 연구하고 있는

대학원 또는 기업의 연구소 수는 상대적으로 적지만, 연구진의

수준은 결코 낮은 편이 아님을 방증한다. ICCV에 다녀온 뒤 느낀 몇

가지 생각과 Best Paper로 선정된 논문들을 정리해 보았다.

트렌드를 따라가는데 기본이 되는 튜토리얼(tutorial)

학회 튜토리얼(tutorial)이 상당히 알찬 경우가 많다. ICCV의 경우

GAN(generative adversarial network) 이라던가 Instance-level

Visual Recognition 등의 주제로 다양한 튜토리얼이 열렸다. 각

튜토리얼은 가장 기본이 되는 논문부터 본 학회에서 발표될 가장

최근의 논문 소개까지 진행됐다. 이러한 튜토리얼은 개별 주제에

익숙하지 않은 연구자라고 하더라도, 최근의 논문을 이해하는데

큰 도움을 줄 것으로 생각됐다. 특히, 각각의 알고리즘을 제안한

저자의 직강을 듣고 질문할 수 있는 프로그램의 경우 잘 알고 있는

분야라고 하더라도 한번 더 확인하는 계기가 됐다. 다른 사람들에게

가장 추천하고 싶은 시간이다.

핵심을 짚어주는 스포트라이트

CVPR 2016에서 처음 접했던 스포트라이트(spotlight)가 ICCV에도

적용이 되었다. 스포트라이트는 포스터 세션(poster session)에

발표될 논문 중 일부에 대해 짧게 핵심만 짚어주는 시간이다. 발표된

논문에 대해 질문을 할 때 알고리즘 설명을 되묻지 않고 바로

질문을 하거나 관심있는 논문을 선별하는데 많은 도움을 주었다.

오랄 세션(oral session)이 1회부터 9회까지의 직관하는 야구

경기라면, 스포트라이트는 스포츠 뉴스처럼 승부처만 요약해서

보는 야구 경기 같은 느낌이었다.

단 한가지 아쉬운 점

아쉬운 점을 딱 한가지 뽑자면, 학회에서 발표되는 논문이 이미

arXiv 를 통해 수 개월, 길게는 수 년 전에 공개되다 보니, 종종 본인

발표 마지막에 추가 연구를 진행한 새 논문의 arXiv 주소를 공개하는

경우가 있었다는 점이다. 최신 논문이 더 이상 최신 논문이 아니다

라는 생각이 들었고, 이미 알고 있는 내용이다 보니 흥미가 떨어지는

순간도 있었다. 이러한 문제는 ICCV 뿐만 아니라 CVPR이나 타 학회

발표 역시 갖고 있다. arXiv를 통한 선공개는 분명 장점이 더 많지만,

이러한 단점에 대한 보완책이 미비하다는 점 또한 아쉬웠다.

최우수 논문 소개

1) Best Student Paper : Focal Loss for Dense Object Detection

딥러닝 기반의 검출기는 크게 Fast/Faster R-CNN^{3,4}처럼

후보 영역을 검출하고 후보 영역에서 정확한 bounding box와

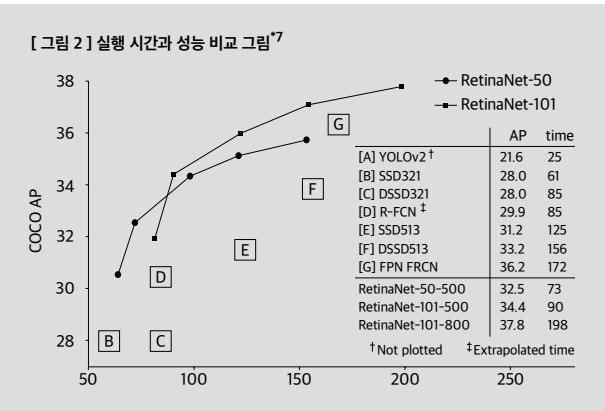
카테고리를 구별하는 2-stage 검출기와, YoLo⁵, SSD⁶처럼

한 번에 bounding box와 카테고리를 구별을 하는 1-stage

검출기로 구별된다. [그림 2]를 보면 2-stage 검출기(A, B, C, D,

E, F)는 상대적으로 느리지만, 더 정확한 성능을 보이고 1-stage

검출기(G)는 반대로 매우 빠르지만 다소 부족한 성능을 보인다.



글 | 이주영 michael.lee@kakaocorp.com

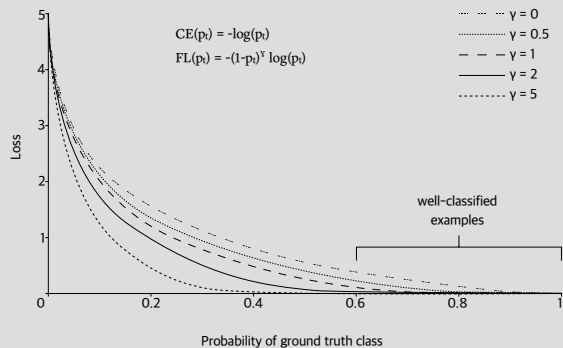
대학원을 졸업하고 9년여의 시간을 컴퓨터비전 관련해서 회사에서 연구/개발 업무를 해왔지만, AI 리포트에 기고할 때마다, 실수한건 없을까 싶어 항상 긴장하게 되는 것 같습니다. 다음에 기고할 기회가 주어진다면, 참관기가 아닌 제 논문에 대한 이야기를 다룰 수 있게 되기를 기대합니다.

글 | 노명철 joshua.roh@kakaocorp.com

오랜만에 참석하는 권위있고 큰 학회였습니다. 많은 논문들이 발표돼 부지런히 돌아 다녔는데, 시차 때문인지 체력적으로는 너무 힘든 학회였습니다. 대부분의 논문 내용들은 인터넷을 통해 검색할 수 있는 시대이지만 전체적인 추세나 분위기의 경우 학회장에 참석하지 않고는 알기 힘들음을 다시 한번 느끼며, 건문을 넓힐 수 있었던 좋은 기회 였다 생각합니다.

이러한 결과는 쉽게 구별되는 객체가 아닌 영역(background)이 객체 영역보다 매우 많이 학습에 참여하게 되어 발생하는 것으로 간주된다. 이를 해결하기 위해 새로운 손실 함수(loss function)를 제안했다. 새로 제안하는 손실 함수는 잘 분류되는 예제는 손실(loss)이 적게 발생해 [그림 3]과 같은 특징을 가지도록 디자인 되었고, 검출 결과에서 좋은 성능을 보였다. 문제로 가정했던 것이 맞았음이 실험적으로 증명된 것이다.

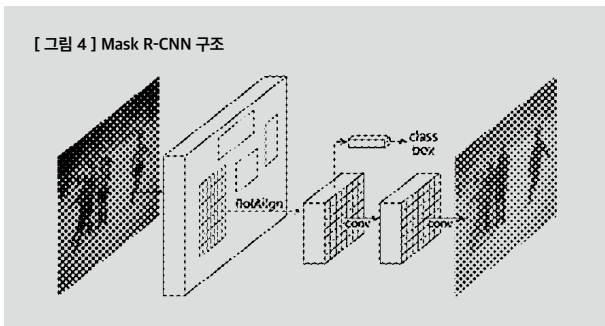
【 그림 3 】 확률 값과 loss 사이의 관계 그래프^{*8}



흥미로웠던 점은 객체 검출에서 성능을 높이기 위해 현재 알고리즘이 구별하기 어려워하는 대상을 학습에 더 많이 참여시키는 방법(online hard example mining, OHEM⁹) 보다도 더 높은 정확도를 보였다는 점이다.

2) 최우수 논문 : Mask R-CNN

최우수 논문은 객체 검출(object detection)과 instance segmentation을 다룬 ResNet을 처음 제안한 카이밍 허(Kaiming He), 사람 검출 알고리즘의 전문가 피오트 달러(Piotr Dollar), Fast(er) R-CNN 을 제안한 로스 걸식(Ross Girshick) 등이 참여한 Mask R-CNN 이었다. 쟁쟁한 저자들이 공동작업한 논문이기에 저자들의 명성만으로도 충분히 좋은 논문이 나올 것 같았다. 이 논문은 특히 쉽고 단순한 방법으로 instance segmentation 알고리즘을 제안했다. 기존에 객체 검출에서 높은 성능을 보인 Faster R-CNN^{*10} 의 네트워크 구조 [그림 4]에 instance segmentation task를 담당하는 분기(branch) 를 추가하였고, 이렇게 새롭게 제안한 네트워크는 COCO 2016 challenge 에서 1등을 하여, 우수함을 증명하였다.



"Simple Is the Best"라는 표현이 더 잘 어울리는 논문은 없을 것 이란 생각이 들만큼, 단순하고 정확한 결과를 보인 알고리즘이라 개인적으로 매우 관심이 가는 논문이다. [그림 5] 참고

【 그림 5 】 Mask R-CNN 결과



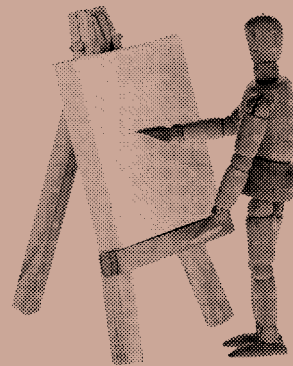
정리하며

학회에 참석하고 나면, 하고 있는 연구와 업무에 더 집중할 수 있고 스스로를 되짚어 보게 되어 항상 좋은 느낌을 받았던 것 같다. 더군다나 2년 뒤에 열리는 ICCV는 대한민국 서울에서 개최될 예정이라고 하니 마음 한구석으로 학회장 어느 한 곳에서 내가 발표를 하게 된다면 어떨까? 라는 상상까지 하게 되었다.

마지막으로, 인공지능 관련 인재 영입을 위해 학회를 찾아온 패션 기업이나 VR, AR 체험 부스 등 이전의 머신러닝 학회에서는 볼 수 없었던 기업들의 참여 모습도 인상적이었다. 또 딥러닝의 시대로 넘어오면서 데이터의 양이 이전 보다 훨씬 중요하게 되었는데, 이러한 데이터의 분류, 마킹(marking)을 대신 해주는 회사들도 부스를 차리고 나와 있었다. 딥러닝이 얼마나 학회와 산업계에 큰 영향을 미치고 있는지를 알 수 있는 부분이었다. 이 딥러닝 기술이 언제까지 지속될지, 또 다른 새로운 기술은 언제 나타날지 벌써부터 궁금해진다.

^{*1} 참고 | <https://brunch.co.kr/@kakao-it/143> ^{*2} 참고 | <http://iccv2017.thecvf.com/files/Opening/ICCV17.pdf> ^{*3} 논문 | Ren, S., He, K. & Sun, J. (2015). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, doi : arXiv:1506.01497. ^{*4} 논문 | Redmon, J., Divvala, S., Girshick, R. & Farhadi, A. (2016) You Only Look Once: Unified, Real-Time Object Detection, doi : arXiv:1506.02640 ^{*5} 논문 | Redmon, J., Divvala, S., Girshick, R. & Farhadi, A. (2016) You Only Look Once: Unified, Real-Time Object Detection, doi : arXiv:1506.02640 ^{*6} 논문 | Liu, W. et. al. (2016). SSD: Single Shot MultiBox Detector, doi : arXiv:1512.02325 ^{*7} 논문 | Lin, T., Goyal, P., Girshick, R., He, K. & Dollar, P. (2017). Focal Loss for Dense Object Detection, doi : arXiv:1708.02002 ^{*8} 논문 | Lin, T., Goyal, P., Girshick, R., He, K. & Dollar, P. (2017). Focal Loss for Dense Object Detection, doi : arXiv:1708.02002 ^{*9} 논문 | Shrivastava, A., Gupta, A., & Girshick, R. (2016). Training Region-based Object Detectors with Online Hard Example Mining, doi : arXiv:1604.03540 ^{*10} 논문 | Redmon, J., Divavala, S., Girshick, R. & Farhadi, A. (2016) You Only Look Once: Unified, Real-Time Object Detection, doi : arXiv:1506.02640

예술에 인공지능을



묻다

AI & media art

송호준 | 예술이 AI를 바라보는 시선

40

최승준 | "X의 목적은 통찰이지 Y가 아니다"

44

언젠가 한 번은 다뤄야 한다고 생각했던 분야였습니다. 예술과 AI란 주제입니다. 합리(合理)/이성(理性)의 영역으로 간주받는 과학과, 창의(創意)/감성(感性)의 영역으로 여겨지는 예술의 간극을 크게 보는 것이 일반적입니다. 최신 기술과 예술의 그것은 더 크다고 볼 수도 있습니다. 이번 호에서는 AI와 예술의 관계를 예술적 시각에서 살펴보고자 했습니다. 두 미디어 아트 작가를 섭외해, 예술이 바라보는 AI, 그리고 유관된 사례 정리를 부탁드립니다. 조금 낯선 글입니다만, 그래도 새로운 분야로 관심의 폭을 확장시키는 차원에서 통독해 보실 것을 권해드립니다.

예술이 AI를 바라보는 시선

요즘은 융합의 시대라고 해도 과언이 아니다. 이러한 시대에 예술은 인공지능을 어떻게 바라보고 있을까? 예술가들은 공학자나 과학자들과 달리, 효율성을 추구하지 않으며 주로 혼자 작업한다. 예술가들은 공학자와 과학자들이 바라보지 못한 혹은 바라보려고 하지 않은 다른 무엇인가를 인공지능에서 엿보고 있을까? 아울러 예술가들이 보여주는 인공지능의 세계는 어떤 것들이 있을까?

예술, 어떻게 정의내릴 것인가?

현재 대중 매체들이 예술을 다루는 방식은 다소 편협하다. 예술가는 아주 일상적이거나 아주 독특한 무언가에 갑자기 영감을 받기도 하고 우울해서 약물에 의존하기도 한다. 동시에 천재이자 비극적 삶의 코드를 적어도 하나씩 가지고 있는 존재로 그려진다. 여전히 가장 많이 알려진 예술가의 표현 방식은 그림을 그리는 것이다. 하지만 지금의 예술이 다루는 내용이나 표현 방식은 대중 매체가 일컫는 '예술'과는 거리가 있어 보인다. 오늘날의 예술가들은 가만히 앉아 누구를 바라보기도 하고, 특정 지역의 역사를 조사해서 보여주기도 하며 환경 문제를 이야기 하기 위해 직접 발로 뛰면서 사진을 찍기도 한다. 지금의 예술은 과거의 예술보다 더 다양한 분야를 이야기하며, 그 표현 방식과 사용하는 재료에 있어서도 이전보다 더 자유롭다고 생각한다. 미술에서도 기술을 주로 사용하는 작가들을 미디어 아티스트 혹은 뉴미디어 아티스트라 분류하고 있긴 하지만 유전자 공학을 사용한 인공지능 기술을 사용하건 예술이라 부르는 시대에 살고 있다. 따라서 무엇이 예술이고 무엇이 과학인지에 대한 구분은 아주 불명확해졌으며, 그 둘을 구분하기 위해서는 세심한 접근이 필요하다. '과학과 예술을 굳이 구분해야 할 이유가 있을까?' 라는 생각도 해본다.

이미 같은 이야기를 하고 있는 것 아닌가?

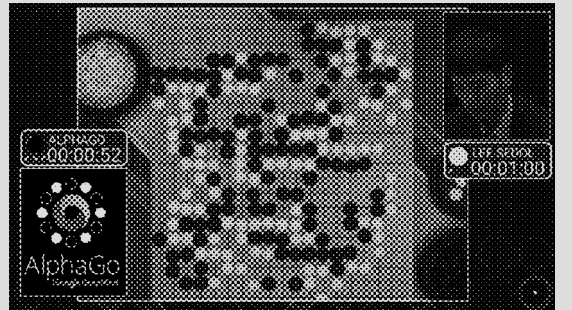
인공지능에 대한 이야기는 보통 기사에서 접하는 여타 다른 기술과 달리 우리에게 많은 생각을 하게 한다. 컴퓨터 메모리 구조가 나노(nano), 펨토(femto) 공정으로 만들어졌다는 기사를 보고 흥분하는 사람들을 찾기란 쉽지 않다. 하지만 인공지능에 대한 기사는 많은 사람들을 흥분시킨다. 그만큼 인공지능에 대한 이야기가 우리의 삶, 그리고 실존에 대한 문제와 맞닿아 있기 때문일 것이다. 여기에서 생각해보면 예술이 인공지능을 다루는 방식은 예술이 우리의 삶을 다루는 기존의 방식과 크게 다르지 않으리라 예상할 수 있다. 그리고 인공지능을 연구하는 공학, 과학자들 역시 예술 혹은 철학과 상당히 비슷한 모습으로 우리의 삶에 관한 이야기를 다루고 있다. 의도했건 안했건 알파고(AlphaGo)의 예만 보더라도 딥마인드(Deepmind)는 기술의 구현을 통해 삶에 대한 질문을 역설적으로 던지는 역할을 충분히 해냈다. 굳이 융합이라는 단어를 찾을 필요도 없이 이미 예술과 인공지능 기술은 같은 이야기를 하고 있는 것처럼 보인다.

알파고

이 글은 예술의 관점에서 바라보는 인공지능에 대해 이야기 한다. 하지만 정작 우리가 관심 있는 것은 '왜 이제서야, 그리고 갑자기

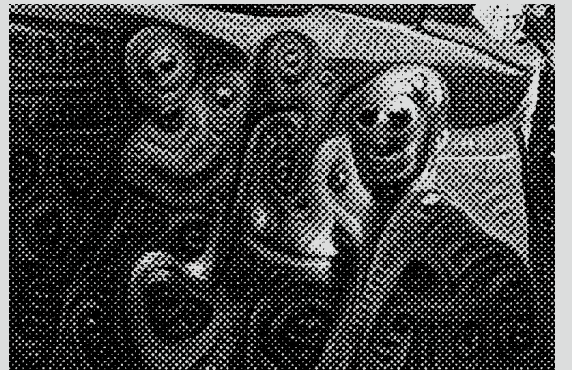
예술이 인공지능을 바라보려고 하는가'에 대한 이야기일 것이다. 이전부터 생각하는 기계, 인공지능을 다룬 소설이나 만화는 상당 수가 있었지만, 최근 인공지능에 대한 관심에 다시금 불을 지피게 된 건 '알파고' 때문일 것이다.

[그림 1] 알파고와 이세돌 9단의 대결 화면¹⁾



알파고를 동작시키는 인공지능 알고리즘은 새로운 게 아니다. 하지만 알파고는 이세돌 9단을 이겼고 이 모든 과정들은 다양한 매체를 통해 전 세계에 생중계되었다. 오랫동안 도전하다 많은 사람들이 포기해왔던 인공지능은 이렇게 몇몇 집념있는 사람들의 연구로 빛을 보고 전 세계에 알려졌다. 비슷한 시기에 기존의 사진을 [그림 2]처럼 재해석해주는 딥드림(deep dream)이 알려지고 테슬라(Tesla)는 자율주행 자동차를 판매하기 시작했다.

[그림 2] deep dream으로 만든 그림. 원제 : 수영장 속 3명의 사람(photograph of three men in a pool)²⁾



이건 마치 300kg 이상의 역기는 들 수 없다고 마음 속으로 한계를 못박았던 많은 사람들이, 누군가 305kg을 든 이후부터는 300kg 이상을 들 수 있게 된 것과 같다. 잘되지 않을거라 믿었던 인공지능 기술이 모든 것을 해결하는 상황이 돼 버린 것이다.

2005년 MIT 미디어 랩에서 인공지능의 아버지라고 알려진 마빈 민스키(Marvin Minsky)의 강연을 들은 적이 있다. 그 당시 민스키는 "인공지능에서 더 이상 가능성을 찾기 힘들어 보이고 규칙기반 추론(rule-based reasoning)이 유일한 방법처럼

글 | 송호준 songhojun@gmail.com

작가 송호준은 국한기술을 이용하고 구현하여 이야기를 만들어 내는 작업들을 하고 있습니다. <이 세상에서 가장 강력한 무기>, <방사능 보석>, <오픈 소스 인공위성 프로젝트>, 그리고 <100년에 한 번 깜박이는 LED> 등의 프로젝트를 진행하고 있으며 최근에는 경험과 우연을 증시하는 인공지능을 통하여 위대한 인간과 보통 인간의 구도를 깨트릴 수 있는 작업을 구상하고 있습니다.

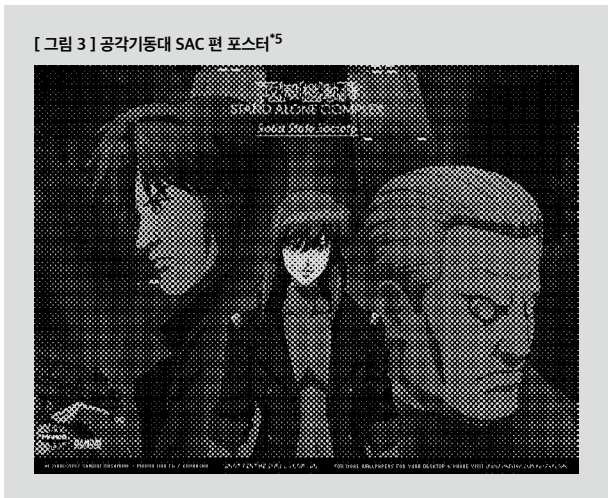
보인다"고 말했다. 이 방식은 극단적으로 말하면 모든 경우에 따른 답을 미리 입력해놓고 질문이 있으면 해당하는 답을 연결해 주는 방식이다. 다시 말해 어떤 추론이나 상상의 여지는 없어 보이는 방식의 인공지능이다. 그저 질문과 답의 쌍을 무수히 많이 저장해 놓은 것이다. 10여 년이 지나 알파고는 신의 수라 일컬어 지는 수를 두며 바둑으로 이세들을 이기기도 했다. 인간의 경험으로 도저히 이해할 수 없는 수를 두었다고 말한다. 규칙기반 추론에서는 불가능한 방식이다. 기계가 의식을 갖을 수 있을 것인가? 의식은 인간 고유의 것인가? 영혼이란 실재하는가? 등의 실존적인 질문에 대한 답변이 될지도 모르는, 역사상 가장 충격적인 이벤트가 전 세계에 중계된 것이다.

