손해배상 사건에서의 인공지능(AI) 활용방안 (후속연구)

연　　구　　자 : 서울대학교 산학협력단
연 구 책 임 자 : 박 상 철(서울대학교 법학전문대학원)
공 동 연 구 원 : 신 효 필(서울대학교 언어학과)
이 상 학(서울대학교 데이터사이언스대학원)
연 구 보 조 원 : 오 채 영(서울대학교 언어학과)

제 출 문

법원행정처장 귀하

본 보고서를 귀 기관으로부터 의뢰받은 「손해배상 사건에서의 인공지능(AI) 활용방안 (후속연구)」의 최종보고서로 제출합니다.

2023. 11. 17.

서울대학교 산학협력단

목 차

그림 목차

제1장

서설

제1절 연구의 배경과 목적

서울대학교 산학협력단(연구책임자 고학수)은 2020. 12. 17. ~ 2021. 9. 30.중 수행된 "손해배상 사건에서의 인공지능(AI) 활용방안" 관련 1차 연구과제에서 손해배상 판결 예측과 관련한 다각적인 연구를 수행하였던바, 특히 교통사고 판결 과실상계 비율 예측과 관련한 이하 연구를 포함하였습니다.[1)]

- 판결문 과실상계 비율 판단에 있어 쟁점이 되는 요소를 자동으로 추출하는 시스템의 구현
- 과실상계 비율 예측 모델의 시범 구현
- 인공지능 활용을 통해 법관에게 실질적인 조력이 이루어질 수 있는지 여부의 검증
- 사건예측결과 및 그 원인에 대한 설명을 시각화하여 제시하는 기능에 대한 시범 구현

그러나 위 1차 연구과제 완료 이후 약 2년간 GPT, BERT 등 Transformer 기반 사전학습 언어모델(pretrained language model; PLM)이 급격히 발달하면서, 이를 반영한 모델로 재검증을 해 볼 필요성이 생겼습니다. 1차 연구과제 수행 과정에서 SK 텔레콤 주식회사가 한국어 코퍼스로 사전훈련한 Korean BERT pre-trained cased (KoBERT)[2)]를 미세조정(fine-tuning)하여 과실상계비율 예측모델을 만들었으나 예측정확도가 기대에는 미치지 못하였습니다(보행자 사고 관련 판결에 대하여 3분위 과실상계비율 예측정확도 69.9%). 이로 인해 대안으로 법관의 과실상계 가감의 판단에 영향을 미칠만한 특성값(feature)을 수동으로 레이블링(labeling)하고 이를 토대로 지도학습 모델을 만들어 3분위 과실상계비율 관련 최대 80.8%의 예측정확도를 달성하였습니다.[3)] 그러나 이처럼 레이블링을 토대로 지도학습(supervised learning)하는 방식은 어텐션(attention) 등 비지도학습(unsupervised learning) 기법들을 중심으

1) 서울대학교 산학협력단(고학수 외), "손해배상 사건에서의 인공지능(AI) 활용방안" (2021).
2) SK텔레콤, Korean BERT pre-trained cased (KoBERT), https://github.com/SKTBrain/KoBERT.
3) 서울대학교 산학협력단, 전게 보고서.

로 하는 최근의 인공지능의 발전방향과는 다른 접근이고, 레이블링에 많은 연구인력과 시간이 소요되어 계속적으로 작성될 다량의 판결문으로 모델을 지속적으로 훈련하고 업데이트하기에 적합한 방식이 아니라는 아쉬움이 있었습니다. 이와 같이 레이블링을 병행하는 과정에서 제공받은 판결 중 보행자 사고 관련 판결만을 분석했다는 점도 한계였습니다.

따라서 2023. 5. 18. ~ 11. 17.간 수행(실질적으로는 판결문 데이터를 입수 완료한 2023. 6. 29.부터 수행)된 이번 후속연구에서는 1차 연구과제 완료 이후 자연어처리(NLP) 기법의 발전을 반영하여 레이블링 등 인적 노력을 요하지 않아 효율성을 비약적으로 높이면서도 우수하거나 비견되는 성능의 과실상계비율 예측모델을 만들 수 있는지 여부를 재차 개념검증(Proof of Concept; POC)하고자 하였고, 이러한 연구를 통해 재산분할, 유류분, 사해행위취소, 회생사건 등 다른 민사사건에서의 예측모델의 가능성도 타진해 보고자 하였습니다.