이와 더불어 SF에서나 가능하리라 여겼던 영생이나 뇌를 전자 장치의 일종으로 만드는 전뇌화(電腦化)가 이제는 정말 가능할 것 같아 보이기도 하고, 먼 훗날이 아닌 수십년 내에 일어날 것으로 예측되기도 한다. 일론 머스크(Elon Musk)는 오픈AI(OpenAI)와 기계와 뇌를 이을 뉴럴링크(neuralink)를 만들었고, 뉴욕에서 열린 Transion 2016 마케팅 컨퍼런스의 메인 주제도 인공지능이었다. 20년 후엔 인간 뇌의 능력을 가진 컴퓨터가 100만원대가 되고 2040년에는 컴퓨터가 의식을 갖게 될 것으로 예상되고 있다. 이것 역시 2017년 기준인데 인공지능의 발전 속도가 예상보다 빨라 인공지능 발전이 훨씬 앞당겨지리라 예측되는 상황이다.

예술에서 화두가 된 인공지능

세상이 인공지능의 가능성에 대해서 이야기 하기 시작했다. 예술도 마찬가지였다. 기존 공상과학(SF) 소설에서처럼 미완의 인공지능 기술을 바탕으로 상상력을 펼치던 예술과 달리 지금의 예술은 조금 더 직접적으로 인공지능을 다룬다. 노버트 위너(Norbert Wiener)의 사이버네틱스(cybernetics), 윌리엄 깁슨(William Gibson)의 뉴로맨서(neuromancer), 애니메이션 공각기동대, 영화 매트릭스(The Matrix) 등에서 이야기되는 전뇌화는 나노기술^{*3} 및 생명 공학의 발달과 최근 알파고로 다시 재조명되기 시작한 딥러닝(deep learning)을 통해 구현이 가능할 듯한 상황이다. 이렇듯 과거에 상상했었던 것들 중 많은 부분이 구체적으로 실체화 된 상황에서, 다양한 소재를 다루는 지금의 예술은 인공지능 기술 중 기계 학습(machine learning)을 직접 다루려고 시도하고 있다. 뉴욕대학교(NYU)의 ITP(interactive telecommunications program)에서는 실제 기계 학습(machine learning) 관련 수업들이 예술가들을 상대로 진행되고 있으며, 진 코건(Gene Kogan)은 예술가들 위한 기계 학습 웹사이트를 운영하며 예술 관련 학생들을 가르치고 있다^{*4}. 예술가들이 주로 사용하는 프로그래밍 플랫폼인

프로세싱(processing.org), 오픈프레임웍스(openframeworks.cc)에서도 인공지능 관련 응용 프로그래밍 인터페이스(API)들이 공개되었고, 실제로 간단하게나마 기계 학습 알고리즘으로 학습한 결과를 바탕으로 프로그래밍한 소프트웨어를 작업으로 선보이는 작가들도 늘어나고 있다.



인공지능에서도 화두가 된 예술

최근 작가들이 인공지능 기술을 따라 작업물을 내놓기 전부터 다양한 매체를 통해 이슈가 된 사례를 살펴보면, 인공지능 기술을 만든 곳에서 예술 작업을 만들어 내는 경우가 대부분이었다. 딥 드림의 이미지가 그 대표적인 예였고 램브란트(Van Rijn Rembrandt)의 기존 작업을 학습하여 새로운 램브란트 풍의 그림을 그려낸 'The Next Rembrandt' 프로젝트도 기술 분야의 주도로 진행된 프로젝트였다.

공학, 과학 분야에서 예술을 다루는 방식은 기존에 우리가 생각하던 '인간은 다른 동물과 달리 직관을 가지고 있다'라는 지점에 질문을 던지는 방식으로 이뤄진다. 작업과 더불어 설명을 통해 의도를 드러내려는 경향이 있는 예술 작업과 달리 기술 주도로 이뤄지는 예술 관련 작업들의 의도를 해석하기는 쉽지 않지만 위의 예들은 딱히 설명이 없어도 이해가 가능하다. 그림이라는 어찌 보면 너무나 익숙한 예술 형태를 취한 것만 보아도 그 의도를 파악하기가 어렵지 않다. 이쯤에서 흥미로운 건 지금의 예술은 오히려 인공지능을 다룰 때 기술에 집착하는 듯 하고, 기술 주도의 프로젝트들은 지난날 예술이 인공지능을 통해 말해 왔던 실존 문제들을 다루고 있다는 점이다.

예술가들은 인공지능 기술을 어떻게 이용하고 있는가

음악에서 기본 곡 작업 후 최종으로 소리의 균형을 잡는 작업을 마스터링(mastering)이라 한다. 기계 학습 기반의 LANDR^{*6}이라는 웹사이트를 통해서 마스터링을 할 수 있게 되었다. 기존에 마스터링 작업에서 창작자는 그들이 원하는 레퍼런스 음악을 함께 들어 가면서 그 곡과 비슷하게 자신의 앨범을 가다듬게 된다. 반면, LANDR은 원하는 분위기나 장르를 선택하고 작업한 곡을 업로드 하게 되면 기존에 학습된 다양한 곡들을 기반으로 업로드 된 곡의 소리 균형을 잡아준다. 예를 들어 내가 베이스 소리가 강한 레게 음악을 좋아한다면 그 곡과 비슷하게 내 곡의 베이스 소리와 인지하지 못한 부분까지 인공지능이 다듬어 준다. 심지어 최근에는 곡을 다듬는 수준을 넘어 곡을 만드는 상황에 이르렀다. AI 작곡 소프트웨어로 곡의 70% 정도를 만드는 사례가 알려지기도 했다.

보통 실험적으로 인공지능을 이용하는 소설 쓰기, 요리 만들기 등과 달리 음악에서는 인공지능 기술이 이미 응용되어 쓰이고 있다. 미술의 경우 특히 기술을 많이 사용하는 미디어 아티스트를 예로 들자면, 조금 더 직접적으로 기술을 이해하려는 경향이 있다. 자신들이 모은 자료들을 바탕으로 직접 컴퓨터를 학습시키고, 그 결과 적용되는 모습을 다양한 형태로 보여주려 한다. 학습된 자료를 바탕으로 새로운 풍의 사진이나 그림을 그리려는 작가들의 시도는 앞서 말한 기술 주도로 이뤄진 프로젝트에 비해 의미를 찾기 어려워 보인다. 하지만 예상되는 인공지능의 쓰임새를 달리하여 진행되는 경우도 있다. 예를 들어, 2016년 카일 맥도날드(Kyle Mcdonald)의 '우린 어떻게 같이 행동하는가(How we act together)'의 경우, 기계학습을 이용하여 학습된 사람들의 얼굴 표정을 바탕으로 시간의 흐름에 따라 비슷한 제스처를 취한 사람들끼리 연결해 준다^{*7}.

예술과 인공지능, 그리고 불확실함

우리는 현재 인공지능의 실체화를 목격한 시점에 살고 있다. 인공지능은 인간의 고유한 위대함과 직관에 대한 반증이다. 역사적으로 위대함이라는 요소가 필요한 예술은 과연 불확실성을 받아 들이려고 하는 지금의 인공지능을 어떻게 바라볼까? 우리가 어떻게 한국말을 잘하게 됐는지 아무도 모른다. 알파고도 왜 이세돌을 이겼는지 알수 없다.

다양한 경험과 학습을 통해 알 수 있는 것은 우리의 뇌 혹은 컴퓨터에서 조합된 결과 뿐이다. 창작자의 의도, 계획이 중요한 예술에 있어 불확실한 블랙박스를 있는 그대로 받아 들이려는 기계 학습의 방식은 개인보다 집단, 순간적인 영감보다 경험을 중시한다. 이렇듯 설명이 불가능한 인공지능과 영웅이 필요한 예술은 공존하기

힘들어 보인다. 새로운 사고 체계가 요구되는 시점에서 예술가들 역시 알파고에 버금가는 질문을 던질 수 있을까?

^{*1} 참고 | <http://www.popularmechanics.com/technology/ai/19863/googles-alphago-ai-wins-second-game-go/> ^{*2} 참고 | [https://en.wikipedia.org/wiki/DeepDream#/media/File:Deep_Dreamscope_\(19822170718\).jpg](https://en.wikipedia.org/wiki/DeepDream#/media/File:Deep_Dreamscope_(19822170718).jpg) ^{*3} 참고 | nanometer: 1×10⁻⁹ m는 세포와 전자 장치가 서로 연결되어 정보를 전달하기 위해서 필요한 해상도로 알려져있다. ^{*4} 참고 | <https://ml4a.github.io/> ^{*5} 참고 | https://en.wikipedia.org/wiki/Ghost_in_the_Shell:_Stand_Alone_Complex#/media/File:Gits_SAC_complete_collection_cover.jpg ^{*6} 참고 | <https://www.landr.com/> ^{*7} 참고 | <https://hwat.schirn.de/>

"X의 목적은 통찰이지 Y가 아니다"

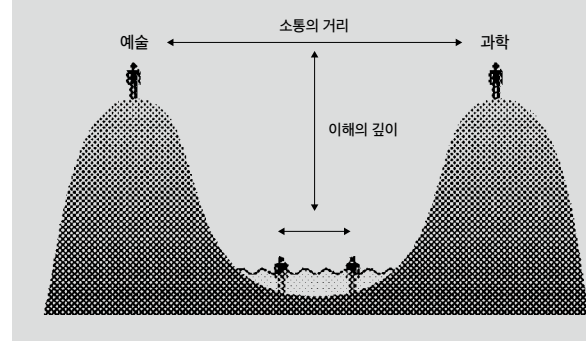
예술은 한 때 예술이 아니라고 생각되던 것(nonart)을 예술(art)로 포섭하며 자신의 지평을 확장하는 경향이 있다. 어떤 새로운 기술이 인류사에 등장했을 때마다 이 동시대 기술로 창작에 관한 새로운 실험을 하는 사람들이 나타났고, 그들의 작품이 어떤 과정을 거쳐 예술 비평의 대상이 되다 보면 어느새 예술은 이전 보다 자신의 색을 다채롭게 하고 경계를 확장한다. 하지만 결과적으로 보면 그 과정이 순탄하지 않을 때가 많고 그 순탄하지 않음이 오히려 예술에 관한 이야기를 더 흥미롭게 만든다.

현대 예술의 한 켠에 불확실한 영토를 확보한 뉴미디어 아트에도 그런 흥미로운 이야기들이 있다. 인공지능과 머신러닝을 창작자가 획득할 수 있는 동시대 기술로 보고, 전문가들 간 소통과 협업 이슈를 예술과 과학사이에서 발생하는 상황에 빗대어 이야기를 시작해 보자.

예술과 과학 사이 각도가 만드는 그림자

미디어 아티스트로 활동할 때, 예술과 과학의 만남을 주제로 삼는 '이벤트'에 참여했던 경험이 몇 번 있었다. 이 경험들은 '애프터 없는 소개팅' 같이 이벤트 후에 지속가능한 관계로 이어지지 않는 경우들이 많았다. 왜 그랬을까 돌이켜 생각해보면, 보통 짧은 일정 속에서 결과물을 만들어 내야 했기 때문에 서로의 전문 분야를 이해할만한 충분한 시간이 없었기 때문이다. 그리고, 서로의 전문 분야에 대한 이해 없이 각자 자기 분야의 압축되어 있는 용어를 사용하며 소통을 시도하는 실책을 범했다. 서로가 상처를 주고 받으며 어떻게든 마무리를 해내더라도, 그 작업이 끝나고 난 이후 지속적으로 뭔가를 함께 하기에는 그 관계가 서먹했었던 것 같다.

[그림 1] 예술과 과학 사이, 그리고 둘 사이의 이해의 계곡

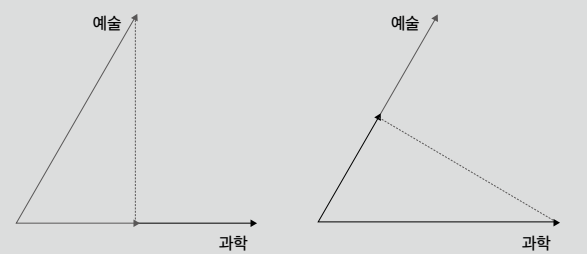


다른 분야 전문가와 함께 무언가를 할 때, 왜 우리는 각자의 전문 분야의 등성이에서 서로를 향해 움직이지 않은 채 소통을 시도하는 것일까? 왜 우리는 계곡을 내려가 발이 닿는 찰박찰박한 시냇가에서 함께 물장구를 치며 목적없이 놀아보는 경험은 지속하려고 하지 않았던 것일까? 자신의 분야에 관한 높은 수준의 이해는 담지 못하더라도 소통을 하기 위해 서로의 눈높이에 맞춰서 이야기를 나누고, 발이 닿는 지점에서 이런 저런 실험과 모험을 꾸준히 반복하다 보면 자연스럽게 더 깊은 물에 도전할 수 있는 용기와 실력을 키울 수 있었을텐데. [그림 1]에서처럼 어느 분야라도 높이 올라가면 첨단의 내용은 사뭇 다르지만 뿌리 쪽은 서로 연결돼 있거나 공통 부분을 가지고 있는 경우가 많고, 이 공통 부분은 허우적거리며 헤엄치다가 힘들어질 때 지지할 수 있는 발판이 되어줄 수 있다.

이 맥락에서 지난 6월에 요아브 골드버그(Yoav Goldberg)의 자연어의 적대적 생성에 관한 적대적 리뷰(an adversarial review of adversarial generation of natural language)로 촉발된 논의들 중에 페이스북 인공지능 연구소의 디렉터인 얀 르쿤(Yann LeCun)이 한 이야기¹⁾가 인상에 강하게 남았다.

만약 커뮤니티를 서로 방향이 다른 단위벡터로 가정한다면, A벡터를 B벡터 위에 투영했을 때 A벡터는 B벡터 보다 짧아진다. 그러므로 A는 B보다 열등하다고 생각할 수 있다. 그러나 반대의 경우도 마찬가지로 B벡터를 A벡터 위에 투영하면 B벡터는 A벡터 보다 짧아진다. 커뮤니티들이 공통의 언어를 개발하고 서로의 방법들 중 가장 좋은 것을 받아들이는데는 시간이 걸린다.

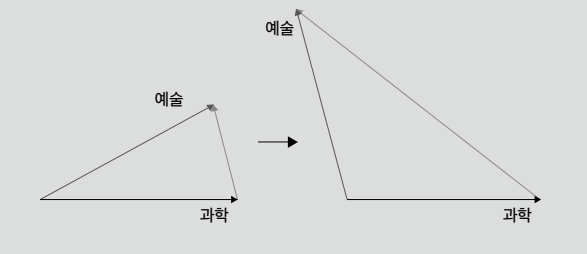
[그림 2] 두 전문 분야 벡터의 상호 투영



이 글을 읽은 사회²⁾를 적다 보니 서로 다른 분야의 전문가가 원활한 소통을 하기 위해 상대방에게 제안하는 의견의 경우 각자의 관점에서 보기에는 당연히 부족함이 있을 수 있단 생각이 들었다. 그러한 부족함에도 불구하고 서로 상호 작용과 소통을 지속해 나간다면 그 과정에서 무엇인가가 일어날 수 있지 않을까 하는 기대도 갖게 됐다.

그런데 만약 과학에서 출발해 예술을 배워 나가거나 아니면 반대로 예술에서 출발해 과학을 배워 나간다면 어떻게 될까? 그 사이를 연결한 벡터는 원래의 벡터들이 서로 더 벌어질 때 더 길어진다. 그러므로 예술과 과학이 찰박찰박한 지점에서 만나 가깝게 이어진 후에는 그 이어진 상태를 유지하면서 각도를 조금씩 늘리는 방법으로 더 긴 제 3의 벡터를 만들 수 있다.

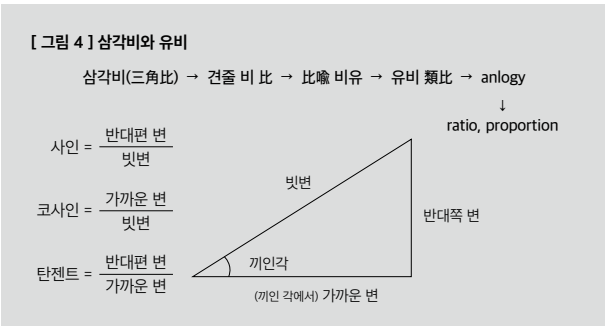
[그림 3] 두 전문 분야가 이어져 있는 상태에서 서로 더 다룰 때



얀 르쿤이 단위벡터로 빗대어 설명한 것에서 비유와 유비(類比, analogy)를 더 이어 나가다 보면 삼각비(三角比)를 연상하게 되는데, analogy의 어원에 비 또는 비율이 있어서 재밌다. 나에게 가까운 변을 과학으로 하고 반대 쪽으로 멀리 있는 변을 예술로 한다면(또는 그 반대) 그 사이에 비스듬하게 세워진 빗변은 무엇일지 궁금하다.

글 | 최승준 erucipe@gmail.com

미디어아티스트로 활동했고 대학에서 인터랙션 디자인을 가르치다가 최근에는 유치원을 운영하며 구성주의 교육에 머신러닝을 접목하는 시도를 하고 있습니다. 인간에게서 배우는 기계의 학습과 기계의 학습에서 배우는 인간의 학습에 관심이 많습니다.



무엇인가의 차이를 늘리고 줄이는 과정은 학습과 관련된 부분이 많은 것 같다. 특히 뉴럴 네트워크에 관해 공부를 하다 보면 두 벡터가 얼마나 다른지를 측정하는 척도로 코사인 유사도(cosine similarity)나 내적(dot product)이라는 개념을 만나게 된다. 뉴럴 네트워크를 도구로 획득하기 위한 오솔길 탐색을 시작하는 지점에서, 예술가들은 이공계 분야 배경을 가진 사람에게 이미 익숙한 이 개념들을 새롭게 볼 수 있는 기회를 얻고, 다른 배경을 가진 사람들은 이제부터라도 선형대수를 진지하게 대면할 수 있는 기회를 얻는다.

이야기를 더 이어 나가기 전에 비유와 모형이 가지는 한계를 살펴 보고자 한다. 더글러스 호프스태터(Douglas R. Hofstadter)의 괴델, 에셔, 바흐³의 표지에 등장하는 입체를 세 가지 방향에서 만들어진 G, E, B의 그림자로 미루어 연상하는 것은 사람에 따라 꽤 생각이 필요하며 시간이 걸리는 일이다. 이해를 쉽게 하기 위해서 빗대어 표현한 개념들, 즉 비유를 사용한 개념들은 그림자와 닮아 있어 거기에 담겨진 정보가 원래의 정보보다 줄어든 때가 있다. 이러한 점을 고려해보면, '과연 예술이나 과학을 벡터에 빗대어 표현하는 것이 얼마나 합당할까?' 라는 의문이 든다. 마찬가지로 모형도 어떤 현상을 그 현상의 중요한 특징들로 추상화했기 때문에 유용하게 다룰 수 있긴 하지만, 모형으로 만드는 과정에서 편향⁴이 필연적으로 발생하기도 하고, 상대적으로 중요성이 떨어지는 일부 내용은 미처 담지 못하기도 한다. 차원 축소 덕분에 소통(전문 분야의 봉우리에서 내려와서)을 할 수도 있고 복잡한 다양체(manifold⁵)와 같이 시각적으로 떠올리기 어려운 높은 차원의 내용을 다양한 관점의 낮은 차원에서 파악한 그림자(또는 잠재공간 - latent space)들로 미루어 짐작하고 이해하는 시도를 할 수도 있다. 그러나 이 접근이 유발할 수 있는 잠재적인 오해와 위험은 늘 조심해야 한다.

물론 무엇인가가 딱 맞아 떨어지지 않고 어긋나 있는 현상은 예술이 잘 다루는 주제다. 존 버거(John Berger)는 '다른 방식으로 보기'⁶에서 르네 마그리트(Rene Magritte)가 언어와 시각 사이의 영원한 어긋남을 '꿈의 열쇠(la clef des songes)'라는 그림 속에 표현했다고 말했다. 그래서 이 글에서도 위험을 감수하며 어긋날 수도 있는 태도를 계속 유지해볼까 한다. 그러므로 이 글을 읽는 독자들 또한 그런 뉘가 어긋나 있는 부분이나 각 전문 분야에서

보면 짧고 부족할 수 있는 내용에 주의를 해가면서 비판적인 태도로 읽어주길 바란다.

벤 라포스키와 만프레드 모어의 후예들

컴퓨터를 활용한 창작의 역사를 살펴보면 1950년대에 오실로스코프(oscilloscope)를 가지고 작업했던 수학자 벤 라포스키(Ben F. Laposky)⁷의 작업을 발견하게 된다. 그의 작업은 컴퓨터를 활용했다기 보단 전자 회로를 활용해서 추상적인 이미지를 표현한 것인데 Electronic Abstractions라는 이름의 순회 전시를 한 벤 라포스키의 경우 수학에서 예술로 향한 벡터의 사례라고 생각해 볼 수 있다.

만프레드 모어(Manfred Mohr)⁸는 1960년대부터 프로그래밍 언어인 포트란(FORTRAN)을 활용해서 추상적인 이미지를 표현하는 작업을 했다. 만프레드 모어는 추상 표현주의 작가이자 재즈 뮤지션이었다. 세계 최대의 컴퓨터 그래픽스 행사인 SIGGRAPH(Special Interest Group on GRAPHics and Interactive Techniques)에서 평생 공로상을 수상하기도 한 만프레드 모어는 예술에서 컴퓨터 과학으로 향한 벡터의 사례로 생각해 볼 수 있다.

예술과 기술에 관한 흥미로운 이야기가 궁금하다면, 프로세싱⁹의 개발자 중 한 명인 케이스 리스(Casey Reas, @REAS)의 '미래 그리고 예술과 기술의 1965년부터의 역사(History of the future, art & technology from 1965)'¹⁰ 발표 영상을 보길 추천한다.

벤 라포스키와 만프레드 모어 둘 다 동시대 기술을 적극적으로 창작에 활용했다는 공통점이 있다. 그렇다면 요즘의 동시대 기술인 '머신러닝'을 창작에 활용하는 사람들은 어떤 부류의 사람들일까?

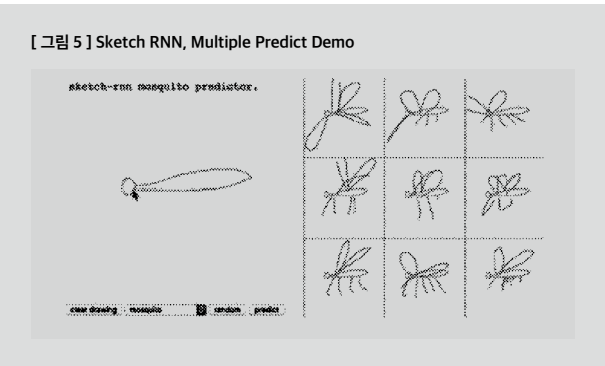
뉴미디어 아트와 가장 유명한 축제인 아르스 일렉트로니카(Ars Electronica)의 2017년 주제는 'AI 또 다른 나(AI the other I)'¹¹였다. 올해 출품작 중에는 2016년 머신러닝의 화두를 대표하는 키워드 중 하나인 GAN(generative adversial network)을 활용한 작업이 있다. 바로 마이크 타이카(Mike Tyka, @mtyka)¹²의 상상 인물의 초상화(Portraits of imaginary people)¹³란 작업이다(※ 영문 이름 뒤에 나오는 '@~'는 해당 인물의 트위터 아이디다). 마이크 타이카는 바이오 피직스 분야의 박사로 2009년에 버닝맨 페스티벌¹⁴에 참여하여 소형 작업을 한 것이 계기가 되었는지, 그 이후로 단백질 구조 등을 조형물로 표현하는 작업을 하며 예술 분야에서 활동한 경험이 있다. 그는 2015년에 구글 리서치 블로그에서 발행한 Inceptionism: going deeper into neural networks¹⁵를 기고한 세 명 중의 한 명이기도 한데, 이 글은 2015년 당시 나의 SNS의 타임라인을 '딥 드림(Deep Dream)'에 관한 실험들로 범람하도록 하는데 기여했다.

다른 두 명인 알렉산더 모드빈체프(Alexander Mordvintsev)와 크리스 올라(Chris Olah, @ch402)는 최근 distill.pub에 특징 시각화(feature visualization)¹⁶라는 제목으로 뉴럴 네트워크가 이미지를 어떻게 인식하는지에 관한 내용을 총 망라한 글을 출판했다. 알렉산더 모드빈체프, 마이크 타이카 그리고 크리스 올라의 이야기는 와이어드(Wired)지의 딥 드림의 내부(INSIDE DEEP DREAMS)¹⁷라는 글에서 관련 일화를 조금 더 자세히 살펴볼 수 있다. 그 밖에도 올해 아르스 일렉트로니카 출품작 중에는 마리오 클링게만(Mario Klingemann, @quasimondo)¹⁸의 X degrees of separation¹⁹이 있는데, 클링게만은 구글의 문화 예술 연구소가 수집한 같은 이름의 예술 유물 사진과 자신의 작품 사이의 시각적인 연관성을 재미있게 보여주는 작업을 구글의 예술 실험실에 올렸다²⁰.

마리오 클링게만은 자신의 습작을 트위터에 꾸준히 올리곤 하는데, 최근의 GAN을 응용한 습작들에 관한 이야기를 '뉴로그래퍼'가 인공지능에 예술을 집어넣다(A neurographer puts the art in artificial intelligence)²¹라는 제목으로 와이어드지에 실리기도 했다. 와이어드지는 예술을 다루는 매체가 아닌 점을 상기해 가며 읽어보길 권한다.

예술가를 위한 머신러닝 강의와 워크숍을 하며 전 세계를 다니고 있는 진 코건(@genekogan)도 올해 아르스 일렉트로니카에 작업²²을 출품했다. 진 코건은 올해 Eyeo 페스티벌에서도 굶주린 컴퓨터 과학자, 머신러닝으로 어떻게 한 폰도 벌지 않을 수 있는지 알려주는 튜토리얼(Starving computer scientist, a tutorial on how to make no money from machine learning)²³이란 머신러닝과 창작에 관한 발표를 했다. 이들 외에도 데이빗 하(David Ha, @hardmaru), 카일 맥도널드(Kyle McDonald, @kcimc), 메모 악탄(Memo Akten, @memotv), 레베카 피에브링크(Rebecca Fiebrink, @RebeccaFiebrink), 루바 엘리엇(Luba Elliott, @elluba), 요탐 만(Yotam Mann, @yotammann), 사뮈 위니거(Samim Winiger, @samim) 등 머신러닝을 창작에 활용하고 있는 다양한 창작자와 연구자들이 있는데, 이 사람들 중 하나의 트위터 계정만 팔로우 해도 꼬리에 꼬리를 물고 머신러닝을 활용하고 있는 창작자들을 더 많이 발견할 수 있다.

이 중 데이빗 하의 이력이 특히 흥미롭다. 어떻게 AI가 음악과 예술을 재구성하는 단위를 만들 수 있는지(How A.I. is creating building blocks to reshape music and art)²⁴에 관한 뉴욕타임즈 기사를 보면, 그가 동경의 골드만삭스에서 트레이더로 일했던 경력이 나온다. 그러다가 자신의 블로그²⁵에 올린 뉴럴 네트워크를 활용한 창작 작업이 알려지면서 구글 브레인팀의 레지던시도 하고 마젠타 팀에 합류하여 Sketch RNN²⁶등의 작업을 했다.



이들이 가지고 있는 배경을 조금 찾아보면, 대부분의 경우 수학이나 컴퓨터 과학을 전공했거나 프로그래밍 리터러시(literacy, 읽고 쓰는 능력)를 가지고 일찌감치 코드로 창작 활동을 해오다 다시 동시대 기술인 머신러닝을 도구로 획득해서 창작 활동을 하고 있는 패턴에 대해 확인할 수 있다. 그렇기에 이들은 아무래도 벤 라포스키의 후예들에 조금 더 가까운 것이 아닐까 생각하게 된다. 그렇다면 만프레드 모어의 후예들은 요즘 무엇을 하고 있을까?

컴퓨터를 활용한 예술 작업의 초창기에는 수학자나 컴퓨터 과학자 중에 창작에 관심이 있는 사람들이 작업을 했는데, 기술의 문턱이 낮아지고 인터페이스의 접근성이 좋아지면서 더 많은 사람들이 이런 방식의 창작의 흐름에 참여했다(물론 아직 머신러닝의 기술 문턱과 인터페이스는 아직 더 많은 사람들을 포용할 수 있는 단계는 아닐지도 모른다). 21세기에 이르러 프로세싱 등 기술의 문턱을 낮추고 접근성을 높게한 도구들이 등장하면서, 코드를 활용해 창작을 하는 디자이너와 예술가들은 폭발적으로 늘어났다. 아직 이 방향에서 접근하는 창작자들 중 머신러닝을 도구로 획득한 경우는 상대적으로 드물지만 이 창작자들이 다룰 수 있는 기술의 난이도와 문턱이 어떤 변곡점을 넘으면 다시 만프레드 모어의 후예 같은 사람들의 활동 또한 매우 왕성해지며 우리가 그 동안 보지 못했던 새로운 창작의 영역들을 개척해내리라 예상해 본다.

구글의 예술가와 기계지능(artists and machine intelligence, AMI)²⁷프로그램을 이끌고 있는 블레이즈 아구에라(Blaise Aguera, @blaiseaguera)는 '컴퓨터가 창의력을 배우는 방법'²⁸이라는 TED강연에서 딥 드림의 메커니즘을 대중이 이해할 수 있도록 쉽게 풀어주면서 미켈란젤로(Michelangelo)를 언급한 바 있다.

마지막으로 저는 미켈란젤로의 생각이 옳았다고 생각합니다. 인식과 창의성은 매우 밀접하게 연결되어 있습니다. 지금까지 보신 것은 세상의 많은 사물들을 식별하거나 인식할 수 있도록 학습한 뉴럴 네트워크로 이 뉴럴 네트워크를 거꾸로 적용하면 그 사물들의 이미지를 생성해 낼 수 있습니다. 제가 이것을 보고 느낀 점 중에 하나는 미켈란젤로만 돌덩이 안에 있는 조각상을 본 것이 아니라 인식행위를 할 수 있는 어떤 생물이나 존재, 심지어 외계인이라도 인식 행위를 할 수 있으면 창작할 수 있다는 것입니다. 인식하는 것과 창작하는 것은 같은 작동 원리를 가지기 때문이죠.

그런데 블레이즈는 AMI의 다른 영상^{*29}과 머신러닝 시대의 예술(Art in the age of machine intelligence)^{*30}에서는 대중이 쉽게 알만한 미켈란젤로의 말이 아니라 발터 벤야민(Walter Benjamin, 기술복제 시대의 예술작품)^{*31}이나 빌렘 플루서(Vilem Flusser)^{*32}와 같이 미디어에 관해 중요한 이야기를 한 철학자와 데이빗 호크니(David Hockney)^{*33}같은 예술가의 말을 인용한다. 이를 통해 머신러닝을 창작의 도구로 삼는 방향의 접근이 예술의 어떤 맥락과 관련이 있는지 짚으려는 조금 더 본격적인 시도를 한다.

이 관점은 우리가 예술을 혼종(hybrid)의 존재에 의해 생성 되기도 하고 소비되기도 하는 무엇으로 다시 생각하게 합니다. 예술적 생산에 관련된 기술은 예술과 '다른'것이 아니라 그 '일부'인 것입니다. 미디어 철학자인 빌렘 플루서가 말했듯이 "도구는 보철 치아, 손가락, 손, 팔, 다리 등 인체 기관의 연장입니다." 붓이나 곡괭이 같이 산업 시대 이전의 도구가 인체의 생체역학을 확장했듯이, 보다 정교한 기계는 인간의 정보 세계와 생각을 보철적(prosthetically)으로 확장합니다. 따라서 카메라를 포함한 모든 장치(컴퓨터뿐만 아니라)는 인공지능입니다.

여기서 도구는 '미디어의 이해'^{*34}의 저자 마셜 맥루한(Marshall McLuhan)이나 사이버네틱스^{*35}의 제창자이자 '인간의 인간적 활용'^{*36}의 저자인 노버트 위너(Norbert Wiener)가 말한 것 처럼, 우리 자신과 피드백 루프를 형성하며 존재를 보다 확장하고 결국 이전과는 확연히 다른 존재로 변화하도록 안내하는 그런 도구일 것이다.

블레이즈는 학창 시절 물리학을 공부하고 마이크로소프트를 거쳐(Seadragon과 Photosynth를 만드는데 기여했다)^{*37} 현재 구글의 AMI를 이끌고 있다. AMI에서 A가 예술(Art)이 아니라 예술가(Artist)인 점이 재밌다. 혹시 그가 과학의 벡터에서 예술의 벡터로 그림자를 내리고 자꾸 예술에 관한 이야기를 언급하면서 예술을 배우고, 예술가와 친해지며, 예술에 관해 실험하는 재미있는 모험을 진행하는 중이 아닐지 추측해 본다. 그리고 이런 부류의 사람들이 자신이 학습한 것을 조금 더 쉬운 언어로 풀어내 더 많은 사람들을 찰박착박한 시넷가로 초청하고 함께 놀면서 그들의 도전과 학습을 돕는 도움계단을 만들고 있는 것은 아닐까?

한편 지금까지 언급한 맥락, 즉 머신러닝을 활용해 창작의 문턱을 낮추거나 창작의 방식을 재구성하는 방향과 일맥상통하긴 하지만 약간은 다르게 접근하고 있는 사례로 어도비(Adobe)의 연구가 있다.

어도비 센세이와 메맥스

알파고 제로에 관한 뉴스^{*38}가 타임라인을 채우던 날 어도비는 자사의 맥스 2017 키노트^{*39}에서 어도비 센세이(Adobe Sensei)^{*40}를 시연했다. 어도비는 시연에서 자연스러운 상호작용(natural interaction), 콘텐츠 지능(content intelligence), 디자인 지능(design

intelligence) 세 가지를 강조했으며, 인간이 프로그래밍 언어 등을 학습해 도구인 기계 쪽으로 다가가서 상호작용 했던 과거와는 달리, 기계가 인간 쪽으로 훌쩍 더 다가와서 상호작용함에 따라 펼쳐지는 창작의 새로운 방향과 가능성을 보여줬다. 이 시연을 보고 느낀 점을 소셜네트워크 서비스^{*41}에 적어 두기도 했는데, 시연은 다음과 같은 내용으로 구성되어 있다.

1. Natural interaction (자연스러운 상호작용)

1.1. 기본 아이디어를 담은 스케치를 불러오면 스케치 안의 구성요소의 의미를 인식한다. 이 때 음성으로 'find images based on my sketch'라고 입력하면, 우주와 여성 등 스케치 안에 등장하는 구성요소에 해당하는 이미지를 스톱 이미지 및 보유하고 있는 이미지에서 찾고 우주가 배경이 되고 여성이 전경에 등장하는 포스터를 만드는 과정을 시작한다.

2. Content intelligence (콘텐츠 지능)

2.1. Adobe Stock에서 관련이 있는 우주 배경을 가지고 온 다음, 가져온 이미지 우하단에 생긴 Sensei 버튼을 클릭한 후 그 우주 이미지를 조금 더 탐색한다. 우주의 하위 개념과 그에 해당하는 이미지의 구성 요소들을 인식하며 시각화 되고 상호작용 가능하게 된다.

2.2. 인식한 요소인 별에서 별들의 밝기와 밀도 그리고 크기를 조정해서 스톱 이미지 중에 원하는 이미지를 찾고 배경으로 가져온다.

2.3. 가지고 있는 이미지에서 찾아서 제안한 여성 이미지를 배치한 후 Sensei 버튼을 클릭하면, 역시 그 이미지 안에 있는 특징들을 추출하고 자체히 조작할 수 있는 화면으로 넘어간다.

2.4. 인식한 얼굴의 특징을 선택한 후 left/right를 클릭하면 얼굴의 위치는 고정된 채, 가지고 있는 인물의 다양한 사진들 중 고개를 왼쪽에서 오른쪽으로 돌린 것들을 탐색한 후 원하는 이미지를 고를 수 있다. 완전히 옆으로 돌린 얼굴까지 깔끔하게 인식한다. 이 외에도 up/down, eyes, smiles 등의 인터페이스를 이용해 다양하게 조정할 수 있다.

2.5. 그렇게 가져온 인물 이미지는 역시 클릭 한번으로 인물 이미지의 배경을 제거하고, 아가 찾아둔 우주 이미지를 배경으로 하는 장면에 합성하고 원하는 위치에 배치한다.

3. Design intelligence (디자인 지능)

3.1. 여기서부터는 미리 준비한 위 작업에서 조금 더 발전시킨 영화 포스터 디자인 시안을 바탕으로 이어간다. 이 때 중요한 것은 창작자가 이전에 작업한 스타일을 학습한 템플릿을 제공하고 이전에 작업한 것과 비슷한 스타일의 작업 요소를 제안하고 지원한다.

3.2. 과거에 이런 작업 맥락에서 사용했던 제목 폰트 중에 하나를 고를 수 있도록 제안하고 이전에 작업했던 제목 텍스트 레이아웃을 제안하며 손쉽게 작업 시안의 공간을 탐색할 수 있도록 돕는다.

3.3. 지금까지는 포토샵에서 작업을 한 것인데, 앞서 한 작업을 토대로 모바일 타겟의 작업을 하기 위해 UI 프로토타이핑을 할 수 있는 XD(http://www.adobe.com/products/xd.html)로 넘어간다. (이미 아이패드에 대응하는 작업은 있는 상태) 그리고 음성입력으로 아이폰 8에 대응하는 레이아웃을 제안하라고 명령하면, 고를 수 있는 시안을 보여준다.

3.4. 그 시안 중에서 마음에 드는 것을 고르면, 원래 작업했던 아이패드 앱용 레이아웃을 아이폰 8에 대응하는 시안으로 제공한다. 기존 프로토타이핑 (시안)작업에서는 타겟 디바이스에 따라 화면비, 해상도 등에 따라 조정이 필요한데 이를 해결한 것이다. 또한 사용자는 아이폰 8으로 시안을 바로 작동해 보고 다시 조정할 수 있다.

3.5. 어도비 센세이는 창작자가 작업하는 모든 과정을 추적하고 학습한다. 지금까지의 최종 시안에 나오는 Sensei 버튼을 누르면 처음 스케치 부터 지금까지 창작자가

선택하고 결정한 모든 과정을 나무 구조로 조망해 볼 수 있다. 이 때 그 나무구조를 Creative Graph라고 부른다.

3.6. Creative Graph에서 창작 과정 중의 한 지점을 다시 재방문해서, 예를 들면 아가 얼굴의 왼쪽/오른쪽을 골랐던 작업 과정을 재방문하고, 다른 각도의 얼굴 사진을 손쉽게 그래픽 상의 UI로 선택하면, 최종 시안에 바로 반영이 된다(앞서 얼굴 사진에 적용했던 창작 과정들은 자동으로 다시 적용된다).

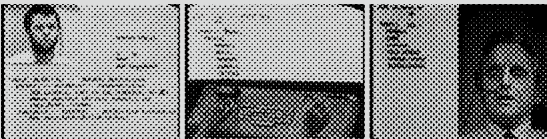
3.7. 심지어 처음 스케치를 입력했을 때 인식했던 요소 중 여성을 남성으로 바꾸면, 이 가상의 영화 포스터에 등장하는 인물 사진은 남자 주인공의 사진으로 바뀌며 앞서 작업했던 스타일/창작과정을 적용한다.

3.8. 그렇게 만들어낸 두 개의 시안을 담은 아트보드를 보여주며 데모를 마친다.

이 시연과 AMI에 패트릭 헤브론(Patrick Hebron, @PatrickHebron)이 기고한 머신러닝 시대에 디자인 도구를 다시 생각해 보기^{*42}는 많은 관련^{*43}이 있다. 패트릭 헤브론은 이 글에서 발현하는 특징들(emergent feature sets), 탐험을 통한 디자인(design through exploration), 기술에 의한 디자인(design by description), 절차를 조직하기와 대화형 인터페이스(process organization and conversational interfaces) 등의 개념을 짚었고, Creative AI^{*44}의 사밌이 쓴 기계의 도움을 받는 글쓰기(assisted writing)^{*45}에서는 근미래의 워드프로세서가 가질 10가지 가능성(추천 recommend, 요약 summarize, 단순화 simplify, 변형 morph, 전이 transfer, 예측 predict, 생성 generate, 번역 translate, 협력 collaborate, 인터페이스 interface)을 짚었는데 두 글 다 머신러닝에 의해 확장될 도구가 어떤 형태가 될 지 상상해 보는데 도움이 된다.

사밌은 본문에 더글러스 엥겔바트(Douglas Engelbart)의 1968년 모든 데모의 어머니^{*46}에 등장하는 NLS(oNLine System)^{*47}의 사진을 인용하며 개념 공간에서의 생각 벡터("Thought vectors in concept spaces."—Douglas Engelbart)라고 적어두었지만, 아쉽게도 더글러스 엥겔바트가 진짜 이런 말을 했다는 참고 자료는 찾을 수 없었다. 생각 벡터는 오히려 제프리 힌트(Geoffrey Hinton)이 사용해서 더 잘 알려진 용어인 것 같다.

【 그림 6 】 모든 데모의 어머니에서 시연한 NLS



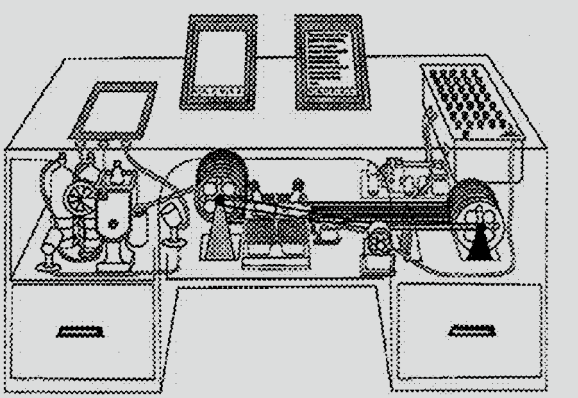
그러므로 사밌의 글은 비판적인 태도로 읽을 필요가 있지만 테드 넬슨(Ted Nelson)의 하이퍼텍스트 개념이나 더글러스 엥겔바트의 인간 지능의 증강^{*48}(또는 지능의 증폭, intelligence amplification, IA^{*49}) 개념과 요즘의 인공지능, 머신러닝의 전개를 연결지어 풀이해 본 시도는 재미도, 의미도 있다고 생각한다.

더글러스 엥겔바트는 바니바 부시(Vannevar Bush)^{*50}의 생각에 영향을 많이 받았는데, 컴퓨터가 단순히 계산기계가 아니라 정보를 처리하고 인간과 긴밀하게 상호작용할 수 있는 존재가 되리란 전망을 담은 우리가 생각하는 대로(As we may think)^{*51}라는 글을 디 아틀란틱(The Atlantic)에 1945년 7월과 9월에 기고했고 거기에는 다음과 같은 내용^{*52}이 있었다.

점점 많은 연구가 진행되고 있다. 그러나 오늘날 분야별 전문화가 확장되면서 과학은 확실히 정체되고 있는 것 같다. 연구원은 다른 수천 개의 연구물과 연구결과에 좌절하게 된다. 연구 결과가 나오면, 기억하기에는 너무 많고, 이해할 시간도 부족하다. 그러나 발전을 위해 전문화는 점점 더 필요하며, 각 분야의 연구를 연결하려는 노력은 피상적인 것이 되어 버린다. 연구 결과를 전송하고 검토하는 전문적인 연구 방식들은 이미 낡은 것일 뿐 아니라, 지금까지의 연구 목적에도 전혀 맞지 않는다. (중략) 하찮은 많은 것들 속에 파묻혀 정작 중요한 업적이 잊혀지는 불행은 지금도 반복되고 있다. 문제는 다양한 관심을 다루는 출판이 오늘날 과도하게 이루어지고 있다는 것이 아니라, 오히려 기록된 것을 유용하게 하는 우리의 현재의 능력을 출판이 훨씬 넘어서고 있다는 것이다. 인간 경험은 대단히 빠른 속도로 증가하고 있으나, 미궁을 뚫고 빠져나가 중요한 아이템으로 나아가기 위해 사용하는 방법은 뚝단배를 타고 다니던 시절과 동일하다.

1945년 8월에 일본에 핵폭탄이 투하되면서 2차 세계대전은 종전으로 치달았고, 루즈벨트(Franklin D. Roosevelt) 대통령의 과학자문이자 맨하튼 프로젝트의 일원이었던 바니바 부시는 기술이 가지는 파괴적인 측면을 우려했다. 그래서 우리가 방대한 정보를 보다 잘 다루게 된다면 조금 더 이성적으로 행동할 것이라는 기대를 가지면서 정보를 검색하고 처리할 수 있는 메멕스(Memex)^{*53}라는 기계에 관한 아이디어를 제시했고 더글러스 엥겔바트는 그 영향^{*54}으로 NLS에 관한 아이디어를 구체화했다.

【 그림 7 】 메멕스(Memex)



즉, 이미 1940년대에도 인류가 축적한 정보는 이미 과잉 상태였고, 우리는 그 정보를 잘 다룰 수 있는 도구가 필요했다. 여기에서 조금 더 나아간 개념이 바로 1960년의 J. C. R. 리클리라이더(J. C. R. Licklider)가 쓴 인간-컴퓨터 공생(Man-computer symbiosis)^{*55}이다. 본문의 소재목만 읽어도 흥미로운데 요약에서 언급한 내용은 다음과 같다.