다만, 이번 연구에서는 1차 연구에 비해 5분의 1의 예산, 상대적으로 소규모의 인력, 단기간을 투입하여 지난 2년간의 기술 발전이 교통사고 판결의 과실상계 비율 예측에 있어 효율성과 성능의 향상을 가져오기에 충분했는지 여부에 관한 실험적 연구를 하는데 집중하였습니다. 따라서 1차 연구와 달리 인공지능의 기본 개념, 판결 예측을 위한 인공지능 기술, 사법예측 기술 등을 재검토하기보다는, 귀원이 1차 연구를 위하여 제공하셨던 2015년~2020년간 자동차 사고 1심 판결문(가명처리)을 그대로 다시 제공받아 신규 기술을 적용하고 성능을 재평가하는 실증연구에 검토 범위를 한정하였습니다.

제2절 연구의 방법

이번 연구기간 중 지난 2년간의 기술 발전을 반영하여 1차 연구 때는 시도하지 못하였던 다음 두 가지 방식으로 연구를 수행하였습니다.

Ⅰ. RoBERTa 모델(KLUE-RoBERTa-Large)의 도메인 적응 및 미세조정을 통한 예측

한국어 사전학습 언어모델인 KLUE-RoBERTa-Large 모델[4]에 2015년~2020년간 자동차 사고 1심 판결문 등 다수의 법률 코퍼스를 추가 투입하여 도메인 적응적 사전훈련(domain adaptive pretraining)을 하였습니다. 이를 통해 특화된 모델을 학습시킨 후 과실상계 예측이라는 하류작업(downstream task)에 맞추어 미세조정(fine-tuning)하는 방식으로 어느 정도의 성능향상이 있는지를 측정해 보았습니다.

특히 법률 코퍼스의 추가 투입을 통한 도메인 적응을 통해 비법률 코퍼스로 훈련된 모델의 법률적 예측 능력을 제고하고자 하였습니다. 이는 트랜스포머 기반 자연어처리에 있어 현재의 정석(定石)적인 방식이라고 할 수 있습니다.

Ⅱ. 도메인 적응된 RoBERTa 모델의 임베딩과 코사인 유사도를 활용한 예측 및 애플리케이션의 가능성 검토

2015년~2020년간 자동차 사고 1심 판결문 중 과실상계 판단 부분에 현출된 사실관계를 위와 같이 도메인 적응된 RoBERTa 모델의 임베딩(embedding)에 배치하여 벡터 공간에서의 위치를 구한 후, 이를 근거로 각각의 판결문과 다른 판결문의 과실상계 판

4) Sungjoon et al., KLUE: Korean Language Understanding Evaluation, arXiv:2105.09680v4 (2021).

단 부분의 위치와의 코사인 유사도(cosine similarity)를 계산하여 유사 판결을 검색하고 상위 검색 판결들의 과실상계 비율을 평균(연속값의 경우)하거나 다수결(삼분위 범주값의 경우)하는 방식으로 과실상계 비율을 예측한 후 성과를 측정해 보았습니다.

나아가, 성과가 유의미하다는 전제 하에서, 법관이 입력할 프롬프트(자신의 사건의 사실관계)의 각각의 판결문들의 과실상계 판단 부분의 위치와의 코사인 유사도를 계산하여 상위 검색 판결들과 유사도를 보여주고 이를 통해 법관의 판단에 도움이 될 수 있는 애플리케이션의 개발 가능성을 검토해 보았습니다.