인간의 두뇌와 계산 기계는 매우 가깝게 결합할 것입니다. 그리고 그 파트너십의 결과로 우리가 오늘날 알고 있는 정보를 다루는 기계 없이 데이터를 처리하거나 생각하는 인간의 두뇌는 없을 것입니다.

1.2 항목의 기계적으로 확장된 인간과 인공지능 사이(Between "mechanically extended man" and "artificial intelligence")라는 제목도 재미있지만 3항목인 실시간의 형식화 사고를 위해 컴퓨터를 참여시킬 필요(Need for computer participation in formulative and real-time thinking)부터 다시 메멕스와 관련된 내용이 나온다.

나는 생각하는 시간의 85퍼센트를 생각하는데 필요한 자세를 취하고, 무엇인가를 결정하고, 내가 알아야 할 필요가 있는 것을 배우는데 씁니다. 정보를 소화하는데 드는 시간 보다 정보를 찾고 모으는데 드는 시간이 훨씬 많이 듭니다. 그래프를 직접 그리거나 조수에게 그래프를 그리는 방법을 가르치는데도 많은 시간이 듭니다. 일단 그래프가 완성된 다음에는 관계들은 즉시 명확하게 보입니다. 그러나 그 관계들이 그렇게 명확하게 보려면 그래프를 그리는 작업까지가 완료되어야만 합니다.

(중략)

다양한 방식으로 이야기했던 것처럼, 인간은 잡음이 많고, 좁은 대역폭의 장치이지만, 인간의 신경계는 동시에 여러가지 일을 처리할 수 있는 병렬성과 즉각적으로 반응하는 회로를 가지고 있습니다. 인간과 비교해 볼 때, 컴퓨터는 대단히 빠르고 정확하게는 하지만, 한번에 한 가지만을 처리하거나, 이미 주어진 몇 가지 기본 연산만을 할 수 있을 뿐입니다.

(중략)

만일 인간과 컴퓨터의 긍정적인 특성을 효과적으로 결합하면, 공생적 협력관계는 대단히 가치 있을 것입니다. 물론, 속도와 언어상의 차이는 극복해야 할 어려운 문제로 남아있지요.

1960년에 쓰여진 이 글은 어도비 센세이의 콘텐츠 지능, 디자인 지능과 관련이 많다. 이 글을 요즘의 머신러닝 맥락에서 보자면, 데이터 처리에 만만치 않게 데이터의 수집과 전처리 그리고 시각화에 많은 수고와 비용이 필요한 상황으로 이해할 수 있을 것이다.

물론 이 과정을 효율적으로 만들기 위해 고안된 컴퓨터는 오히려 더 많은 정보를 생산하고 소비하는 도구가 되었기 때문에 오늘 날에도 여전히 통하는 이야기다. 리클라이더의 다른 아이디어인 범 우주적인 컴퓨터 네트워크(intergalactic computer network)는 인터넷의 전신인 ARPANET의 근간이 되는 개념이다. 레이 커즈와일(Ray Kurzweil)이 이 글의 전문을 자신의 사이트에 옮겨둔 것⁵⁶이 있다.

X의 목적은 통찰이지 Y가 아니다

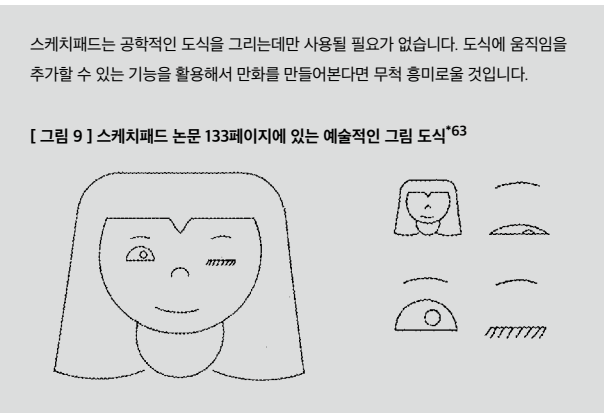
리처드 해밍(Richard Hamming)⁵⁷도 1962년에 "컴퓨팅의 목적은 통찰(insight)이지 숫자가 아니다."라고 말했는데, 요즘의 머신러닝 연구자들이 많이 사용하는 주피터 노트북⁵⁸의 전신이 되는 IPython 노트북을 만든 페르난도 페레즈(Fernando Perez)도 '문학적

컴퓨팅'에 관한 글⁵⁹에서 이 말을 인용했다. 앞으로 이런 인간의 통찰을 돕는 글쓰기(코드) 창작 도구들이 기본적으로 가지고 있을 요소에 머신러닝이 들어가는 것은 분명한 흐름이다.

어도비 센세이와 머신러닝 시대의 창작 도구를 다시 생각해 보자는 맥락에서 이런 1940-1960년대 아이디어들의 흔적을 살펴볼 수 있는 것은 사실 매우 당연하다. 어도비는 제록스 PARC 연구소에서 POSTSCRIPT⁶⁰를 연구하던 두 연구원이 창립했다. 어도비 일러스트레이터의 역사⁶¹를 보면 창립자 중 한 명인 존 워녹(John Warnock)에 관한 이야기가 나오는데 그의 아내인 마바 워녹(Marva Warnock)은 그래픽 디자이너였고 존 워녹이 프로그래밍 리터러시를 가지고 있어야만 사용할 수 있었던 POSTSCRIPT에 GUI를 붙여 기술의 문턱을 낮춘 일러스트레이터를 만들었던 배경에는 아내와 스승의 영향이 있었을 것이라 추측할 수 있다. 존 워녹의 스승은 아이반 서덜랜드(Ivan Sutherland)인데 그가 바로 1960년대 초에 '스케치 패드: 인간-기계 간의 그래픽 커뮤니케이션 시스템'⁶²만든 사람이다.



1963년에 발표된 그의 논문에는 다음과 같은 구절이 나온다.



아이반 서덜랜드의 스승은 정보이론의 선구자인 클로드 새넌(Claude Shannon)이다. 어도비 센세이의 시연에서 볼 수

있듯이 결국 창작 또한 정보를 다루는 작업의 맥락에서 이해할 수 있다. 이런 역사에 관해서는 '전쟁, 기술, 창작-인간과 컴퓨터의 공생에 관한 역사 속의 씨줄과 날줄'⁶⁴과 '창작자를 위한 머신러닝에 관한 안내, 교육에 관한 함의'⁶⁵로 조금 더 자세히 적어두었다.

구글의 시니어 과학자인 피터 노빅(Peter Norvig)이 뉴욕의 Lisp⁶⁶ 커뮤니티⁶⁷에서 발표한 '우리가 생각하는 대로'를 오마주한 '우리가 프로그램하는 대로'⁶⁸에서도 머신러닝에 의해 변화할 인간의 생각을 돕는 도구에 관한 내용들이 많이 있다. 2016년에 구글의 문서 도구 오른쪽 하단에는 탐색(explore) 버튼이 생겼는데, 피터 노빅은 2017년 4월 발표에서 자연어로 데이터를 다룰 수 있는 것에 관한 이야기를 했고, 머지 않은 6월에 구글은 머신러닝으로 구글 스프레드시트 안의 데이터를 바로 시각화 하기⁶⁹를 구글의 제품으로 발표하면서 스프레드시트의 데이터를 자연어로 다룰 수 있도록 했다. 탐색 버튼과 어도비 센세이의 버튼이 참 닮았다는 생각이 든다.

컴퓨터 프로그래밍의 여명기에 폰 노이만이 제자인 프린스턴 대학 학생들이 기계어로 바로 프로그래밍 언어를 작성하는 것이 아니라 어셈블러(assembler)⁷⁰를 만들고 있는 것을 보며 소중한 컴퓨터를 그런 사소한 일에 쓴다며 화를 냈다는 일화⁷¹는 유명하다. 그런데 시대는 바야흐로 인간이 기계 쪽으로 다가가서 소통하는 시대가 아니라 기계가 인간 쪽으로 다가와서 소통할 수 있는 시대로 넘어가고 있다. 피터 노빅은 발표에서 필요(need)와 바람(want)의 차이와, 정말로 필요하거나 원하는 것을 아는 것은 쉬운 일이 아니라고 이야기하며 아래의 AI 기본 공식을 소개했다. "취해야 할 최선의 행동은 현재 상태에서 행한 행동들 중 우리가 추구하고자 하는 가치의 기대를 최고로 충족하는 확률을 가진 행동을 찾아서 얻을 수 있다." 이 공식이야 어쨌든, 추구하고자 하는 것을 얻기 위해 구체적으로 어떻게(how) 코드를 구현할 지 고민하는데 에너지를 많이 썼던 시절에서, 무엇(what)을 추구할 지 잘 생각하는 것이 더 중요해지는 시절로 넘어가는 중인 것은 분명한 것 같다.

[그림 10] 피터 노빅의 AI 기본 공식
Fundamental Formula of AI
act* = argmax a in actions E(utility(Result(a, s)))
· State Estimation: s
· Model of World: Actions, Result
· Probabilistic Reasoning: E
· Search Algorithm: Argmax
· Our Values/Desires: Utility

어떻게 코드를 구현 할지에 관한 혁신은 계속해서 진행 중이다. d3.js⁷²를 만든 마이크 보스톡(Mike Bostock, @mbostock)도 더 좋은 코드를 작성하는 방법(A better way to write code)⁷³에서 "시각화의 목적은 통찰이지 그림이 아니다"를 인용하면서 발표를 시작했다. 이 말은 리처드 해밍의 말을 벤 슈나이더만(Ben

Shneiderman)이 각색한 것이다. 발표에서 마이크 보스톡은 브렛 빅터(Bret Victor, @worrydream)의 아이디어를 여러번 인용하면서 주피터 노트북이 아직 다루지 않는 개념을 가진, 관찰 가능한 노트북(observable notebooks)⁷⁴을 소개한다. 이 도구는 코드를 저장하는 동안 끊임없이 시각적인 피드백을 주어 코드를 저장하는 사람의 통찰을 돕는다. 이 도구는 구글의 PAIR(People+AI Research Initiative)⁷⁵에서 만든 웹에서 바로 뉴럴 네트워크를 다룰 수 있게 해주는 deeplearn.js의 상호작용 놀이터⁷⁶에 탑재될 예정이다. 뉴럴 네트워크는 이제 코드의 한 단락(cell)으로 들어갈 수 있고, 하나의 블럭처럼 사용할 수 있다. 최근 안드레이 카파시(Andrej Karpathy, @karpathy)가 소프트웨어 2.0이란 글⁷⁷을 공개했는데, 뉴럴 네트워크가 소프트웨어를 작성하는데 있어 공간이 되는 또 다른 스택이 될 것이라는 이야기를 했다. 무엇인가 딱 맞아 떨어지는 느낌이다.

익숙한 것을 새롭게 보기

1940-1960년대의 아이디어는 어떻게 보면 첨단 분야에서는 케케묵은 오래된 생각일 수 있다. 그러나 뉴럴 네트워크 또한 비슷할 정도로 오래된 생각인 점을 기억하면 이런 예전의 아이디어들을 다시 되새김질하며 새롭게 조명하는 가운데 얻을 수 있는 힌트들도 있지 않을까 싶다. 예전에는 실현하기 어려운 개념이었지만 지금이라면 가능한 것들, 더 확장해서 해낼 수 있는 것들이 있을지도 모른다. 그 시기에 중요한 아이디어들이 나온 이유 중 하나는 아직 컴퓨터가 어떤 것인지 무엇을 할 수 있는지가 명확하게 정의되어 있지 않았기 때문에 선입견 없이 모든 가능성을 열어놓고 다양한 방향에서 탐구하고 실험했기 때문이다.

신경정보처리시스템학회(neural information processing systems, NIPS) 같은 학회에서 선구자들의 오래된 아이디어를 언급하며 지적인 크레딧의 영토 분쟁 시비⁷⁸를 거는 연구자로 LSTM(long short-term memory)을 고안한 유르겐 슈미트후버(Jurgen Schmidhuber)를 들 수 있다. NIPS 2015에서 안 르쿤과 요수아 벤지오(Yoshua Bengio)의 튜토리얼에 질문을 가장하여 저격을 하는 모습⁷⁹이나 NIPS 2016에서 이안 굿펠로우(Ian Goodfellow)의 GAN 튜토리얼에서 또 비슷한 시도⁸⁰를 하는 것을 보고 소셜네트워크 서비스 타임라인에선 슈미트후버드(Schmidhubered)라는 표현을 써가며 굵지 않은 시선을 보냈다. 그의 태도에는 분명 아쉬움이 있다고 생각하지만 요즘 혁신의 공간이 되었던 역사에 관해서 짚어보는 것은 의미있는 일이라고 생각한다. 늘 앞으로만 달려 나가는 것이 아니라 현재의 방향(벡터)이 어디에서 시작된 것인지를 살펴볼 때 배울 수 있는

가치도 있다. 슈미트후버에 관해서 또 한 가지 재미있는 일이 있다. 그는 이번 NIPS 2017의 창의성과 디자인을 위한 머신러닝(machine learning for creativity and design)^{*81}워크숍의 키노트를 맡았는데 다른 발표들과 출판된 작업들도 궁금하지만 슈미트후버가 어떤 이야기를 할지도 무척 궁금하다. 브렛 빅터가 '프로그래밍의 미래' 발표^{*82}에서 앨런 케이(Alan Kay)에게 받은 탈학습(unlearn)에 관한 조언을 회고한 글이 기억이 난다.

지식에 관한 트릭 중에 하나는 "지식을 획득한 후 그 향기를 빼고는 다 잊어버리는 것"입니다. 왜냐하면 지식은 소용갈기도 해서 자기 "두뇌의 목소리"를 찾아들게 만들기 때 문입니다. 향기 부분이 중요한 것은 향기는 그 지식을 다시 찾을 수 있도록 돕기 때문입 니다. 내적인 열망이 선택한 도착지를 찾아 갈 수 있도록 돕기 때문입니다.

아카이브(arXiv)^{*83}에 읽어야만 할 것 같은 머신러닝 관련 논문이 범람하는 요즘 지식(정보)에 휘둘리지 않고 가고자 하는 길을 잘 항해하는 것은 무척이나 중요하게 느껴진다. 사이버네틱스의 어원에 방향을 조종하는 '키잡이'를 의미하는 퀴베르네테스(kyvernitis)가 있는데, 노버트 위너 또한 1954년에 출판한 '인간의 인간적 활용, 사이버네틱스와 사회'에서 노하우(know-how)보다는 노왓(know-what)을 중요하게 여겼다. 어떻게 할 것인가 보다는 추구하고자 하는 것이 무엇인지 아는 것, 또는 어디로 갈 것인가 그 방향을 설정하는 것의 중요성에 관해 생각해 보게된다.

현대의 인간, 특히 현대의 미국인들은 많은 '노하우'를 갖고 있을지 몰라도 '노왓'을 갖고 있는 경우는 거의 없다. 그들은 기계를 움직이는 동구나 원치 등은 제대로 살펴보지도 않은 채, 기계가 정해 주는 결정의 뛰어난 기술력만 받아들일 것이다. 그렇게 함으로써 조만간 자기 스스로를 제이콥스의 '원숭이 발'에 나오는 아버지의 처지로 만들 것이다. (중략) 아니면 '천일야화'에 나오는 아랍의 어부처럼 성난 자니가 들어 있는 병뚜껑의 솔로몬 봉인을 열었을 때와 같은 일이 벌어질 수도 있다.

안드레이 카파시의 소프트웨어 2.0을 바탕으로 생각해 볼 때 뉴럴 네트워크가 창작자들도 활용할 기본적인 스택이 될 것은 자명하다고 생각한다. 자바를 기반으로 하는 DSL(domain-specific language)^{*84}인 프로세싱이 자바보다 훨씬 간결한 문법을 가지며 예술가나 디자이너가 프로그래밍을 활용해 창작을 시작하는 경험에 도움이 되는 것 처럼, 텐서플로우(Tensorflow)보다 파이토치(PyTorch)가 더 쉽게 접근할 수 있는 것처럼, 뉴럴 네트워크의 모델을 만드는 것이 코드가 아니라 스크래치(Scratch)같은 블록 기반의 시각적인 언어를 쓰는 VPL(visual programming language)^{*85}로 접근하는 것이 가능해진 것처럼, 관찰 가능한 노트북으로 코드의 상태를 암중모색 하지 않아도 되는 것처럼, 어도비 센세이 같이 다른 층위의 추상화 수준이 높은 도구가 나타나는 것처럼 어떻게 그것을 할 것인지(know-

how)에 관한 문턱은 계속 낮아질 것이다. 세계 최고의 두뇌들이 기술의 문턱을 낮추고 시장을 교란하는 일에 자본을 투자하며 앞장서고 있다(기술의 문턱을 낮출수록 기반이 되는 낮은 수준의 기술이 가지는 권력은 높아진다). 다만 무엇(know-what)을 할 것인지에 관한 가치는 그 가치를 교란하고자 하는 다양한 시도들에도 덜 흔들리지 않을까 예상해 본다.

어떻게(know-how) 할 지에만 천착하면 니콜라스 카(Nicholas Carr)가 유리감옥^{*86}에서 언급한 것 처럼 자동화와 기계에 의해 도움을 받는 창작이 오히려 우리 생각의 감옥으로 작동할 가능성도 있다. 뭔가 이전에 없던 창작을 하려고 했는데 기계가 학습하는 데이터가 핀터레스트(Pinterest)^{*87}와 비헨스(Behance)^{*88} 등에서 학습할 수 있는 유행을 따르는 편향된 스타일 뿐이라면 어떻게 되겠는가? 그러므로 다양한 창작의 배경이 되는 생각과 역사 속의 도전과 고민의 기록들에서 향기를 취하고 세부사항은 잊어버린 후 다시 향기를 따라 우리의 신경망을 재연결(rewiring)하는 시도가 창작자들의 방향을 설정하는데 도움이 될 거란 가설을 세워보게 된다.

마지막으로 이 글을 쓰는데 도움이 됐지만, 각주에는 달지 못했던 책들을 적어둔다.

· 멀티미디어 - 바그너에서 가상현실까지, 랜덜 패커, 켄 조던 지음

· 소프트웨어가 명령한다, 레프 마노비치 지음

· CODE, 찰스 펫졸드 지음

· 드림잉 인 코드, 스콧 로젠버그 지음

· 그래픽 디자인의 역사, 필립 B 맥스 지음

· Moving Innovation, Tom Sito 지음

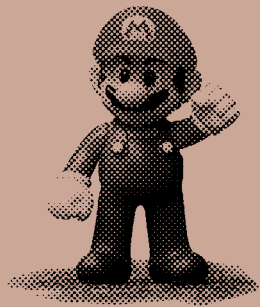
각 권이 일부 지면을 할애해서 이 글에서 소개한 역사들을 다루고 있다. 그런데 이 글에서 예찬하다시피 한 혁신적인 아이디어가 나왔던 1960년대가 '흑인 여성 컴퓨터'(※ 필자 주 : 이 시기의 컴퓨터는 사람이 맡았던 계산하는 '직업'이었고, 이 당시의 NASA에도 컴퓨터로 고용된 여성들이 있었다)의 이야기를 다룬 영화 히든 피겨즈(Hidden Figures)^{*89}의 배경이 되는 시대라는 지점에서 온도 차이가 느껴지기도 한다. 이 시기의 백인 남성 위주의 선구자들 중 훗날이라도 페미니즘 등의 다양성을 강조^{*90}하며 인간의 능력에 관한 가부장제적인 선입견을 중화하려는 시도를 한 사람은 내가 아는 한 시모어 페퍼트(Seymour Papert)^{*91}밖에 없다. 그는 마빈 민스키(Marvin Minsky)와 함께 퍼셉트론즈(Perceptrons)^{*92}를 출판해서 Si의 겨울^{*93}이 오는데 일부 기여를 하기도 했으며, 얀 르콘도 자신의 페이스북 계정을 통해 학창시절 시모어 페퍼트에게 받았던 영향 때문에 후에 뉴럴 네트워크를 공부하는 계기가 되었다^{*94}는 이야기를 했었다.

한편, 2016년 스탠포드의 페이페이 리(Fei-Fei-Li) 교수가

진행했던 'CNN에 대한 소개 및 역사적 맥락' 강연^{*95}에서는 시모어 페퍼트가 1966년에 여름 비전 프로젝트(The summer vision project)^{*96}를 제안한 인물로만 등장한다. 시모어 페퍼트가 진정으로 하고 싶었던 것^{*97}에 관해 알고 있던 모르건 간에, 페이페이 리는 현재 AI4ALL^{*98} 등을 통해서 AI 기술의 민주화^{*99}에 관한 노력을 하고 있는 중이다. 시모어 페퍼트가 요즘 소프트웨어 교육에서 화두가 되기도 하는 계산적 사고(computational thinking)라는 용어를 고안하며, 이를 중요하게 생각한 이유의 근원에는 전 세계의 어린이들이 비판적인 사고를 할 수 있는 시민으로서 자립하길 바라는 액티비스트(activist)적 관점이 자리잡고 있다.

반면 우리의 소프트웨어 교육은 아직 비판적 사고를 겸양할 수 있도록 돕는 교육보다는 알고리즘 만들기 교육 쪽에 무게를 두고 있다. 이제는 제법 익숙해진 계산적 사고라는 개념에 대해 새롭게 볼 수 있으면 좋겠다는 바람을 가지는 한편, 액티비스트였던 애런 슈워츠(Aaron Swartz)의 죽음을 떠올리며 고민하게 된다. 비판적인 생각을 하며 행동하는 젊은이가 살아가기에 우리 사회는 과연 어떨까. 물론 이는 현대 예술 또한 액티비즘(Activism)과 많은 관련이 있다. 예술이 다루는 심미의 지평이 다채롭게 넓어진 지는 이미 오래고 뉴럴 네트워크가 이를 더욱 다채롭고, 넓고, 깊게 할 것임에 틀림없다.

^{*1} 참고 | <https://www.facebook.com/yann.lecun/posts/10154498539442143> ^{*2} 참고 | <https://www.facebook.com/seungjoon.choi/posts/10212300525203813> ^{*3} 참고 | Godel, Escher, Bach: an eternal golden braid, https://en.wikipedia.org/wiki/Godel,_Escher,_Bach, 1979년 ^{*4} 참고 | 편견에 관해서는 구글의 기계학습과 인간의 편견(Machine Learning and Human Bias) <https://www.youtube.com/watch?v=59bMh59JQDo> 영상과 캐시 오닐의 '대량 살상 수학무기: 어떻게 빅데이터는 불평등을 확산하고 민주주의를 위협하는가' 등의 콘텐츠를 추천한다 ^{*5} 참고 | <https://en.wikipedia.org/wiki/Manifold> ^{*6} 참고 | 다른 방식으로 보기 (Ways of seeing), John Berger, https://en.wikipedia.org/wiki/Ways_of_Seeing ^{*7} 참고 | Ben F. Laposky https://en.wikipedia.org/wiki/Ben_F._Laposky 벤 라포스키에 관해서는 '기계가 예술을 만들었을 때'(When the Machine Made Art)란 책에 잘 나와있다. ^{*8} 참고 | 만프레드 모어에 관해서는 '앨리스 온'이 2013년에 인터뷰한 내용(<http://aliceon.tistory.com/2227>)이 좋고, 그의 작업을 비롯한 코드를 활용한 초기의 창작 작업은 ReCode 프로젝트(<http://recodeproject.com>)에서 보다 현대적인 프로그래밍 언어로 복원한 것을 살펴볼 수 있다. ^{*9} 참고 | 자바 기반의 프로그래밍 언어로, 예술가들이 작품을 만들 때 사용하곤 한다.(<https://processing.org>) ^{*10} 참고 | History of the Future, Art & Technology from 1965 - Yesterday | Casey Reas | The Gray Area Festival <https://www.youtube.com/watch?v=mHox98NFU3o> ^{*11} 참고 | <https://www.aec.at/ai>, 이번 아로스 일렉트로니카에 출판된 더 많은 AI 관련 작업들은 <https://www.aec.at/ai/en/media-art-between-natural-and-artificial-intelligence> 에서 살펴볼 수 있다. ^{*12} 참고 | <http://www.miketyka.com>, 마이코 타이카는 자신의 고향상도 GAN에 관한 작업 방식을 Superresolution with semantic guide (<https://mtyka.github.io/machine/learning/2017/08/09/highes-gan-faces-followup.html>)라는 글로 정리해 두었다. ^{*13} 참고 | <https://www.aec.at/ai/en/portraits-of-imaginary-people> ^{*14} 참고 | <https://burningman.org> ^{*15} 참고 | <https://research.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html> ^{*16} 참고 | <https://distill.pub/2017/feature-visualization> ^{*17} 참고 | <https://www.wired.com/2015/12/inside-deep-dreams-how-google-made-its-computers-go-crazy> ^{*18} 참고 | <http://quasimondo.com> ^{*19} 참고 | <https://www.aec.at/ai/en/x-degrees-of-separation>, 케빈 베이컨의 6단계 https://en.wikipedia.org/wiki/Six_Degrees_of_Kevin_Bacon 에서 차용 ^{*20} 참고 | <https://experiments.withgoogle.com/arts-culture/x-degrees-of-separation> ^{*21} 참고 | <https://www.wired.com/story/neurographer-puts-the-art-in-artificial-intelligence> ^{*22} 참고 | <https://www.aec.at/ai/en/image-transformations> ^{*23} 참고 | Gene Kogan at Eyeo 2017, Starving Computer Scientist, <https://vimeo.com/232544884> ^{*24} 참고 | <https://www.nytimes.com/2017/08/14/arts/design/google-how-ai-creates-new-music-and-new-artists-project-magenta.html> ^{*25} 참고 | <http://otoro.net> ^{*26} 참고 | <https://magenta.tensorflow.org/sketch-rnn-demo> ^{*27} 참고 | <https://ami.withgoogle.com> ^{*28} 참고 | https://www.ted.com/talks/blaise_aguera_y_arcas_how_computers_are_learning_to_be_creative?language=ko ^{*29} 참고 | <https://www.youtube.com/watch?v=pFwW-BJQhpc> ^{*30} 참고 | <https://medium.com/artists-and-machine-intelligence/what-is-ami-ccd936394a83> ^{*31} 참고 | https://en.wikipedia.org/wiki/The_Work_of_Art_in_the_Age_of_Mechanical_Reproduction ^{*32} 참고 | https://en.wikipedia.org/wiki/Vil%C3%A9m_Flusser ^{*33} 참고 | https://en.wikipedia.org/wiki/David_Hockney ^{*34} 참고 | Understanding media: The extensions of man, https://en.wikipedia.org/wiki/Understanding_Media ^{*35} 참고 | <https://ko.wikipedia.org/wiki/사이버네틱스> ^{*36} 참고 | The human use of human beings, https://en.wikipedia.org/wiki/The_Human_Use_of_Human_Beings, <https://archive.org/details/NorbertWienerHumanUseOfHumanBeings> 에 전문이 공개되어 있다. ^{*37} 참고 | https://en.wikipedia.org/wiki/Seadragon_Software ^{*38} 참고 | AlphaGo Zero: Learning from scratch, <https://deepmind.com/blog/alphago-zero-learning-scratch> ^{*39} 참고 | Accelerating Your Creativity - Adobe MAX 2017 - Day 1 Keynote 2시간 6분 부터 https://www.youtube.com/watch?v=4j_pdlawSac&t=7599 ^{*40} 참고 | <http://www.adobe.com/sensei.html> ^{*41} 참고 | <https://www.facebook.com/seungjoon.choi/posts/10213452482162017> ^{*42} 참고 | Rethinking Design Tools in the Age of Machine Learning, <https://medium.com/artists-and-machine-intelligence/rethinking-design-tools-in-the-age-of-machine-learning-369f3f07ab6c> ^{*43} 참고 | <https://twitter.com/patrickhebron/status/920708009331974144> ^{*44} 참고 | <http://www.creativeai.net> ^{*45} 참고 | Assisted Writing, <https://medium.com/@samim/assisted-writing-7adea9aed19> ^{*46} 참고 | https://en.wikipedia.org/wiki/The_Mother_of_All_Demos ^{*47} 참고 | [https://en.wikipedia.org/wiki/NLS_\(computer_system\)](https://en.wikipedia.org/wiki/NLS_(computer_system)) ^{*48} 참고 | AUGMENTING HUMAN INTELLECT, <https://www.dougenelbart.org/pubs/augment-3906.html> ^{*49} 참고 | https://en.wikipedia.org/wiki/Intelligence_amplification ^{*50} 참고 | MIT에서 만든 바니바 부시에 관한 짧은 영상, <https://www.youtube.com/watch?v=flsPFW5Yf3c> ^{*51} 참고 | <https://www.theatlantic.com/magazine/archive/1945/07/as-we-may-think/303881> ^{*52} 참고 | 아트센터나비 학예팀의 번역 ^{*53} 참고 | <https://en.wikipedia.org/wiki/Memex> ^{*54} 참고 | https://en.wikipedia.org/wiki/History_of_hypertext ^{*55} 참고 | <https://groups.csail.mit.edu/medg/people/ps2/Licklider.html> ^{*56} 참고 | <http://www.kurzweilai.net/memorandum-for-members-and-affiliates-of-the-intergalactic-computer-network> ^{*57} 참고 | https://en.wikipedia.org/wiki/Richard_Hamming ^{*58} 참고 | <http://jupyter.org> ^{*59} 참고 | "Literate computing" and computational reproducibility: IPython in the age of data-driven journalism, <http://blog.fperex.org/2013/04/literate-computing-and-computational.html> ^{*60} 참고 | <https://en.wikipedia.org/wiki/PostScript> ^{*61} 참고 | The Story Behind Adobe Illustrator, https://www.youtube.com/watch?v=lgaCKT_Ncdk ^{*62} 참고 | Sketchpad: A man-machine graphical communication system, <https://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-574.pdf> ^{*63} 참고 | <http://worrydream.com/refs/Sutherland-Sketchpad.pdf> ^{*64} 참고 | <https://goo.gl/cpNU11> ^{*65} 참고 | <https://goo.gl/630MPL> ^{*66} 참고 | [https://en.wikipedia.org/wiki/Lisp_\(programming_language\)](https://en.wikipedia.org/wiki/Lisp_(programming_language)) ^{*67} 참고 | <http://lispnyc.org> ^{*68} 참고 | Peter Norvig on "As We May Program", <https://vimeo.com/215418110> ^{*69} 참고 | Visualize data instantly with machine learning in Google Sheets, <https://www.blog.google/products/g-suite/visualize-data-instantly-machine-learning-google-sheets> ^{*70} 참고 | https://en.wikipedia.org/wiki/Assembly_language ^{*71} 참고 | John von Neumann's reaction to assembly language and Fortran, <http://www.columbia.edu/cu/computinghistory/index.html> ^{*72} 참고 | <https://d3js.org> ^{*73} 참고 | A Better Way to Write Code, <https://www.youtube.com/watch?v=aT4JvF7sglg> ^{*74} 참고 | <https://observablehq.com> ^{*75} 참고 | <https://ai.google/pair> ^{*76} 참고 | deeplearn.js interactive playground, <https://deeplearnjs.org/demos/playground/examples.html> ^{*77} 참고 | Software 2.0, <https://medium.com/@karpathy/software-2-0-a64152b37c35> ^{*78} 참고 | Critique of Paper by "Deep Learning Conspiracy", <https://plus.google.com/10084985654000067209/posts/9BDtGwCDL7D> ^{*79} 참고 | <https://www.youtube.com/watch?v=74vUX2szsms&t=7168> ^{*80} 참고 | <https://channel9.msdn.com/Events/Neural-Information-Processing-Systems-Conference/Neural-Information-Processing-Systems-Conference-NIPS-2016/Generative-Adversarial-Networks%20the%2014년%203분%20조%20부터> ^{*81} 참고 | Machine Learning for Creativity and Design, <https://nips2017creativity.github.io> ^{*82} 참고 | The Future of Programming, <http://worrydream.com/dbx> ^{*83} 참고 | <https://arxiv.org> ^{*84} 참고 | https://en.wikipedia.org/wiki/Domain-specific_language ^{*85} 참고 | https://en.wikipedia.org/wiki/Visual_programming_language, 뉴럴 네트워크 콘솔 https://dl.sony.com/go/모델_빌더 <https://deeplearnjs.org/demos/model-builder> 등의 예시가 있다. ^{*86} 참고 | 유리 감옥 - 생각을 통제하는 거대한 힘, 니콜라스 카 지음, 한국경제신문 출판 ^{*87} 참고 | <https://www.pinterest.com> ^{*88} 참고 | <https://www.behance.net> ^{*89} 참고 | https://en.wikipedia.org/wiki/Hidden_Figures ^{*90} 참고 | Epistemological Pluralism and the Revaluation of the Concrete, <http://www.papert.org/articles/EpistemologicalPluralism.html> ^{*91} 참고 | 생각을 생각하는 법을 생각하게 한 거인, <http://www.hankookilbo.com/v/f81d49a25f2545589e80d7a960d22378> ^{*92} 참고 | [https://en.wikipedia.org/wiki/Perceptrons_\(book\)](https://en.wikipedia.org/wiki/Perceptrons_(book)) ^{*93} 참고 | https://en.wikipedia.org/wiki/Al_winter ^{*94} 참고 | <https://www.facebook.com/yann.lecun/posts/10153713733897143> ^{*95} 참고 | <https://www.youtube.com/watch?v=NfnWJUyUYU&t=1837> ^{*96} 참고 | <https://dspace.mit.edu/handle/1721.1/6125> ^{*97} 참고 | Thinking about Thinking about Seymour, <https://www.media.mit.edu/events/papert> ^{*98} 참고 | <http://ai-4-all.org> ^{*99} 참고 | 페이페이 리의 모두를 위한 AI, <https://brunch.co.kr/@kakao-it/70>



슈퍼마리오

그리고 GAN :

두 번째 이야기

exercise

송호연 | 강화학습으로 풀어보는 슈퍼마리오 part2.

56

유재준 | Do you know GAN? (2/2)

64

강화학습을 슈퍼마리오 게임에 적용한 송호연 님의 글과 GAN의 개념을 자세하게 설명 해주시는 유재준 님의 글 모두 두 번째 편이 이번 호에 게재됩니다. 송호연 님은 딥마인드의 DQN 알고리즘이 학습을 하는 과정을 설명해 주셨습니다. 유재준 님은 GAN과 기존 모델을 비교하여, 어떠한 장점과 단점이 있는지를 서술해 주셨습니다. 책으로 배운 지식을 실전에 쓰고 싶으신 분들에게 두 분의 글을 추천해 드립니다.

강화학습으로 풀어보는 슈퍼마리오 part 2.*

오늘은 네이처(Nature)지의 표지를 장식한 딥마인드(Deepmind)의 Deep Q-Network 알고리즘과 텐서플로우(Tensorflow)로 구현된 OpenAI baselines 코드¹를 보면서 딥마인드의 DQN(Deep Q-Network) 알고리즘이 학습하는 과정을 순서대로 하나 하나 리뷰해보려 한다.

사람마다 공부하는 성향이 다르다. 기본기부터 하나씩 쌓으며 조심스럽게 응용기술을 공부하는 사람이 있고, 응용기술 구현으로 시작해 기본기를 채워 나가는 스타일이 있다. 필자는 후자다. 동작하는 프로그램을 만들고 개선을 위해 해당 구성 기술 요소에 대해 하나씩 이해해 나가기 시작한다. 필자의 첫 번째 튜토리얼을 따라했다면(카카오 AI 리포트 7호 참고)², 응용 기술부터 시작한 것이다. 그렇다면, 우리는 일단 동작하는 강화학습 프로그램을 돌려본 적이 있는 셈이다.

자, 이제 핵심 기술을 이해를 할 시간이다. 각 구성 요소와 프로그램이 동작하는 원리를 이해하고, 무엇을 개선해야 더 나은 학습이 되는지 이해해 보도록 하자.

이번 튜토리얼의 목적은 간결하지만 도전적이다. 바로 DQN(Deep Q-Network) 알고리즘을 공식적으로 세상에 알리고 네이처지의 표지를 장식한 딥마인드의 'Playing Atari game with deep reinforcement learning' 논문을 이해하는 것이 첫 번째 목표이다. 두 번째 목표는 논문에 적힌 알고리즘이 실제로 구현된 OpenAI의 baselines 오픈소스에 있는 DQN 코드를 읽고 하나씩 이해하는 것이다. 전설적인 과학자 리처드 파인만(Richard Feynman)이 말한 명언과 함께 공부를 시작해보자. 논문을 읽은 것 만으로는 이해했다고 할 수 없다. 논문의 내용과 구현체를 같이 살펴 보면서 실제로 어떻게 DQN(Deep Q-Network)이 작동하게 되는지 이해해 보는 것이다.

"내가 만들지 못하는 것은,
내가 아직 이해하지 못한 것이다."

- 리처드 파인만(Richard Feynman)

Deep Q-Network 알고리즘의 특징

DQN은 총 세가지 특징으로 설명할 수 있다. 첫 번째 특징은 타겟 Q함수다. 두 번째 특징은 게임 플레이 기록을 저장하는 리플레이 메모리(replay memory, replay buffer)다. 그리고 세번째 특징은 가장 중요한 두뇌의 역할을 하는 Q함수(Q function)를 인공신경망(artificial neural network, ANN)으로 만들었다는 점이다. 그리고 이 알고리즘을 실제로 동작하게 만드는 시뮬레이션 환경이 필요하다.

1) 강화학습을 위한 시뮬레이션 환경

딥마인드의 논문에서 사용되었던 환경은 아타리(atari) 게임들이다. 우리가 사용할 게임은 바로 슈퍼마리오다. 우리가 강화학습에서 다루어보는 슈퍼마리오 학습환경은 OpenAI에서 개발한 gym과 게임 에뮬레이터인 fceux를 활용한다.

OpenAI의 gym은 생각보다 아주 심플하고 단순한 함수들로 이루어져 있다. 가장 중요한 3개의 함수를 소개하겠다.

(1) reset() 함수

reset() 함수는 게임 환경을 초기화하는 일을 한다. 주로 맨 처음 게임을 시작할 때, 그리고 에이전트가 죽어서 게임이 끝났을 때 게임을 초기화하는 역할을 담당한다. reset() 함수는 게임 환경을 초기화하면서 observation 변수를 같이 반환한다.

(2) step() 함수

step() 함수는 에이전트에게 명령을 내리는 함수이다. 사실상, 가장 많이 호출하게 되는 함수이다. 이 함수로 action 명령을 보내고,

환경에서 observation 변수, 보상(reward), 게임 종료 여부 등 변수를 반환한다.

(3) render() 함수

render() 함수는 화면에 해당 상태를 표시하는 역할을 한다. Gym 환경에서는 render() 함수로 현재 게임 상태를 화면에 출력할 수 있다. 조금 헛갈릴 수 있는데, 슈퍼마리오에서는 이 함수를 호출하지 않더라도 화면이 표시된다. 그러니 슈퍼마리오 환경에서는 이 함수를 신경쓰지 않아도 된다.

2) 타겟 Q함수(Q-function)

타겟 Q함수는 학습의 대상이 되는 Q함수가 학습이 되면서 계속 바뀌는 문제를 해결하기 위해 딥마인드에서 제시한 해법이다. 타겟 Q함수가 추가로 생기면서, DQN의 Q 함수는 총 두 개가 된다. 이 기법은 우선 학습을 시키는 기준으로 타겟 Q함수를 사용하며, 실제 그라디언트 디센트(gradient descent)로 Q 함수를 학습시킨다. 일정 스텝이 될 때마다 Q함수의 가중치(weights)들을 타겟 Q함수에 업데이트한다.

3) 리플레이 메모리(Replay Memory)

리플레이 메모리는 강화학습에서 학습의 재료가 되는 중요한 샘플(sample)을 저장해두는 저장소다. 딥마인드에서 제안한 DQN 논문 'Playing Atari with deep reinforcement learning'에서는 샘플들 간의 상관관계(correlation)으로 인해 학습 속도가 느려지는 문제를 해결하기 위하여 Long-Ji Lin이 'Reinforcement learning for robots using neural networks'³에서 제안한 experienced replay를 이용한 학습법을 사용했다. 즉 게임을 플레이하면서 모든 과정이 들어오는 즉시 훈련시키지 않고, 일단 메모리에 저장해뒀다가 나중에 일정 수의 샘플을 랜덤으로 꺼내서 학습을 시키는 방식을 사용하였다. 플레이 기록을 저장해두고 미니배치(minibatch)를 활용해 학습시킨다.

우리는 gym agent에서 매번 명령을 내리면서, state(observation), reward, done 세 가지 정보를 받아올 것이다. 달리 표현하면, 한 번 명령을 내릴 때 SARS(S : state, A : action, R : reward, S' : next state) 이 네 가지 정보를 확보할 수 있다는 것이다.

4) CNN과 MLP을 활용한 Q함수

딥마인드 논문에서 딥러닝 기술이 발전하면서 딥러닝과 강화학습을 접목시키는 아이디어가 조명받고 있다고 했다. DQN은 인공신경망으로 정책 함수(policy function)를 근사(approximate)했다. 인공신경망의 아키텍처는 튜토리얼 1편(카카오 AI 리포트 7호 참고)⁴에서 설명했듯이

글 | 송호연 chris.song@kakaocorp.com

강화학습과 씬을 타다가 최근에 연애를 시작했습니다. 자기 전에도, 아침에 일어났을 때도 강화학습 생각이 납니다. 앞으로 세상을 파괴적으로 혁신시킬 범용인공지능(Artificial General Intelligence)에 완전히 빠져 있습니다. 카카오에서는 대규모 데이터 유저 프로파일링, 머신러닝 기술을 활용해 현실의 문제를 해결하는 데이터 엔지니어로 일하고 있습니다.

*감사 인사

이종원(RLCode 리더), 유승일(Google Senior Software Engineer), 민규식(한양대학교 박사 과정) 본문 튜토리얼의 내용에 대한 감사와 조언에 깊은 감사의 말씀을 전하고 싶습니다.

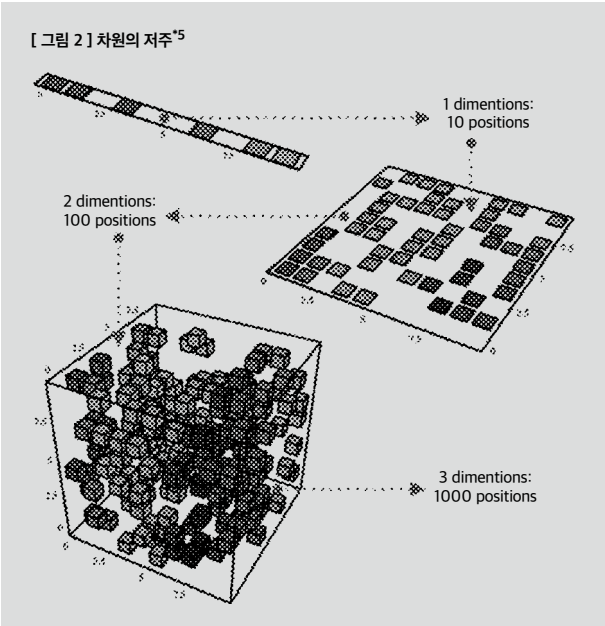
CNN(convolutional neural network)과 MLP(multi layer perceptron)로 이루어져 있다.



조금 더 깊은 이해를 위해, 우리가 만들 DQN의 Q 함수 특징을 한번 설명해 보겠다.

(1) 모델 프리(model-free)

근사함수를 활용한 방식은 에이전트가 행동한 샘플을 가지고 직접적으로 정책을 근사(approximate)시킨다. 이렇게 함으로써, 우리는 다이나믹 프로그래밍(dynamic programming)에서 겪었던 차원의 저주에서 벗어날 수 있게 되었다. 강화학습을 차원의 저주에서 벗어나게 해준 대표적인 기법은 인공신경망(artificial neural network, ANN)을 활용한 근사(approximation)다. 최근 딥러닝 시대에서는 대부분 인공신경망을 활용해 정책 네트워크를 훈련시키고 있다.



Tip. Model-based 강화학습과 Model-free 강화학습이란?
딥마인드의 알파고(AlphaGo) 팀 리더이자 강화학습의 전설인 데이비드 실버(David Silver)의 정의로 model-based 강화학습과 model-free 강화학습을 설명해 보겠다.⁶

Model-free 강화학습:

- 단어 그대로, 모델(model)이 없다.
- 샘플로부터 직접적으로 가치 함수(value function) 혹은 정책 함수(policy function)

를 훈련시킨다.

Model-based 강화학습:

- 샘플로부터 모델을 학습시킨다.
- 모델로부터 가치 함수(value function) 혹은 정책 함수(policy function)를 계획(plan)한다.
- 장점 : 지도 학습(supervised learning)을 통해 효과적으로 모델을 훈련시킬 수 있다. 모델의 불확실성에 대해 논할 수 있다.
- 단점 : 먼저 모델을 학습시킨 후, 모델로부터 가치 함수(value function)를 만들어낸 다. → 추정의 에러(approximation error)가 모델과 가치 함수 두 영역에서 동시에 생긴다.