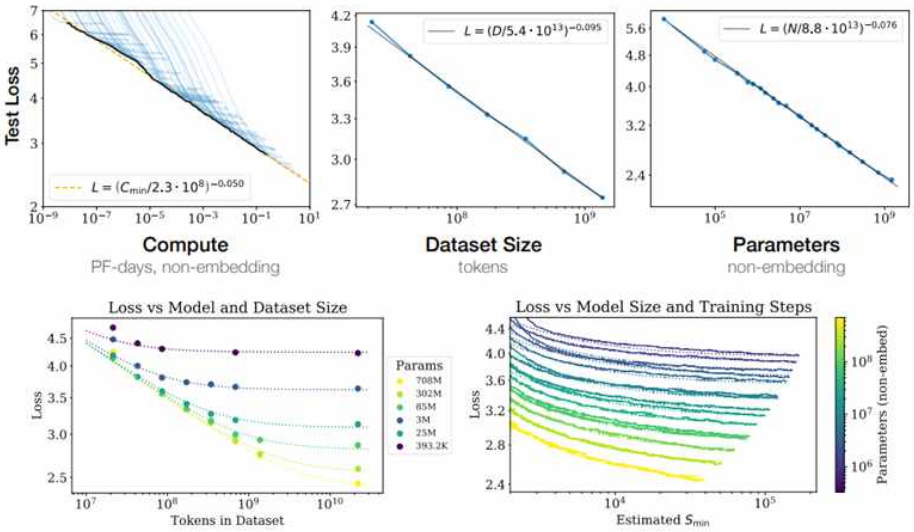
제2장

1차 연구 이후 2년여간 자연어처리 기술의 발전과 본 후속연구에 대한 시사점

제1절 언어모델의 발전

1차 연구가 종결된 2021. 9. 이후 가장 괄목할 만한 발전은 역시 ChatGPT의 2022. 11. 30.자 출시에 이은 2023년 동안의 GPT-3, 4 모델의 대중화라고 할 수 있습니다. 트랜스포머(Transformer) 기술은 이미 2017년에 개발되었으나,[5] 이에 기한 GPT-3, 4 모델의 개발은 다음과 같은 중요한 의미들이 있었습니다.

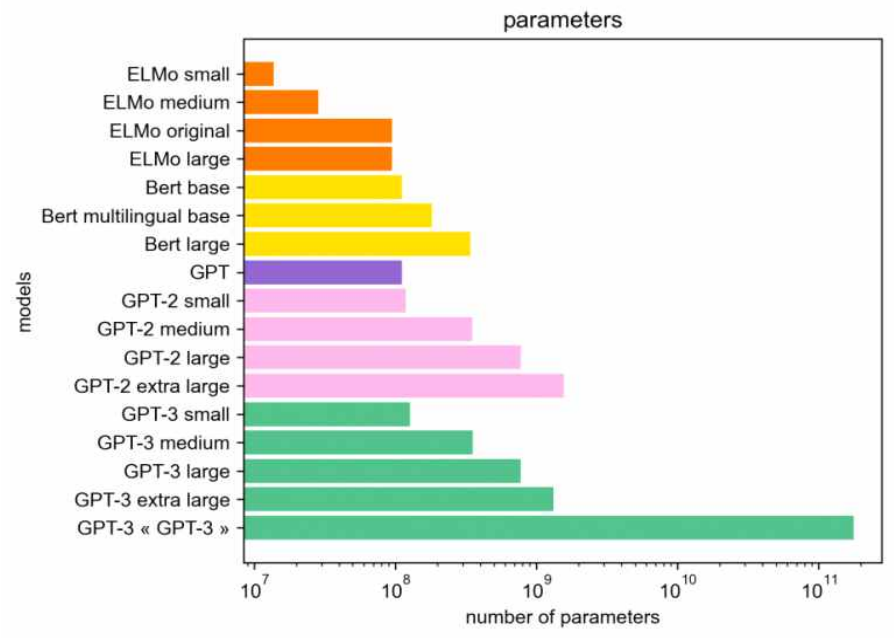
첫째, GPT-3부터 모델의 크기(파라미터의 수)와 투입되는 데이터셋의 규모(토큰 수)가 비약적으로 늘어나며 대규모 언어모델(large language model; LLM)의 시대를 열었습니다. 특히 OpenAI 팀은 2020년 논문을 통해 모델의 크기, 데이터셋의 크기, 훈련에 소요된 연산량이 언어모델의 성능(cross-entropy loss)을 좌우하며, 이 중에서도 특히 모델의 크기가 중요하다고 주장하였습니다.[6]



[그림 1] 연산량, 데이터셋 크기, 모델 크기에 따른 언어모델의 성능 증대[7]

5) Ashish Vaswani et al., Attention is All You Need, NIPS 2017 (2017).
6) Jared Kaplan et al., Scaling Laws for Neural Language Models, arXiv: 2001.08361 (2020).
7) *Id.*

OpenAI는 이에 따라 특히 모델의 크기(파라미터 수) 뿐 아니라 데이터셋의 크기(토큰 수)를 비약적으로 증대시킨 GPT-3를 개발하여 출시하게 됩니다.



[그림 2] GPT-3와 기존 모델의 파라미터수[8]

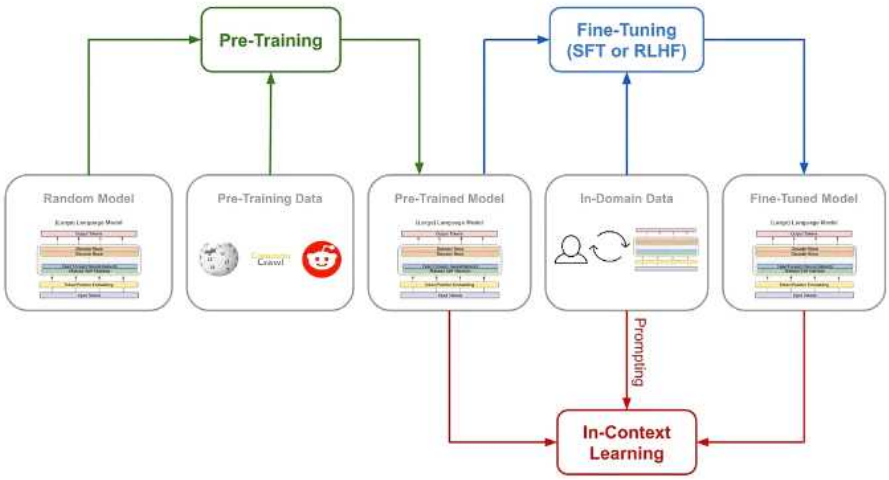
175B의 파라미터수를 가진 GPT-3는 280B의 파라미터수를 가진 Gopher, 530B의 파라미터수를 가진 Megatron 등 언어모델들의 크기 경쟁을 촉발하였습니다.

다만, 이와 같이 크기가 전부라는 믿음은 최근 조금씩 흔들리고 있습니다. DeepMind는 위 OpenAI 논문을 비판하면서, GPT-3, Gopher, Megatron이 심각하게 과소훈련(undertrain)되어 있음을 지적하였고, 모델 크기 뿐 아니라 데이터셋 크기와 연산량도 못잖게 중요하다는 사실을 규명합니다.[9] DeepMind는 이에 따라 모델 크기를 70B로 줄이고, 토큰 수를 1.4조로 늘린 후, 좀 더 오래 훈련하여, 위 초대규모 모델에 못잖은 성능을 달성하는 친칠라(Chinchilla)라는 언어모델을 출시하여, 소규모 언어모델(small language model; SLM)의 포문을 엽니다. 메타(Meta) 또한 DeepMind의 위 논문에 근거하여 언어모델을 경량화/효율화하였고, 이에 따라 7~65B의 파라미터,

8) Orange, The GPT-3 Language Model, Revolution or Evolution?, https://hellofuture.orange.com/.
9) Jordan Hoffmann et al., Training Compute-Optimal Large Language Models, arXiv:2203.15556v1 (2022).