(2) 오프폴리시(Off-policy)

DQN은 오프폴리시 방식을 사용했다. 오프폴리시는 단순히 말하면, 다른 에이전트가 하는 행동을 보고 배운다는 뜻이다. DQN 알고리즘은 자신이 플레이했던 기록을 리플레이 메모리에 넣어두고 랜덤으로 샘플링하여 학습을 진행한다. 이렇게 리플레이 메모리를 쓰는 경우, DQN 에이전트는 과거의 정책이 만들어냈던 샘플들로부터 배우게 된다. 즉, DQN 자신의 정책과 자신이 학습을 하는 기준이 되는 샘플을 만들어낸 정책이 다르게 된다.

Tip. 온폴리시(on-policy)와 오프폴리시(off-policy)는 어떤 차이일까?
딥마인드의 데이비드 실버의 표현에 따르면, 온폴리시와 오프폴리시는 이렇게 표현할 수 있다.⁷

온폴리시(on-policy):

- 자신이 하는 행동으로부터 배운다
- 자신의 정책 π 로부터 나온 sample들로 자신의 정책 π 를 학습시킨다.

오프폴리시(off-policy):

- 다른 에이전트가 하는 행동을 어깨 넘어 보고 배운다
- 행동 정책(behavior policy) b 로부터 나온 sample들로 자신의 타겟 정책(target policy) π 를 학습시킨다.

조금 더 구체적으로 설명하기 위해 서튼(RS Sutton) 교수님의 'Reinforcement learning : an introduction' 책에 있는 설명으로 온폴리시와 오프폴리시에 대해 부연 설명을 해보겠다⁸.

강화학습에서 정책을 정하는 함수를 업데이트 해나가는 방법에는 크게 두 가지 방법이 있다. 하나는 온폴리시 방식, 다른 하나는 오프폴리시 방식이다. 강화학습에서 정책 함수를 업데이트시킬 때 딜레마가 하나 존재한다. 그것은 바로, 탐험(exploration)을 해야 하기 때문에 에이전트가 최적의 경로로 움직이지 않는다는 점이다. 탐험을 하려면 새로운 시도를 해야 하는데, 이는 최적화되지 않은 행동(non-optimal)이다. 이 관점에서 온폴리시와 오프폴리시 방식을 각각 설명해 보겠다.

온폴리시 방식은 모델이 오프폴리시에 비해 단순히 구현하기 쉬운 반면 사실 약간의 타협을 한 방식이다. 온폴리시 방식은 최적의 정책(optimal policy)을 보고 배운다기 보다는 최적이 가까운 정책(near-optimal policy)을 보고 배운다. 온폴리시 방식은 단 하나의 정책으로 행동을 하는 동시에 같은 정책을 학습시킨다. 즉, 여전히 탐험을 하고 있는 정책으로부터 배운다.

오프폴리시 방식은 이 문제를 해결하기 위해 직접적인 해결책을 제시한다. 바로 타겟 정책과 행동 정책을 나누는 것이다. 타겟 정책은 우리가 강화학습 에이전트에게 가르치기 위한 기준이 되는 정책이다. 그리고 행동 정책은 탐험을 하며 새로운 행동을 만들어내는 정책이다. 이 경우에 우리는 에이전트를 타겟 정책으로부터 떨어진(off) 데이터로

부터 학습을 시키며, 이 전체적인 프로세스를 오프폴리시 학습(off-policy learning)이라고 한다. 일반적으로 오프폴리시 방식이 좀 더 강력하고 범용적인 성능을 보인다. 다만, 샘플링을 해야 하며 두가지 폴리시를 다뤄야 하기에 구현하기는 좀 더 어렵다.

(3) 미니배치(Minibatch)

미니배치란 많은 데이터 중 임의로 샘플을 뽑아 학습시키는 것을 의미한다. DQN은 리플레이 메모리와 미니배치 기법을 활용하여 연속적인 샘플들(samples) 간의 강한 상관관계를 제거하고자 하였다.

(4) 가치 기반(Value-Based) 강화학습

DQN은 대표적인 가치 기반(value-based) 강화학습으로서 가치 함수(value function)를 먼저 근사(approximate)시켜서 정책을 만들어내는 방식으로 학습을 시킨다.

(5) 감소하는 입실론 그리디(Decaying Epsilon-Greedy)
탐험과 활용(exploration & exploitation)은 강화학습이라는 시스템에서 해결해야 하는 아주 중요한 문제 중 하나다. DQN 강화학습 에이전트는 처음에는 랜덤으로 움직인다. 그러다가 어느 정도 학습을 하면 Q 함수 혹은 정책 네트워크가 알려주는 행동(action)을 취한다. 예를 들어, DQN에서 강화학습 에이전트는 처음에는 99%의 확률로 랜덤 행동을 취하도록 되어 있고, 특정 시기에 도달하기까지 랜덤 액션을 취하는 확률 ϵ (입실론)은 지속적으로 줄어든다. 그리고 ϵ 이 1%가 되면 감소를 멈추고 1%로 고정되게 된다. 이러한 방법론을 바로 '감소하는 입실론 그리디(decaying epsilon-greedy) 방식'이라고 한다. 감소하는 입실론 그리디(decaying epsilon-greedy)에 나오는 두 가지 하이퍼파라미터는 아래와 같다.

exploration_fraction: 언제까지 감소시킬 것인가? 0.5가 기본값으로 설정되어 있으며, 이 경우에는 총 timesteps의 50%가 될 때까지 랜덤액션을 취할 확률 ϵ 이 계속 줄어들게 된다.

exploration_final_eps: 최종 ϵ 값이다. 기본 값은 0.01로 설정되어 있다. 이 의미는 ϵ 이 계속 줄어 들다가 0.01이 되면 감소를 멈추고 ϵ 값이 0.01로 유지되는 것이다. 훈련이 완료되더라도 1%의 랜덤액션을 계속 취하게 되는 셈이다.

Tip. 연속적인 샘플들(samples)간의 강한 상관관계를 줄인다?
"연속적인 샘플들 간의 강한 상관관계를 줄인다"라는 말이 어려울 수 있을 것이다. 좀 더 쉽게 설명을 해보겠다. 게임을 할 때마다 다른 패턴을 발견하게 된다. 예를 들기 위해 문제를 아주 쉽게 제한해 보겠다. 플레이 방식에 따라 10가지 플레이 패턴이 나온다고 하자. 그 중에 7번째 패턴으로 플레이하면 게임 점수가 100점으로 가장 좋다. 그런데 1번째 패턴으로 플레이하면 성능이 50점 정도가 나온다. 그리고 나머지 8개 패턴은 다 10점 이하의 점수를 내놓는다고 치자.

강화학습을 진행할 때 리플레이 메모리가 없이 들어오는 순서대로 바로 학습을 시키는 상황을 가정하겠다. 에이전트가 1번 패턴에서부터 학습을 하게 된다고 하면, 에이전트의 머릿 속에는 금방 1번 패턴만 가득차게 될 것이다. 그러면서 자연스럽게 Q 함

수는 1번 패턴이 최고라고 익히게 되고, 결국 에이전트는 1번 패턴의 지역최적해(local minima)에 갇히게 된다.

이러한 문제는 Long-ji Lin의 'Reinforcement learning for robots using neural networks'에서 다루어진 바가 있고, 이를 해결하기 위해 경험 재생(experience replay)라는 솔루션이 제시되었다.

DQN에서는 이런 연속적인 기록의 강한 상관관계를 미니배치와 리플레이 메모리를 통해 해결했고, 향후에 나오게 되는 Actor-Critic 유형의 알고리즘은 멀티쓰레딩(multi-threading)⁹ 환경에서 다양한 에이전트를 동시에 띄워서 다양한 샘플들(samples)을 동시에 얻게 만들었고 이를 통해 샘플들 간의 강한 상관관계를 줄였다.

Deep Q-Network 알고리즘 구현체 설명

딥마인드 논문에 적혀있는 DQN의 알고리즘을 살펴보겠다. 그리고 알고리즘의 각 라인에 해당하는 OpenAI의 DQN 구현체 코드를 확인하며 구현체를 이해해 보도록 하겠다.

[그림 3] Open AI의 DQN 구현체 코드

```
Algorithm 1 Deep Q-learning with Experience Replay
Initialize replay memory D to capacity N
Initialize action-value function Q with random weights
for episode = 1, M do
  Initialize sequence  $s_1 = \{s_1\}$  and preprocessed sequence  $\phi_1 = \phi(s_1)$ 
  for  $t = 1, T$  do
    With probability  $\epsilon$  select a random action  $a_t$ 
    otherwise select  $a_t = \arg \max_a Q^*(\phi(s_t), a; \theta)$ 
    Execute action  $a_t$  in env and observe reward  $r_t$  and image  $s_{t+1}$ 
    Set  $\phi_{t+1} = \phi(s_{t+1})$  and preprocess  $\phi_{t+1} = \phi(s_{t+1})$ 
    Store transition  $(\phi_t, a_t, r_t, \phi_{t+1})$  in D
    Sample random minibatch of transitions  $(\phi_j, a_j, r_j, \phi_{j+1})$  from D
    Set  $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$ 
    Perform a gradient descent step on  $(y_j - Q(\phi_j, a_j; \theta))^2$  according to equation 3
  end for
end for
```

Line 1) Initialize replay memory D to capacity N

리플레이 메모리 D를 N의 크기로 초기화한다. mario-rl-tutorial 프로젝트의 해당 위치로 이동해보자. deepq/deepq.py(폴더명/파일명) 파일을 열고 198라인으로 이동한다.

```
replay_buffer = ReplayBuffer(buffer_size)
beta_schedule = None
```

deepq/deepq.py 198라인부터 199라인 두 줄이 바로 리플레이 메모리를 생성하는 곳이다.

Line 2) Initialize action-value function Q with random weights

Q 함수를 초기화하는 부분으로, 행동 가치 함수(action-value function) Q를 임의의 가중치(weight)로 초기화한다. 우리는 CNN과 fully connected layer를 조합해서 Q 함수를 구현한다. 텐서플로우 명령어로 모델의 구조를 설정한다.

슈퍼마리오 튜토리얼 1편¹⁰에서 우리는 Q 함수의 구성을 정의한 적이 있다. 여기에서 우리는 매개 변수 만을 설정했다.

```
# 4. Create a CNN model for Q-Function
model = cnn_to_mlp(
    convs=[(32, 8, 4), (64, 4, 2), (64, 3, 1)],
    hiddens=[256],
    dueling=FLAGS.dueling
)
```

위 코드는 3개의 CNN(convolutional neural network)의 구조를 설정하고, CNN의 결과값을 fully connected layer에 연결하는 형태로 Q 함수를 구성한다. 우리가 원하는 모델의 매개변수를 받아서 진짜로 구현하는 구현체를 살펴보도록 하자.

deepq/models.py 파일의 33라인부터 90라인까지 전체가 Q 함수 설계를 정의하는 스크립트다. 지면 상 코드를 다 실을 수 없어 링크와 QR코드로 대체한다. deepq/models.py 링크: <https://goo.gl/E53fM3>

【 그림 4 】Q 함수 설계를 정의하는 스크립트



여기에서 cnn_to_mlp() 라는 함수는 람다(lambda) 기법으로 구현이 되어 있다. cnn_to_mlp 함수는 convs, hiddens, dueling 파라미터를 사용해 CNN과 MLP의 구조를 설정한다. 하지만, cnn_to_mlp 함수는 실제 텐서플로우 그래프를 만들어서 반환하지는 않는다. cnn_to_mlp() 함수는 실제 텐서플로우 레이어들을 반환하지 않고 람다 함수를 반환한다. 람다 함수는 CNN과 MLP 구조에 대한 값만 받은 상태이고 아직 inpt, num_actions, scope, reuse, layer_norm 같은 값들을 받지 못한 상태다. 이 람다 함수는 향후에 build_graph.py 에서 inpt, num_actions, scope, reuse, layer_norm 값들과 함께 호출되어 실제 텐서플로우 그래프를 생성한다.

Tip. Python lambda 문법을 활용하여 함수의 일부만 구현하기

Python에서는 람다 문법을 활용해 좀 더 자유로운 표현을 할 수 있다. 람다 기법에 익숙하지 않은 분들은 cnn_to_mlp() 함수를 보면 이해하는 데 어려움이 있을 것 같다. 최소한의 예시로 이해를 돕도록 해보겠다. 예시 코드는 mario-rl-tutorial Github에 lambda_ex1.py 라는 파일에 구현해두었다.

t2() 함수는 람다 함수를 반환한다. t2() 함수는 1이라는 값을 받아서 a라는 매개변수에 값을 설정한 상태로 람다 함수를 반환한다. t2(1) 함수의 반환값을 lambda_fun 이라는 변수에 할당한다. lambda_fun 변수는 특정한 스칼라 값이 아니라 람다 함수가 된다. 그래서 우리는 다시 lambda_fun 함수를 호출할 수 있다. 이 때, 우리는 t2() 에서 설정하지 않았던 매개변수 b와 c 값을 설정하여 호출한다.

```
def t(a,b,c):
    print(a,b,c)
```

```
def t2(n):
    return lambda *args, **kwargs: t(a=n, *args, **kwargs)

lambda_fun = t2(1)
lambda_fun(b=2, c=3)
# 1 2 3
```

cnn_to_mlp() 함수에 CNN의 구조와 MLP의 구조를 매개변수로 전달하면, q_func 람다 함수가 반환된다. 하지만, 아직 모델 생성이 끝나지 않았다. 힘들더라도 조금만 더 힘내기 바란다.

deepq/deepq.py 파일의 168라인부터 187라인까지는 우리가 만든 q_func 람다 함수와 여러 매개변수를 활용해 act, train, update_target, debug 함수를 만든다.

```
# Create all the functions necessary to train the model

sess = U.make_session(num_cpu=num_cpu)
sess.__enter__()

def make_obs_ph(name):
    return U.BatchInput((84,84,4), name=name)

act, train, update_target, debug = build_graph.build_train(
    make_obs_ph=make_obs_ph,
    q_func=q_func,
    num_actions=env.action_space.n,
    optimizer=tf.train.AdamOptimizer(learning_rate=lr),
    gamma=gamma,
    grad_norm_clipping=10,
    param_noise=param_noise
)
act_params = {
    'make_obs_ph': make_obs_ph,
    'q_func': q_func,
    'num_actions': env.action_space.n,
}
```

Line 3) for episode = 1, M do

에피소드를 1부터 M까지 반복한다. 여기서 에피소드란 게임 한 판을 의미한다. 우리가 실행하고 있는 mario-rl-tutorial 예제에서는 M 대신 max_timesteps 라는 변수로 최대 실행 횟수를 설정해 두었다. deepq/deepq.py 224라인에 해당 반복문이 설정되어 있다.

```
for t in range(max_timesteps):
```

이 train.py 훈련 스크립트를 실행할 때 max_timesteps를 설정할 수 있게 구현했다. 예를 들어 1000스텝만 실행하고 싶다면 이렇게 훈련 스크립트를 실행하면 된다.

```
python train.py --timesteps=1000
```

Line 4) Initialise sequence $s_1 = \{x_1\}$ and preprocessed

sequenced $\phi_1 = \phi(s_1)$

가장 처음의 상태(state, s_1)를 설정한 후 전처리(preprocess)한다.

```
reshape_obs = np.reshape([obs], (1, 84, 84, 1))
history = np.append(reshape_obs, history[:, :, :3], axis=3)
processed_obs = np.reshape([history], (84, 84, 4))
```

Line 5) for t = 1, T do

t가 1부터 T가 될때까지 반복한다. 여기서 T는 게임의 한 에피소드가 끝나는 시점을 의미한다. 즉, 게임이 끝날 때까지 반복한다는 뜻이다. 우리가 실행해본 mario-rl-tutorial 구현체는 한 에피소드를 위한 반복문을 따로 설정하지는 않았다. 대신, done 변수의 결과에 따라 에피소드가 종료될 때 환경을 reset하는 등의 처리를 해주고 있다.

```
if done:
    obs = env.reset()
    episode_rewards.append(0.0)
    reset = True

    history = np.stack(
        (next_state, next_state, next_state, next_state), axis=2)
    history = np.reshape([history], (1, 84, 84, 4))
else:
    history = next_history
```

그리고 에피소드가 끝날 때마다 tensorboard에 로그를 남긴다.

```
if done and print_freq is not None and len(episode_rewards) % print_freq == 0:
    logger.record_tabular("steps", t)
    logger.record_tabular("episodes", num_episodes)
    logger.record_tabular("reward", epi_reward)
    logger.record_tabular("mean 100 episode reward", mean_100ep_reward)
    logger.record_tabular("% time spent exploring", int(100 * exploration.value(t)))
    logger.dump_tabular()
```

Line 6) With probability ϵ select a random action a_t

ϵ 의 확률로 랜덤한 액션 a_t 를 선택한다.

Line 7) otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$

그렇지 않으면 $a_t = \max_a Q^*(\phi(s_t), a; \theta)$ 공식으로 다음 a_t 를 선택한다. 랜덤한 행동을 취하는 ϵ 은 가장 먼저 LinearSchedule 이라는 클래스로 정의되고, 위치는 deepq/deepq.py 파일의 201라인이다.

```
# Create the schedule for exploration starting from 1.
exploration = LinearSchedule(schedule_timesteps=int(exploration_fraction *
    max_timesteps),
    initial_p=1.0,
    final_p=exploration_final_eps)
```

그리고 t 스텝에서의 ϵ 값을 조회하는 코드는 231라인에 있다.

```
update_eps = exploration.value(t)
update_param_noise_threshold = 0.
```

그리고, 247라인에서 우리가 만들어왔던 action() 함수에 현재 상태값을 넣으면서 ϵ 값을 같이 넣어준다.

```
action = act(np.array(processed_obs)[None], update_eps=update_eps, **kwargs)[0]
```

이렇게 act() 함수에 전처리된 상태값과 ϵ 값을 같이 넣어주면, ϵ 값에 따라 랜덤한 행동이 반환되거나 e-greedy 방식에 따른 최적의 행동이 반환된다. 여기에서 사용된 act() 함수가 어떻게 구현되어 있는지 build_graph.py 안의 정의를 확인해보자.

```
build_graph.py의 build_act() 함수
def build_act(make_obs_ph, q_func, num_actions, scope="deepq", reuse=None):
    with tf.variable_scope(scope, reuse=reuse):
        observations_ph = U.ensure_tf_input(make_obs_ph("observation"))
        stochastic_ph = tf.placeholder(tf.bool, (), name="stochastic")
        update_eps_ph = tf.placeholder(tf.float32, (), name="update_eps")

        eps = tf.get_variable("eps", (), initializer=tf.constant_initializer(0))

        q_values = q_func(observations_ph.get(), num_actions, scope="q_func")
        deterministic_actions = tf.argmax(q_values, axis=1)

        batch_size = tf.shape(observations_ph.get())[0]
        random_actions = tf.random_uniform(tf.stack([batch_size]), minval=0,
maxval=num_actions, dtype=tf.int64)
        chose_random = tf.random_uniform(tf.stack([batch_size]), minval=0, maxval=1,
dtype=tf.float32) < eps
        stochastic_actions = tf.where(chose_random, random_actions, deterministic_
actions)

        output_actions = tf.cond(stochastic_ph, lambda: stochastic_actions, lambda:
deterministic_actions)
        update_eps_expr = eps.assign(tf.cond(update_eps_ph >= 0, lambda: update_
eps_ph, lambda: eps))
        act = U.function(inputs=[observations_ph, stochastic_ph, update_eps_ph],
            outputs=output_actions,
            givens=(update_eps_ph: -1.0, stochastic_ph: True),
            updates=[update_eps_expr])
        return act
```

우리가 build_act() 함수를 실행해서 얻는 act() 함수는 U.function()

이라는 함수를 활용해 만들어낸다. U.function()은 원래 텐서플로우에서 지원하는 함수는 아니고 OpenAI에서 텐서플로우 연산을 단순한 함수 형식으로 변환한 것으로, 필요할 때 함수만 호출하면 텐서플로우 연산을 돌릴 수 있도록 만든 모듈이다.

여기에 정의된 act()함수는 ϵ 값과 4 프레임의 상태값을 받아서 다음에 취할 행동(action)을 반환하는 함수다. U.function을 활용해 입력 값을 넣으면 행동을 반환해주는 함수를 만들어 낸 디자인 패턴이 필자는 너무 마음에 든다. 텐서플로우의 복잡한 연산을 거치지 않고 심플하게 함수로만 호출해서 처리할 수 있게 만들다니 개발자로서 그저 대단하다는 생각만 든다.

Line 8) Execute action a_t in emulator and observe reward r_t and image x_{t+1}

에뮬레이터에서 다음 행동 a_t 를 실행하고 다음 보상 r_t 와 다음 상태 x_{t+1} 을 받는다. deepq/deepq.py 파일에서 249 라인에 있다. 환경에서 action을 실행한 후에 새로운 state(new_obs)와 reward(rew), done을 반환 받는다.

```
new_obs, rew, done, _ = env.step(action)
```

Line 9) Set $s_{t+1} = s_t$, a_t , x_{t+1} and preprocess $\phi_{t+1} = \phi(s_{t+1})$
 $s_{t+1} = s_t$, a_t , x_{t+1} 을 세팅하고 $\phi_{t+1} = \phi(s_{t+1})$ 을 전처리한다. deepq/deepq.py 파일의 251라인부터 255라인까지이다. 새로운 상태를 4프레임 stack에 쌓아 넣는다.

```
obs = new_obs

next_state = np.reshape([new_obs], (1, 84, 84, 1))
next_history = np.append(next_state, history[:, :, :, :3], axis=3)
processed_new_obs = np.reshape([history], (84, 84, 4))
```

먼 길을 달려왔다. [그림 3]의 DQN 알고리즘을 다시 한 번 확인해 보기를 바란다.