즉 GPT-3의 1/10 정도의 파라미터를 사용한 LLaMA-13B로 GPT-3의 성능지표를 앞서는 결과를 발표하였습니다.[10] 급기야는 스탠포드대 연구팀은 GPT-3.5에 일정한 프롬프트에 대한 출력들을 생성하게 하고 이를 통해 LLaMA-7B를 미세조정하여 단 600불 이내의 예산으로 랩탑에 저장할 수 있는 크기의 소규모 언어모델(Alpaca)을 개발하기에 이릅니다.[11]

이러한 도전에도 불구하고, 특히 GPT-4는 압도적인 모델과 데이터셋의 규모로 GPT-3의 수준을 또 한 번 뛰어넘는 전례 없는 성능을 보여주고 있습니다. 특히 대규모 언어모델의 경우 언어모델의 크기가 커지면서 일정한 임계치를 넘으면, 소규모 언어모델에 없는 창발적 능력(emergent abilities)을 획득하여 비약적으로 성능이 향상된다는 점까지 지적되어,[12] 대규모 언어모델의 시대는 계속되리라는 믿음을 강화하고 있습니다.



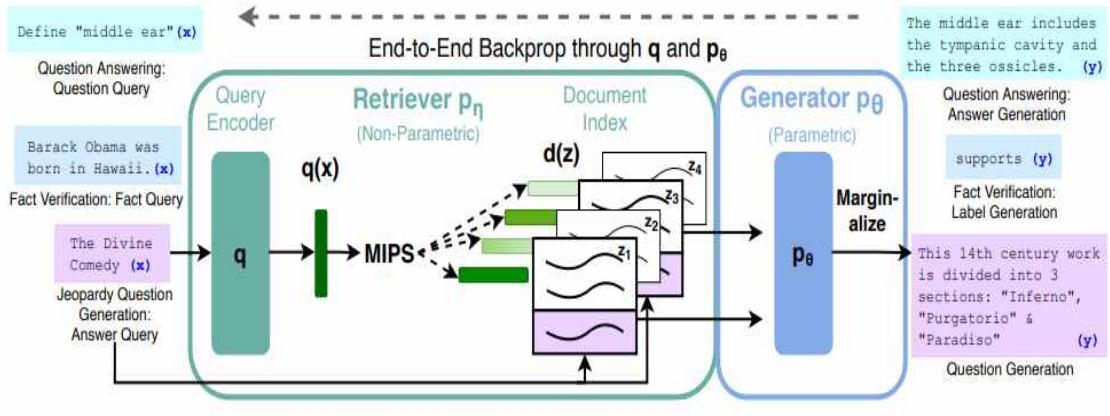
[그림 3] 미세조정과 문맥 내 학습[13]

둘째, 기존 언어모델들은 특정 하류작업(downstream task)에 적응하기 위해 미세조정(fine-tuning) 기법에 의존하였으나, GPT-3 이상의 모델들은 개발자들이 의도

10) Higo Touvron et al., LLaMA: Open and Efficient Foundation Language Models, arXiv:2302.13971 (2023).
11) Rohan Taori et al., Stanford Alpaca: An Instruction-following LLaMA Model, https://github.com/tatsu-lab/stanford_alpaca (2023).
12) Jason Wei et al., Emergent Abilities of Large Language Models, Transactions on Machine Learning Research (2022).
13) Bijit Ghosh, Empowering Language Models: Pre-training, Fine-tuning, and In-Context Learning, Medium (2023).

하지 않았던 또 다른 창발적 능력으로서 프롬프트(prompt)를 통해 특정 하류작업에 적응하는 문맥 내 학습(in-context learning) 능력을 보였고, 이에 따라 프롬프트 엔지니어링(prompt engineering) 기법들이 발전하고 있습니다.

미세조정이나 문맥 내 학습 외 하류작업을 위한 다양한 기법들도 계발되고 있습니다. 특히 주목받는 기법은 미세조정 등을 통해 모델의 파라미터들의 전부 또는 일부를 조정하는 대신, 기존에 훈련된 모델은 그대로 두되 모델 외부 문서를 검색하여 그 내용을 생성하면서, 생성된 정보의 출처(source)를 달아주어, 언어모델 특유의 환각(hallucination; 생성과 사실의 불일치) 문제도 해결하는 검색증강생성(retrieval-augmented generation; RAG) 기법입니다.[14]



[그림 4] 검색증강생성 기법의 개요[15]

셋째, OpenAI는 ChatGPT의 개발 과정에서 비단 수동적인 필터링이나 이에 기한 지도학습에 그치지 아니하고 사람의 피드백에 의한 강화학습(reinforcement learning with human feedback; RLHF) 기법을 개발하여 법적, 사회적으로 민감한 내용들을 효율적이고 광범위하게 필터링할 수 있었습니다.[16] 이는 법적 책임 등의 우려를 어느 정도 낮추고 ChatGPT의 대중화를 촉진하는 결정적인 계기가 되었습니다.

14) Patrick Lewis et al., Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks, NeurIPS 2020 (2020).
15) *Id.*
16) Long Ouyang et al., Training Language Models to Follow Instructions with Human Feedback, arXiv:2203.02155 (2022).

제2절 사법예측 모델에 미칠 영향

이와 같은 획기적인 발전 이전에 전 세계적으로 사법예측 모델이 빠르게 발전해 왔다고는 할 수는 없습니다. 미국 다수 주들의 재범예측모델(COMPAS)의 공판전 조사(pre-trial investigation) 및 양형 등을 위한 활용, 중국의 "유사사건 유사판결(类案类判)" 원칙 준수를 위한 AI 기반 검색시스템의 활용, 호주의 친권, 재산배분 등 제안서비스, 싱가폴, 프랑스 등에서의 경미사건을 중심으로 한 판결 자동화의 연구 사례들이 있으나 예측 모델이 사법절차에 활발히 활용되거나 깊이 있게 연구되지는 아니하였던 것이 사실입니다.[17]

나아가 프랑스처럼 법관과 사법부 구성원의 신원 데이터를 직업적 실행을 평가, 분석, 비교, 예측할 목적이나 효과를 위해 재사용하는 것을 형사처벌(art L10, Code de justice administrative)하는 경우도 있고, EU AI법안(AI Act)이 예정대로 곧 통과될 경우 일정한 유예기간(현재 안 기준 2년)을 거쳐 범죄/재범예측 AI를 금지하고 사법, 거짓말탐지기, 증거 평가, 수사/기소를 위한 프로파일링, 범죄분석을 위한 AI의 활용을 광범위하게 제약할 예정인 등 사법예측 모델에 대해 법적 제한이 가해질 수 있습니다.

다만 GPT-4는 법률 분야 중 일부 제한된 분야에서 사람보다 높은 성능을 보이면서 일종의 법적 튜링테스트(legal Turing test)를 통과하고 있습니다. 이하 표는 GPT-4가 미국 변호사시험(bar exam) 상위 10%, 로스쿨 입학시험(LSAT) 상위 12%를 달성하였음을 보여줍니다.

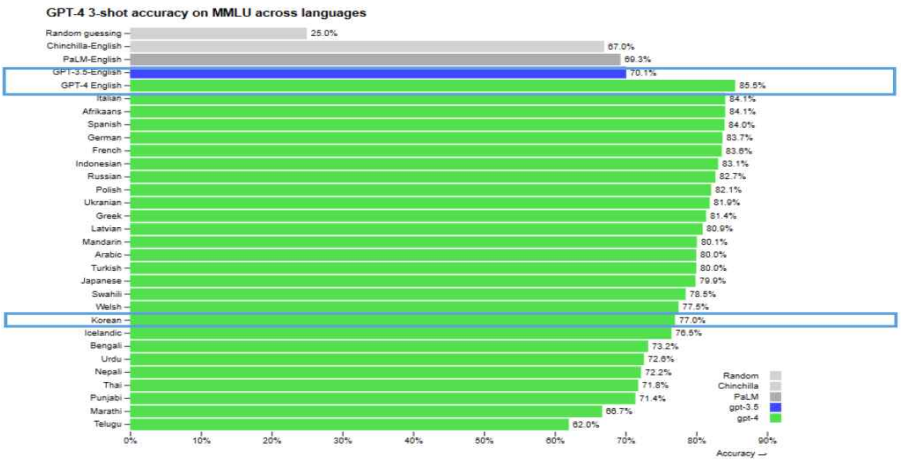
17) 상세한 사항은 서울대학교 산학협력단, 위 보고서.