Line 10) Store transition (ϕ_t , a_t , r_t , ϕ_{t+1}) in D
(ϕ_t , a_t , r_t , ϕ_{t+1}) 기록(trajecotory)를 리플레이 메모리 D에 넣는다.

```
# Store transition in the replay buffer.
replay_buffer.add(processed_obs, action, rew, processed_new_obs, float(done))
```

Line 11) Sample random minibatch of transitions (ϕ_i , a_i , r_i , ϕ_{i+1}) from D
리플레이 메모리 D에서 랜덤으로 미니배치 기록(minibatch

trajectories)인 ϕ_i , a_i , r_i , ϕ_{i+1} 를 추출한다. deepq/deepq.py 파일에서 278,279 라인에 있다. replay_buffer.sample(batch_size)가 리플레이 메모리에서 batch_size 만큼의 기록을 미니배치로 추출하는 함수다.

```
obses_t, actions, rewards, obses_tp1, dones = replay_buffer.sample(batch_size)
weights, batch_idxes = np.ones_like(rewards), None
```

Line 12) Set $y_i = r_i$ (for terminal ϕ_{i+1}) $y_i = r_i + \gamma \max_a Q(\phi_{i+1}, a; \theta)$ (for non-terminal ϕ_{i+1})
 $y_i = r_i$ (게임이 끝인 경우 ϕ_{i+1}) / $y_i = r_i + \gamma \max_a Q(\phi_{i+1}, a; \theta)$ (게임이 아직 안끝난 경우 ϕ_{i+1})

Line 13) Perform a gradient descent step on $(y_i - Q(\phi_i, a_i; \theta))^2$ according to equation 3
공식 3에 따라 $(y_i - Q(\phi_i, a_i; \theta))^2$ 그라디언트 디센트를 실행한다.

```
td_errors = train(obses_t, actions, rewards, obses_tp1, dones, weights)
```

여기에서 호출된 train() 함수가 구현된 build_graph.py의 build_train() 함수로 다시 가보겠다.
지면 상 코드를 다 실을 수 없어 링크와 QR코드로 대체하겠다.

deepq/build_graph.py 코드 링크 : <https://goo.gl/2Kgi3S>



build_graph.build_train() 함수에 여러 가지 매개변수를 넣으면 조건에 맞는 모델을 생성함과 동시에, 상태를 넣으면 행동을 알려주는 act() 함수, 모델을 학습시키는 train() 함수, 타겟 네트워크를 업데이트 시키는 update_target() 함수, 디버깅을 위한 debug 변수를 반환한다. Debug 변수는 트레이닝 중간에 생성되는 q_value 등 중간 산출물을 넣어서 같이 보내기 위한 통로로 쓰인다. build_train() 함수는 Q함수를 업데이트 시키는 로직이 구현되어 있다. (1) 튜토리얼 초반에 선언했던 q_func 람다 함수에 나머지 값들을 마저 넣어서 Q 네트워크 모델과 타겟 Q 네트워크 모델을 만들어낸다.

```
# q network evaluation
q_t = q_func(obs_t_input.get(), num_actions, scope="q_func", reuse=True) #
reuse parameters from act
```

~

```
q_func_vars = U.scope_vars(U.absolute_scope_name("q_func"))

# target q network evaluation
q_tp1 = q_func(obs_tp1_input.get(), num_actions, scope="target_q_func")
target_q_func_vars = U.scope_vars(U.absolute_scope_name("target_q_func"))
```

(2) 그리고 Q 네트워크를 구성하는 변수들을 조회한 후 q_func_vars 저장한다.

```
q_func_vars = U.scope_vars(U.absolute_scope_name("q_func"))
```

(3) 현재 Q 네트워크와 업데이트의 타겟이 되는 값을 구한다.

```
# q scores for actions which we know were selected in the given state.
q_t_selected = tf.reduce_sum(q_t * tf.one_hot(act_t_ph, num_actions), 1)

q_tp1_best = tf.reduce_max(q_tp1, 1)
q_tp1_best_masked = (1.0 - done_mask_ph) * q_tp1_best

# compute RHS of bellman equation
q_t_selected_target = rew_t_ph + gamma * q_tp1_best_masked
```

(4) 에러 값을 구한다.

```
# compute the error (potentially clipped)
td_error = q_t_selected - tf.stop_gradient(q_t_selected_target)
errors = U.huber_loss(td_error)
weighted_error = tf.reduce_mean(importance_weights_ph * errors)
```

(5) 그라디언트 디센트를 수행한다. 여기서 grad_norm_clipping이라는 값이 있는데, 학습을 시키다 보면 종종 급격한 그라디언트(gradient) 값으로 인해 모델이 망가지는 경우가 있다. 이를 막기 위해 tf.clip_by_norm() 함수를 사용한다. tf.clip_by_norm() 함수는 그라디언트를 정규화한 특정 값을 넘어가면 그라디언트를 잘라준다.

```
# compute optimization op (potentially with gradient clipping)
if grad_norm_clipping is not None:
    optimize_expr = U.minimize_and_clip(optimizer,
                                       weighted_error,
                                       var_list=q_func_vars,
                                       clip_val=grad_norm_clipping)
else:
    optimize_expr = optimizer.minimize(weighted_error, var_list=q_func_vars)
```

이렇게 DQN 알고리즘 구현체의 코드 리뷰를 마쳤다. 상당히 길고 험한 과정이었다. 그리고 OpenAI의 baselines 오픈소스 코드의 추상화 레벨이 아주 높기 때문에 초심자의 경우에는 이해하기가 상당히 어려울 것이라 생각한다.

최고의 작가가 되는 첫 걸음은 필사(베껴 쓰기)라고 배웠다.

최고의 개발자가 되기 위한 첫 걸음 역시 우수한 코드를 보고 따라서 작성해보는 것이 아닐까 생각된다. 조금은 벅차더라도, 잘 정돈된 코드를 보고 익히며 소화해 보도록 하자.

이번 튜토리얼에서는 알고리즘의 구현체에 대한 코드레벨 관련 설명을 하였다. 다음 편에서는 Actor-Critic 강화학습 알고리즘을 소개하도록 하겠다.

***1** 참고 | OpenAI, Github baselines : <https://github.com/openai/baselines> ***2** 참고 | <https://brunch.co.kr/@kakao-it/144> ***3** 논문 | Lin, L. (1993). Reinforcement Learning for Robots Using Neural Networks, CMU ***4** 참고 | <https://brunch.co.kr/@kakao-it/144> ***5** 참고 | <https://haifengl.wordpress.com/2016/02/29/there-is-no-big-data-in-machine-learning/> ***6** 참고 | RL Course by David Silver - Lecture 8: Integrating Learning and Planning. <https://youtu.be/ltMutbeOHtc> ***7** 참고 | RL Course by David Silver - Lecture 5: Model Free Control. https://youtu.be/0g4j2k_Ggc4 ***8** 참고 | Sutton, R. & Barto, A. (2017). Reinforcement Learning: An Introduction 2nd edition ***9** 설명 | 멀티쓰레딩은 프로세스 (현재 실행중인 프로그램)에 두 개 이상의 스레드(thread)를 할당하여 하나의 프로세스에서 병렬적으로 여러 개 작업을 처리하는 것을 일컫는다. 예컨대 멀티쓰레딩 환경을 통해 카카오톡에서 채팅을 하면서 동시에 파일을 다운로드 받을 수 있다. ***10** 참고 | <https://brunch.co.kr/@kakao-it/144>

참고자료
[1] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing Atari with Deep Reinforcement Learning, NIPS 2013 Workshop, 2013 **[2]** Chris Hoyean Song, Github mario-rl-tutorial : <https://github.com/chris-chris/mario-rl-tutorial>

Do you know GAN? (2/2)

GAN은 생성 모델로써 이미지를 만들어내는 생성기와 생성기로부터 만들어진 이미지와 실제 이미지를 구별하는 구별자가 서로 경쟁을 하는 가운데 서로의 성능이 향상되는 재미있는 구조를 갖고 있다. GAN이 각광을 받는 이유는 성능이 좋기도 하지만 이렇게 독특한 구조 때문이기도 하다. 그동안 있어 왔던 생성 모델들과는 궤를 달리하는 형태이고 이로부터 다양한 아이디어들이 나오면서 최근에는 상당히 높은 질의 이미지들을 만드는 수준에 이르렀다. 지난 글에서 GAN의 기본 원리와 배경 이론에 대해 살펴 보았다면, 이번 글에서는 GAN에 대한 기본적인 이해를 바탕으로 GAN이 기존 모델과 다른 점이 무엇인지 그리고 어떤 장단점이 있는지 조금 더 심화된 내용을 다루고자 한다.

Deep Convolutional GAN

초창기 GAN에 대해 얘기하려고 하면 빼놓을 수 없는 연구가 바로 DCGAN(deep convolutional GAN)* 이다. 지금은 GAN이 매우 뛰어난 결과들을 보여주고 있지만, 초기 결과는 아이디어의 참신성에 비해 그리 인상적이지 않았다. GAN 구조가 학습시키기 매우 어려웠다는 것도 여러 이유 중 하나였다. DCGAN이 나온 이후, GAN 결과들이 매우 급격하게 발전하기 시작했다. DCGAN 논문이 기여한 바를 정리해보면, 다음과 같다.

- 대부분의 상황에서 언제나 안정적으로 학습이 되는 구조를 제안하였다.
- word2vec과 같이 DCGAN으로 학습된 생성기는 벡터 산술 연산이 가능한 성질을 갖고 의미론적(semantic)으로 신규 샘플을 생성할 수 있다는 것을 보여 주었다.

DCGAN 논문은 GAN이 잘 학습되는 구조를 매우 세세한 가이드 라인으로 제시한 연구였다. 이 논문 이후에 나온 대부분의 GAN 연구들이 어떤 형태로든 DCGAN 구조를 바탕으로 하고 있다고 할 정도로 DCGAN은 매우 잘 확립된 구조이다. 일단 DCGAN 에서 제시한 가이드 라인대로 GAN 구조를 짜면 상당히 안정적으로 학습이 된다. 이런 구조를 발견하기 위해서 대학원생들이 얼마나 힘들었는지 논문의 한 구절에서 언뜻 느껴볼 수 있다.

"그러나 광범위한 모델 탐구 끝에 우리는 다양한 데이터셋을 대상으로 안정적인 학습을 할 수 있고, 더 높은 해상도와 더 깊은 생성 모델(deeper generative models)을 가능하게 하는 구조의 모음(a family of architectures)을 발견했다."

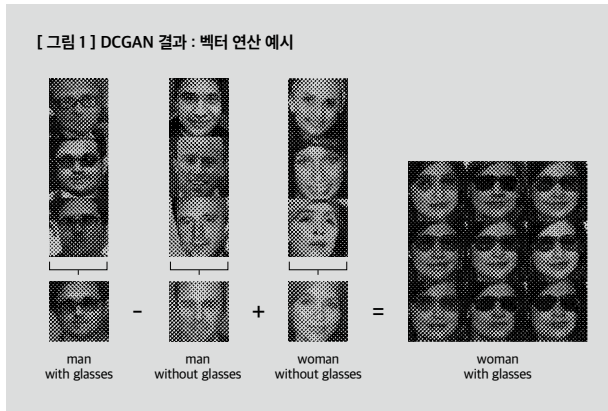
DCGAN은 이름에서 알 수 있듯이 convolution 구조를 GAN에 잘 결합한 것이다. CNN(convolutional neural network)이 지도 학습(supervised learning)에서 매우 큰 성공을 거둔 것에 비해 비지도 학습(unsupervised learning)에서는 상대적으로 잘 사용되지 못했다. 이러한 측면에서 DCGAN 논문은 지도 학습과 비지도 학습에서의 격차를 줄이는 데 큰 역할을 했다고 평가된다.

DCGAN 결과에서 가장 재미있는 부분은 아래와 같이 생성기의 입력으로 들어가는 z 은닉 공간(latent space)에서 벡터 산술 연산이 가능하다는 점이다. 가장 흔한 예시로 word2vec 연구가 있다. 다음 수식을 계산하고 답을 추정해보자.

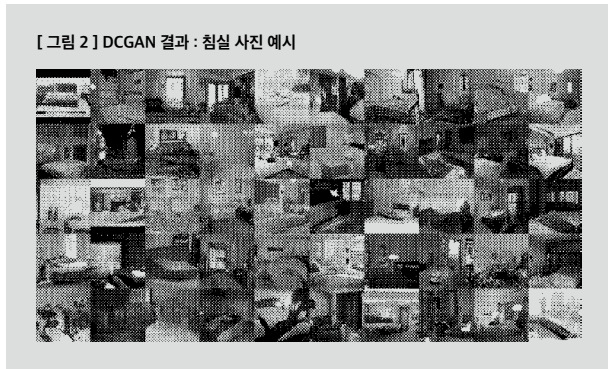
KING - MAN + WOMAN

사람은 생각보다 쉽게 위 구조 속에서 QUEEN이란 단어를 연상할 수 있지만 컴퓨터에게 이런 연산은 단어의 의미를 이해하고, 그에 맞는 새로운 단어를 찾는 등의 매우 고차원의 처리가 필요한 어려운 문제다. 기존의 word2vec 연구에서는 뉴럴 네트워크를 사용하여 말뭉치에서 실제로 단어 간의 관계를 학습하는 것을 보여 주었고, DCGAN은 이를 말뭉치가 아닌 이미지에서 하는 것이 가능하다는

것을 시사했다. 아래 [그림1]이 바로 그 결과이다.



실제로 모두 DCGAN으로 생성된 결과들이다. 상당히 진짜같은 결과 만으로도 놀라운 데, 이미지가 갖는 의미를 바탕으로 직관적인 벡터 산술이 가능하다는 것을 알 수 있다. 안경을 쓴 남자와 안경을 쓰지 않은 남자 그리고 안경을 쓰지 않은 여자를 생성하게 하는 입력 값들이 은닉 공간에 각각 벡터로 존재하고 있을 텐데, 각각의 벡터를 서로 빼고 더해 주면 최종적으로는 안경을 쓴 여자를 생성하는 입력 벡터를 찾을 수 있다는 것이다. 물론 z벡터 하나만 생성기에 입력해서는 깔끔한 결과가 나오지 않기 때문에 세 개 정도를 추리고 그 세 개의 평균 z벡터를 사용해서 결과를 만든 것이기는 하다. 하지만 신기한 것은 매한가지다. 어떻게 보면 네트워크가 영상의 의미를 이해했다고 생각할 수 있다. 이 외에도 [그림2]와 같이 침실을 생성한 결과 그림들을 보면 작은 그림이긴 하지만 꽤나 그럴듯한 결과를 만들어 냈다는 것을 확인할 수 있다.

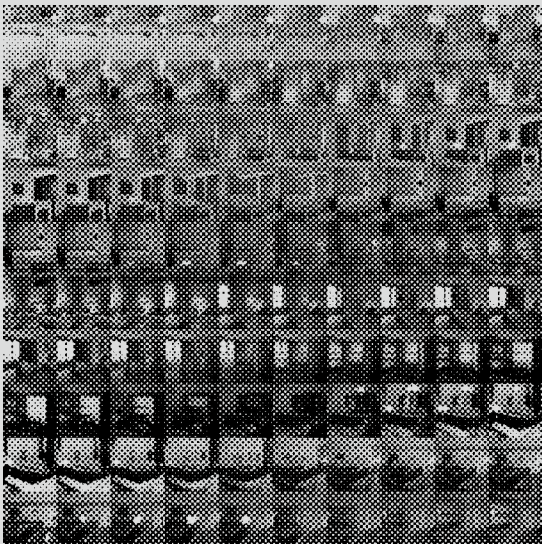


논문에서 "공간을 걷는다" 라고 표현했듯이 [그림3]처럼 은닉 공간에서 천천히 벡터의 값을 바꾸면, 생성기가 내보내는 이미지가 하나의 침실에서 다른 침실로 위화감 없이 부드럽게 변화하는 것을 볼 수 있다. 특히 벽이었던 부분이 자연스럽게 하나의 창으로 변하는 것을 보면 매우 놀랍다.

글 | 유재준 jaejun2004@gmail.com

카이스트 바이오및뇌공학과 학부를 졸업하고 동 대학원 박사 과정을 밟고 있는 공돌이입니다. 바이오영상신호처리 연구실에서 전통적인 신호처리 이론을 이용한 영상 복원이나 대수적 위상수학을 이용한 뇌 네트워크 분석을 연구하였으나, 최근에는 기계학습에 빠져서 이를 이용한 의료 영상 복원에 적용해보고 있습니다. 처음 가본 NIPS 2016 학회에서 GAN을 접하고 매우 흥분하여 열심히 가지고 놀아보고 있지만 사실 당장 졸업에 필요한 연구 주제와는 전혀 관계가 없다는 것이 함정입니다. 딥러닝은 이제 갓 1년차 공부 중인 초짜 대학원생이며, 혼자 공부하던 것을 블로그에 정리하던 것이 취미였는데 많은 분들이 좋아해 주셔서 매우 신이 나 있는 상태입니다. 최근 졸업이 다가오면서 외부활동이 점차 줄어들고 있지만, 앞으로 더 많은 분들과 교류하고 배워 이 분야에서 내가 아는 것을 꾸준히 나누는 사람이 되는 게 꿈입니다.

【그림 3】“은닉 공간에서 돌아다니기”



만약 생성기가 단순히 영상을 외워서 보여줄 뿐이라면 주어진 특정 입력에 대해 특정 이미지를 내보내는 일대일 대응 함수를 학습한 것으로 생각할 수 있다. 이럴 경우 은닉 공간에서 굳이 부드러운 변화가 있을 이유가 없다. 바로 옆의 z 벡터가 전혀 다른 샘플과 일대일로 연동될 수 있기 때문이다. 이렇듯 은닉 공간에서 벡터 연산이 가능하다는 것과 입력에 변화를 줬을 때 생성되는 결과가 부드럽게 변하는 것을 보는 분석이 중요한 이유는 우리가 학습시킨 GAN의 생성기가 일대일 대응 함수와 같이 매우 단순하고 의미 없는 함수(mapping)를 학습한 것이 아닌란 것을 시사하기 때문이다.

이렇게 수많은 이미지를 표현할 수 있는 정보들을 포괄하면서도 부드러운 변화에 대응할 수 있는 함수를 학습하기 위해서는 은닉 공간을 잘 정하는 것도 매우 중요한 일이다. GAN에서는 보통 z 은닉 공간은 고차원의 가우시안 분포를 많이 사용한다. 적절한 가정 하에서 충분히 복잡한 함수를 사용하여 대응시킬 수만 있다면 임의의 d 차원 분포는 정규 분포를 따르는 d 개의 변수로 나타낼 수 있다는 것이 알려져 있기 때문이다².

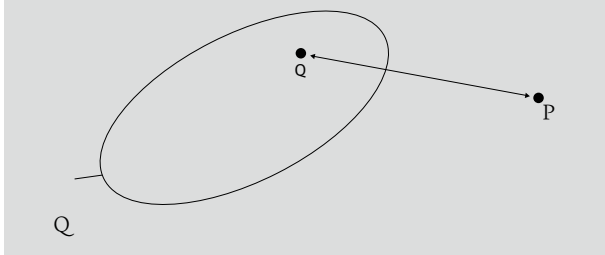
기존 생성 모델과 GAN의 차이점

그렇다면 GAN은 기존의 생성 모델들과 어떤 면이 다르기에 이렇게 비교적 또렷한 이미지를 만들 수 있는 것일까? GAN의 특징이자 기존 생성 모델들과 구별되는 가장 큰 차이점은 바로 GAN이 사실 샘플러(sampler)라는 것이다. 즉, 직접적으로 데이터의 분포를 학습하는 형태가 아니라 하나의 입력이 들어갔을 때 하나의 출력을 주는 샘플러를 학습한다는 독특한 특징을 지닌다.

조금 더 자세히 이해하기 위해 “확률 모델을 학습한다” 는 것에 대해 생각해보면, 모델을 추정한다는 것은 일반적으로 우리가 알고 싶은 P 라는 진짜 데이터의 분포가 어딘가에 있을 때,

- (1) 이로부터 얻은 샘플들이 독립항등분포(independent and identically distributed, I.I.D.)의 성격을 띄고 있는 것으로 관측되었다고 가정한 후,
 - (2) Q 라는 모수 모델 클래스(parametric model class)를 만들어,
 - (3) 그 안에서 P 와 가장 가까운' 모델의 모수를 찾아내는 것을 말한다.
- 가장 간단한 예시로 데이터의 분포가 정규 분포를 따를 것이라 가정하여 가우시안 모델을 세우고, 현재 내가 갖고 있는 데이터를 가장 잘 설명할 수 있는 평균과 분산 값을 찾는 과정을 생각해볼 수 있다. 이를 위해서는 P 와 Q 의 '차이' 혹은 '거리'를 계산할 수 있어야 한다. 그러면 구한 두 분포 사이의 거리를 줄여나가는 방향으로 모델의 모수들을 조정할 수 있다. 이 과정을 적합(fitting)이라고 한다. GAN도 Jensen-Shannon divergence라는 측도를 사용하여 분포 간의 거리를 계산하고 이를 최소화하는 방향으로 생성기를 학습한다고 분석할 수 있다³.

【그림 4】확률 모델 학습



보통 기존의 방식에서는 아래와 같은 Q 에 대한 가정들을 사용한다.

- 쉬운 샘플링이 가능해야 한다.
- 샘플에 대한 모수의 경사(gradient)가 계산 가능해야 한다.
- 우도 함수가 다루기 쉬운 형태여야 한다.

이 중 가장 강력한 가정이 바로 우도 함수(likelihood function)가 계산 가능하다(tractable)는 것이다. 많은 경우 현실의 모델은 계산이 불가능한 형태의 수식으로 나타나는 것을 상기한다면, 기존의 모델들이 얼마나 강력한 가정을 사용하고 있는지 알 수 있다. GAN 형태의 모델들은 임의의 확률 변수 입력 값을 사용하여 비선형 변환 함수를 통과시키면 출력으로 샘플이 하나 튀어 나오는 구조다. 마치 버튼을 누르면 샘플이 튀어 나오는 자판기처럼 생각할 수 있다. GAN 모델들의 독특한 점은 바로 이 부분에서 나온다. GAN 모델들은 다른 확률 모델들과는 달리 우도 함수를 근사하려 하지 않고 샘플을 뽑아주는 함수를 학습했을 뿐이기 때문에 우도 함수 모델링이 필요 없는(likelihood-free) 모델이라고 할 수 있다.

물론 이 부분은 GAN의 특징일 뿐, 장점일 수도 있고 단점일 수도 있다. 데이터 분포에 대한 모델을 특정하지 않고 하나의 샘플을 뽑아서 보여주기 때문에 고정된 모델의 한계에 제약 받지 않고 또렷한 이미지를 보여줄 수 있기도 하지만, 다른 한편으로는 정작 이미지를 잘 뽑더라도 데이터의 분포에 대한 정보를 직접적으로 얻기는 어렵다. 때문에 분포를 알았을 때 시도해볼 수 있는 많은 분석들을 시도할 수 없다. 이 부분에 대해서는 이안 굿펠로우(Ian Goodfellow)의 NIPS 2016 tutorial 논문⁴이나 같은 시기, 같은 워크숍에서 발표된 세바스찬 노워진(Sebastian Nowozin)의 f-GAN⁵ 논문을 참고한다면 조금 더 심화된 내용을 확인할 수 있다.

이외에도 이후에 연구된 WGAN⁶에서는 기존의 GAN이 divergence를 측도로 사용 하기 때문에 생기는 여러 가지 문제를 지적하며 다른 방식의 측도를 제안하는 등, 점차 수식적인 분석과 함께 GAN의 가치 함수 자체를 근본적으로 수정하는 방향으로 연구가 발전되었다. 이를 바탕으로 보다 안정적인 학습과 결과를 보여 주었는데, EBGAN, LSGAN, BEGAN 등 이후 나온 많은 GAN들이 WGAN의 갈래로 분류 될 수 있다.

이렇게 보면 모든 연구가 끝나서 더 이상 할 것이 없는 것처럼 보이고, 점차 이론적인 문제로 깊게 들어가면서 공학자들이 개입할 수 있는 여지가 없는 것 같지만 아직은 풀어야 할 문제가 많다.

학습이 예전에 비해 수월해졌다는 것이지 정말 쉬워졌다는 것을 의미하진 않는다. 또한 네트워크의 학습이 안정적으로 이미지의 질을 보장하지 않는 경우 또한 많다. 수렴은 여전히 어렵고 이어 소개할 mode collapse나 모델 평가 등 아직도 풀어야 할 문제가 산적해 있다. 그런 의미에서 GAN 학습이 어려운 이유를 하나씩 알아 보자.

GAN 학습이 어려운 이유 I: Convergence

지난 글에서 소개했듯이 원 논문에서 GAN에 대한 이론적 근거를 증명해 주었지만 아쉽게도 실제 구현은 이론적 배경이 되는 가정과 거리가 있다. 때문에 GAN 가치 함수를 뉴럴 네트워크를 이용해 풀었을 때 이상적인 전역해로 수렴한다는 보장이 되지 않는다. 게다가 풀어야 하는 문제의 형태부터 이미 쉽게 문제를 풀 수 있는 볼록 함수 형태가 아닌 변수 두 개가 서로 엮여 있는 안장점 문제(saddle point problem)를 고려해야 하기 때문에 GAN은 학습이 매우 어렵기로 유명하다.

실제로도 많은 사람들이 간단한 예제에서도 문제를 풀기가 어려울 수 있는데 더 복잡한 문제에서 GAN 형태가 잘 풀릴 것인지에 대해 의문을 제기한 바 있다. 이 문제를 약간 더 직접적으로 느끼기 위해서 실제로 간단한 예제인 $V(x, y) = xy$ 가

어떻게 생겼는지 그려보면 [그림 5]와 같다.

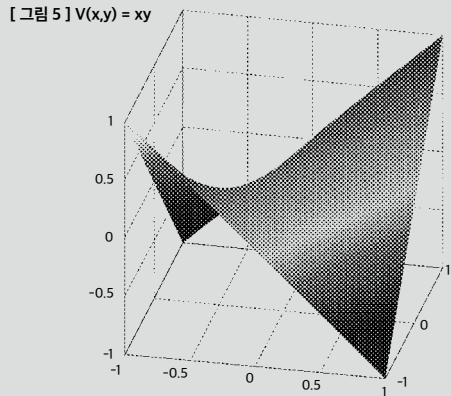
이 문제는 $x = 0, y = 0$ 에서 안장점을 갖는 매우 대표적인 예시다. 그리고 x 와 y 에 대해 최소최대 문제를 풀면 이 안장점이 평형점이라는 것도 쉽게 알 수 있다. 사실 안장점이 모두 평형점이 되는 것은 아니지만, 이 경우 하나의 변수에 대한 작은 변화가 다른 변수에 대해 가치 값을 줄일 수 없기 때문에 평형점이 되는 것이다. 만약 이 문제를 경사하강법(gradient descent)으로 풀면 결과가 평형점 주변에서 수렴하지 못하고 최초 시작점에 따라서 반지름이 정해지는 궤도(orbit) 위에서 영원히 헤매는 것을 확인할 수 있다. 심지어 학습률(learning rate)이 크면, 바깥 방향으로 발산하는 경우도 생길 수 있다.

이를 수식과 함께 확인해보면 조금 더 명확해진다. 학습율 γ 를 고정하고 $n \in \mathbb{N}$ 일 때, 각 변수에 대해 경사하강법으로 번갈아 계산하는 것은 [수식 1]과 같다.

$$\begin{aligned} \text{[수식 1]} \quad x_{n+1} &= x_n - \gamma y_n \\ y_{n+1} &= y_n + \gamma x_n \end{aligned}$$

[수식 1]은 [수식 2]처럼 다시 표현할 수 있고, [그림 5]처럼 $V(x, y) = xy$ 이므로

$$\text{[수식 2]} \quad \begin{bmatrix} x_{n+1} \\ y_{n+1} \end{bmatrix} = \begin{bmatrix} 1 & -\gamma \\ \gamma & 1 \end{bmatrix} \begin{bmatrix} x_n \\ y_n \end{bmatrix}$$



여기서 다시 [수식 3] 으로 바꾸고

$$\text{[수식 3]} \quad \begin{bmatrix} 1 & -\gamma \\ \gamma & 1 \end{bmatrix} = \frac{1}{\alpha} \begin{bmatrix} \alpha & -\alpha\gamma \\ \alpha\gamma & \alpha \end{bmatrix} = \frac{1}{\alpha} \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix}$$

α 를 [수식 4]와 같다고 해보겠다.

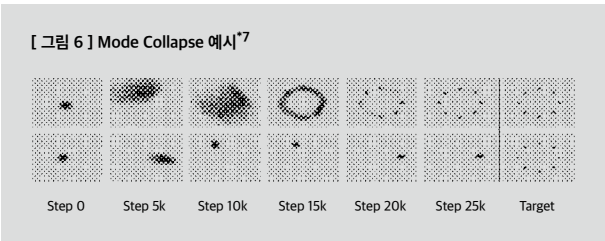
【수식 4】

$$\alpha = \sqrt{\frac{1}{1+\gamma^2}} \in (0, 1), \theta = \cos^{-1} \alpha \in \left(0, \frac{\pi}{2}\right)$$

고등학교에서 회전 행렬에 대해 배운 사람이라면 위의 행렬식이 매우 익숙할 것이다. $\gamma \approx 0$ 일 때, 즉 학습율이 충분히 작아서 $\alpha \approx 1$ 이면 경사하강법의 결과가 언제나 안정한 궤도(stable orbit)로 빠지고, $\alpha < 1$ 인 경우 (x_n, y_n) 이 나선형으로 발산하게 된다.

GAN 학습이 어려운 이유 II: Mode Collapse

앞서 소개한 문제도 그렇지만 GAN 학습의 어려움은 대부분 그 가치 함수의 형태에서 기인한다. 두 번째로 소개할 mode collapse 문제 역시도 GAN의 독특한 가치 함수와 그 문제 풀이 방식 때문에 발생하는 것으로 해석할 수 있다.



[그림 6]은 전형적인 mode collapse 문제의 예시다. 맨 오른쪽의 목표 분포를 보면 가우시안 혼합 모델로 총 여덟 개의 최빈값(mode)이 있는 것을 볼 수 있다. 아래 줄의 그림이 이 목표 분포를 근사하기 위해 GAN으로 여러번 반복하여 학습한 결과다. GAN이 뽑은 샘플들을 보면 각각의 최빈값들을 각 단계마다 돌아가며 방문하는 것을 볼 수 있다. 즉, 원래라면 윗줄과 같이 전체 최빈값들을 보고 목표 분포가 여덟 개의 최빈값을 갖는 분포라는 것을 찾아내야 하지만, GAN은 그렇게 하지 못 하는 모습을 보여준다. 조금 더 직접적인 예를 들자면 숫자가 1부터 8까지 섞여 있는 이미지들로 데이터 분포가 있을 때, 우리는 GAN이 1부터 8까지 모든 숫자들을 만들어낼 수 있기를 바라는데 실제로는 1과 같이 가장 쉬운 한 가지 숫자만 만드는 모델을 학습한다는 것이다. 이런 현상을 하나의 최빈값에만 함몰되는 모델이나 함수를 학습한다고 하여 mode collapse 문제라고 부른다.

사실 우리가 매 단계마다 최적의 구별자 D^* 를 계산할 수 있다면 이런 문제가 생기지 않겠지만, 뉴럴 네트워크로 모델을 만들 경우 수렴이 될 때까지 계산을 매우 여러번 해야 하고 여러 가정이 깨지면서 수렴이 보장되지도 않는다. 따라서 현실적으로는 가치 함수를 각각의 변수에 대해 일정 횟수만큼 번갈아 푸는 방식을 택하는데 이런 방식 때문에 문제가 생기게 된다. 원래 풀고자 하는

GAN의 가치 문제는 [수식 5]와 같은 최소최대 문제이다.

【수식 5】

$$G^* = \min_G \max_D V(G, D).$$

그렇지만 실제 학습을 할 때는 G와 D에 대해 번갈아가며 풀어주기 때문에 뉴럴 넷의 입장에서는 이러한 최소최대 문제와 [수식 6]과 같은 최대최소 문제가 구별되지 않는다.

【수식 6】

$$G^* = \max_D \min_G V(G, D).$$

문제는 최대최소 문제에서 생긴다. [수식 6]의 안쪽부터 살펴보면 G에 대한 최소 문제가 먼저 있기 때문에 생성자의 입장에서선 현재 고정되어 있는(비최적, non-optimal) 구별자가 가장 헛갈려 할 수 있는 샘플 하나만 학습하면 된다. 즉, 가치 함수를 가장 최소화할 수 있는 최빈값 하나만 내보내면 된다. 이렇듯 GAN의 가치 함수 자체와 엮여 있는 문제이기 때문에 mode collapse 문제는 아직도 GAN에서 완전하게 해결되지는 않고 있다.

GAN 학습이 어려운 이유 III: Evaluation

이에 더해 모든 생성 모델이 갖는 고질적인 문제가 바로 평가의 객관성이다. 생성한 이미지의 질을 평가할 수 있는 객관적인 기준을 정하는 것이 매우 어렵기 때문에 새로운 모델이 예전의 모델에 비해 발전한 것인지 평가하는 것이 쉽지 않고 연구의 방향을 잡기도 어렵다.

현재 사용되는 방식을 몇 가지 살펴보자면 대표적인 것이 아마존 메카니컬 터크(Amazon Mechanical Turk)를 사용하여 사람이 직접 평가하도록 하는 방식이다. 그러나 이런 방법은 매우 주관적이고 일관된 평가가 어려우며, 위에서 설명한 mode collapse가 일어난 경우 전혀 모델의 문제점을 파악할 수 없다는 단점이 있다. Mode collapse가 일어난 모델의 경우 생성하는 이미지의 다양성이 부족할 뿐이지 단일 이미지 자체의 품질은 상당히 좋을 수 있기 때문이다.

두 번째로는 inception score라고 하여 구글의 인셉션 이미지 분류 모델에 생성된 이미지를 넣어 나오는 값을 바탕으로 평가를 하는 방식이 있다. 이 방법은 값이 일관되고 어느 정도 객관성을 보장할 수 있다는 장점이 있어 꽤 자주 사용되고 있 다. 하지만 굳이 인셉션 모델을 사용해야하는 이유도 없고 어떤 모델의 경우 인셉션 모델에 특화(overfitting)되어 실제 이미지의 질과는 무관하게 점수가 좋게 나올 수 도 있다는 문제를 안고 있다. 이렇게 앞서 소개한 문제들 외에도 다양한 연구 거리가 남아 있겠지만 세 가지로

크게 정리해 보았다. GAN 연구가 활발하고 매일 하루가 멀다 하고 쏟아지는 만큼 더 이상 연구할 것이 없고 너무 늦었다고 생각할 수도 있으나 알고 보면 아직 가야할 길이 멀다.

‘창GAN기’-믿거나 말거나

마지막으로 GAN에 대한 탄생 비화를 소개하면서 글을 마무리하겠다. 이안 굿펠로우가 최초로 만든 GAN은 multi-layer perceptron(MLP)을 이용한 매우 단순한 형태의 GAN이었다고 한다. 거짓말같지만 단 한 번의 시도만에 바로 성공했는데… 물론 매우 간단한 문제에 대해 적용해봤을 것으로 추측되지만 GAN이 수렴시키기 어렵기로 악명 높다는 것을 생각해보면 솔직히 믿기지 않는 일화다.(마치 박혁거세 설화를 보는 느낌이랄까…)

이안 굿펠로우가 GAN에 대한 아이디어를 처음 떠올린 순간은 몬트리올의 ‘The 3 Brewers’라는 펍에서 친구들과 얘기를 하던 중이었다고 한다. 박사를 마치고 연구실을 떠나는 친구를 송별하는 자리였는데, 그렇게 모인 친구들 중 한 명이 모든 사진의 통계적 정보를 넣어서 스스로 사진을 만들어 낼 수 있는 기계에 대해 얘기를 꺼냈다. 즉석에서 친구들끼리 어떻게 하면 그런 기계를 현실적으로 만들 수 있을 지에 대해 논쟁이 벌어졌다.

존재하는 모든 사진에 대한 통계적인 정보를 얻는 것부터 일단 말이 되지 않으니 불가능한 프로젝트라고 생각하다가 순간 뉴럴 네트워크를 사용해 기계를 가르치면 더 진짜 같은 사진을 만들 수 있지 않을까 하는 생각이 들었다고 한다. 하지만 친구들은 그 아이디어에 대해 부정적이었고, 살짝 열이 받은 이안은 새벽에 술자리에서 돌아오자마자 앉은 자리에서 노트북으로 GAN을 코딩했다고 한다. 그리고 거짓말같이 단 만에 성공했다는데, 이후 인터뷰⁸에서도 “매우 매우 운이 좋았다. 만약 GAN이 한 번에 성공하지 않았다면, 그냥 포기했을 것이다” 라며 스스로도 운이 좋았다고 말한 바 있다.

이 인터뷰 내용이 사실이라면 여기서 우리는 여러가지 교훈을 얻을 수 있다(옛날 이야기에는 언제나 교훈이 있는 법). 문제를 설정하고 풀 때 직관력이 매우 중요하다는 것과 그 직관으로 얻은 아이디어를 바로 실험해보는 실행력이 중요하다는 점, 술자리에서 연구 얘기만 주구장창 하여도 진지하고 재미있게 들어줄 사람들이 있는 집단에 들어가야 한다는 것, 그리고 마지막으로 되는 놈은 뭘 해도 된다고 운도 조금은 좋아야 한다는 것이다. 희망적인 것은 어떤 문제에 대한 직관력은 그 분야와 연관된 깊은 수학적 지식에서 나올 수도 있지만 수많은 시행착오(a. k. a. 삽질)를 바탕으로 한 경험에서 길러진다는 점이다. 그리고 매우 다양하게 많은 시도를 하다보면 올바른 방법을 찾을 확률이

높으니 운이라는 것도 어느 정도 통계에 기대볼 수 있을 것 같다. 이렇게 열정적으로 문제를 풀다보면 비슷한 사람들이 모여 있는 집단(카카오..?)에 갈 수 있는 기회가 생기고, 더 재미있게 잘 연구할 수 있는 선순환이 이루어지지 않을까?

엄밀한 수학 지식을 바탕으로 차근차근 쌓아 올렸을 것 같은 매우 복잡한 이론들도 순간적인 직관에서 시작하는 경우가 매우 많은 것 같다. 그러니 수학이 어려운 공학자들이여 우리 모두 힘을 내자! 수학을 잘하는 수학자들과, 실험을 바탕으로 감이 살아있는 공학자들이 각자의 영역에서 연구를 하며 협업을 통해 문제를 발전시켜 나간다면 언젠가는 정말로 이미지뿐만 아니라 사람의 생각을 모사하는 궁극의 생성 모델 또한 만들어 낼 수 있지 않을까 기대해본다.(필자의 꿈이기도 하다!)

⁷1 논문 | Radford, A. (2016). Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. doi : arXiv:1511.06434. ²2 논문 | Doersch, C. (2016). Tutorial on Variational Autoencoders. doi : arXiv:1606.05908. ³3 논문 | Goodfellow, I. et al. (2014). Generative Adversarial Nets. doi : arXiv:1406.2661. ⁴4 논문 | Goodfellow, I. (2016). NIPS 2016 Tutorial: Generative Adversarial Networks. doi : arXiv:1701.00160 ⁵5 논문 | Nowozin, S. et al. (2016). f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization. doi : arXiv:1606.00709. ⁶6 논문 | Arjovsky, M. & Chintal, S. (2017). Wassertein GAN. doi : arXiv:1701.07875. ⁷7 논문 | Metz, L. et al. (2016). Unrolled Generative Adversarial Networks. doi : arXiv:1611.02163 ⁸8 참고 | <https://www.wired.com/2017/04/googles-dueling-neural-networks-spar-get-smarter-no-humans-required/>

석/박사생을 위한 카카오의 상시 연구지원 프로그램

석/박사생을 위한 상시 연구지원 프로그램

(학비/연구비지원, 해외학회참관지원, 연구인턴십)

카카오의 인공지능(AI) 플랫폼, '카카오(아이)'는 카카오가 보유한 핵심 기술이자 미래 기술의 집약체입니다. 카카오는 AI를 통해 많은 사람들이 일상의 모든 것들을 더 편리하게 이용하고, 특별한 지식이 없어도 누구나 기술 진화의 혜택을 맛 볼 수 있는 세상을 꿈꾸고 있습니다. 카카오에서는 우수 인재들이 대학원에서 꿈을 키워나가는 과정을 지원하기 위해 몇 가지 프로그램을 마련했습니다. 학비/연구비 장학금 지원, 해외 학회 참관비용 지원, 연구 인턴십 등을 통해 더 나은 연구 환경을 제공하고, 이를 통해 언젠가는 카카오의 비전에 동참할 수 있기를 기대합니다. 우수 인재들의 많은 관심과 참여 부탁드립니다.


연구장학금 (학비/연구비 지원)

모집대상 | 석사/박사/석·박통합과정 재학생

모집분야 | AI 및 Computer Science/Engineering 관련 전 분야

모집시기 | 상시

선발절차 | 서류심사 > 온라인 인터뷰 > 오프라인 인터뷰 > 최종합격


지원방법 |  온라인 접수
<https://careers.kakao.com/jobs/P-10461>

혜택 및 의무 | (1) 학비/연구비 일체 지원(연간 최대 2,000만원까지)
(2) 수혜 기간 중 학회 참여비용 연 1회씩 지원
(3) 수혜 기간과 동일 기간 카카오 정직원 의무 근무

해외 학회 참관비용 지원

모집대상 | 석사/박사/석·박통합과정 재학생

선발절차 | 서류심사 > 오프라인 면접(1회) > 최종합격

지원방법 |  온라인 접수
<https://careers.kakao.com/jobs/P-10476>

혜택 및 의무 | (1) 왕복항공권, 숙박비, 참가비 전액 지원
(2) 종료 후 2개월 내 카카오에서 참관 내용 공유 PT 진행

연구인턴십


모집대상 | 석사/박사/석·박통합과정 재학생

모집분야 | AI 및 Computer Science/Engineering 관련 전 분야

모집시기 | 상시

근무기간 | 2~6개월 (조율 가능)

선발절차 | 서류심사 > 온라인 테스트 > 오프라인 면접(2회) > 최종합격

지원방법 |  온라인 접수
<https://careers.kakao.com/jobs/P-10459>

혜택 및 의무 | (1) 카카오 데이터 사용 가능
(2) 근무지 부근(판교 혹은 제주) 숙소 제공
(3) 인턴십 중 수행한 연구로 논문 게재 시 최대 1,000만원 인센티브 추가 지급

석/박사생을 위한
카카오 상시
연구지원 프로그램

kakao i
RESEARCH
SUPPORTING
PROGRAM

학비·연구비지원 · 해외학회참관지원 · 연구인턴십

연구장학금(학비/연구비 지원)
최소 2년간 석·박통합과정 수료시까지 지원 가능
최대 2,000만원 (석·박통합과정 수료 후 2년간 지원 가능)
지원자격: 석·박통합과정 재학생
지원기간: 2024.01.01 ~ 2024.12.31
지원방법: <https://careers.kakao.com/jobs/P-10461>

해외 학회 참관비용 지원
최대 2,000만원 (석·박통합과정 수료 후 2년간 지원 가능)
지원자격: 석·박통합과정 재학생
지원기간: 2024.01.01 ~ 2024.12.31
지원방법: <https://careers.kakao.com/jobs/P-10476>

연구인턴십
최소 2개월 이상 근무 가능
최대 6개월 (조율 가능)
지원자격: 석·박통합과정 재학생
지원기간: 2024.01.01 ~ 2024.12.31
지원방법: <https://careers.kakao.com/jobs/P-10459>

카카오는 여러분의 연구를 위한 최고의 파트너입니다.
카카오 AI 플랫폼을 통해 여러분의 연구를 위한 최고의 파트너입니다.
카카오 AI 플랫폼을 통해 여러분의 연구를 위한 최고의 파트너입니다.
카카오 AI 플랫폼을 통해 여러분의 연구를 위한 최고의 파트너입니다.

70

71

마치며

'인공지능(AI)'과 '4차 산업혁명' 이 두 단어가 빠지면 회의나 세미나가 이뤄질 수 없다." 학계에서 요즘 주로 나오는 말입니다. 그런데 회의나 세미나에서 AI를 다루는 주된 방식은 방향에 따라 극단적인 경우가 많습니다. 혹시 모를 위험에만 무게를 뒀서 AI에 대해 지나친 경계론을 펼치기도 하고 또는 AI가 가져올 편리성에만 중심을 둔 채 막연한 장밋빛 이상론을 전개합니다.

카카오 AI 리포트는 다양한 각도로 AI의 현 주소와 미래를 고찰함으로써, 좀 더 입체적인 AI 논의가 이루어지는 디딤돌의 역할을 하길 바랍니다. 2018년을 눈 앞에 두고 카카오 AI리포트 편집진은 보다 나은 내용과 구성을 선보이기 위해 속고 중입니다.

저희의 고민을 함께 나눠 주실 분은 카카오 AI리포트 편집진 이메일(kaka oaireport@kakaocorp.com) 혹은 카카오 AI리포트가 게재되는 카카오 정책지원파트 브런치(<https://brunch.co.kr/@kakao-it>)를 통해 댓글로 기탄없이 의견을 주세요. 어떤 의견이든 환영입니다. 함께 고민한 결과에 대해선 빠르면 다음 AI리포트, 늦으면 신년호를 통해 소상히 설명드리겠습니다.

기온이 낮아져, 날이 꽤 많이 곤두서 있습니다. 맨손에 스치는 바람에 베이지는 얇을까 부지불식 간에 양손을 주머니에 넣게 되는 요즘입니다. 건강 조심하세요. 다음 호로 찾아 뵈겠습니다.