Simulated exams	GPT-4 estimated percentile	GPT-4 (no vision) estimated percentile	GPT-3.5 estimated percentile
Uniform Bar Exam (MBE+MEE+MPT)[1]	298/400 ~90th	298/400 ~90th	213/400 ~10th
LSAT	163 ~88th	161 ~83rd	149 ~40th
SAT Evidence-Based Reading & Writing	710/800 ~93rd	710/800 ~93rd	670/800 ~87th
SAT Math	700/800 ~89th	690/800 ~89th	590/800 ~70th
Graduate Record Examination (GRE) Quantitative	163/170 ~80th	157/170 ~62nd	147/170 ~25th
Graduate Record Examination (GRE) Verbal	169/170 ~99th	165/170 ~96th	154/170 ~63rd
Graduate Record Examination (GRE) Writing	4/6 ~54th	4/6 ~54th	4/6 ~54th
USABO Semifinal Exam 2020	87/150 99th–100th	87/150 99th–100th	43/150 31st–33rd
USNCO Local Section Exam 2022	36/60	38/60	24/60
Medical Knowledge Self-Assessment Program	75%	75%	53%
Codeforces Rating	392 below 5th	392 below 5th	260 below 5th
AP Art History	5 86th–100th	5 86th–100th	5 86th–100th
AP Biology	5 85th–100th	5 85th–100th	4 62nd–85th
AP Calculus BC	4 43rd–59th	4 43rd–59th	1 0th–7th

[그림 5] GPT-4의 시험 성적 평가 결과[18]

이러한 성과는 GPT-4를 특별히 법률 코퍼스에 맞게 도메인 적응(domain adaptation)하지 않고 얻은 성과라는 점에서 특히 주목할 만합니다.

덧붙여 특기할 만한 사실은 다양한 방식의 질의응답(question answering) 과업들에 대해 GPT-4의 한국어 성능이 GPT-3.5의 영어 성능을 상회하고 있다는 점입니다.



[그림 6] GPT-4의 다국어 처리 정확도 비교[19]

GPT-4가 이처럼 법률문서 및 다국어 처리에 대해 전례 없는 성능을 보이면서 한국어 법률문서들의 분석과 예측에 대한 기대도 커지고 있습니다.

18) OpenAI, Research: GPT-4, https://openai.com/research/gpt-4 (2023).
19) *Id.*

제3절 본 후속연구에 대한 시사점과 한계

다만, 최근의 언어모델의 발전을 이끄는 GPT-4는 폐쇄모델로서, 귀원이 제공하신 미공개 판결문을 모델의 미세조정(fine-tuning)이나 RAG 기법 적용을 위하여, 또는 임베딩의 위치를 얻기 위해 GPT-4 API에 투입할 경우 OpenAI 소유의 해외 서버에 미공개 판결문이 전송되는 결과가 초래됩니다. 이러한 방식의 연구는 귀원에 대한 기밀유지 의무상 불가능했기 때문에, 효율적이고 상당한 성능향상이 기대됨에도 이를 시도할 수는 없었습니다.

따라서 본 후속연구에서는 개방모델인 RoBERTa 모델을 연구수행기관(서울대학교)의 시스템에 올린 후 귀원으로부터 제공받은 판결문 데이터 전체를 투입하여 도메인 적응을 함으로써 법률 특화 모델을 만들고 이를 기반으로 예측모델을 미세조정하는 기법을 우선 시도하였습니다. 나아가, 위와 같이 데이터를 추가 투입하여 훈련한 RoBERTa 모델의 임베딩 상에서 과실상계비율 산정의 근거가 되는 사실관계 설시 부분과 코사인 유사도가 높은 판결들을 검색해서 해당 판결의 과실상계비율들을 평균하거나 다수결로 집계하여 예측하는 방식을 활용함으로써, 법관의 판단이 도움이 되는 애플리케이션의 실마리를 찾고자 하였습니다.

다만, 이러한 과정에서 GPT-4 등 첨단(state-of-the-art)의 언어모델 등 민간의 다양한 개발역량을 활용, 접목하기 위해 향후 판결문 공개 등 정책이 병행될 필요가 있다는 어려운 고민을 남기게 되었습니다.

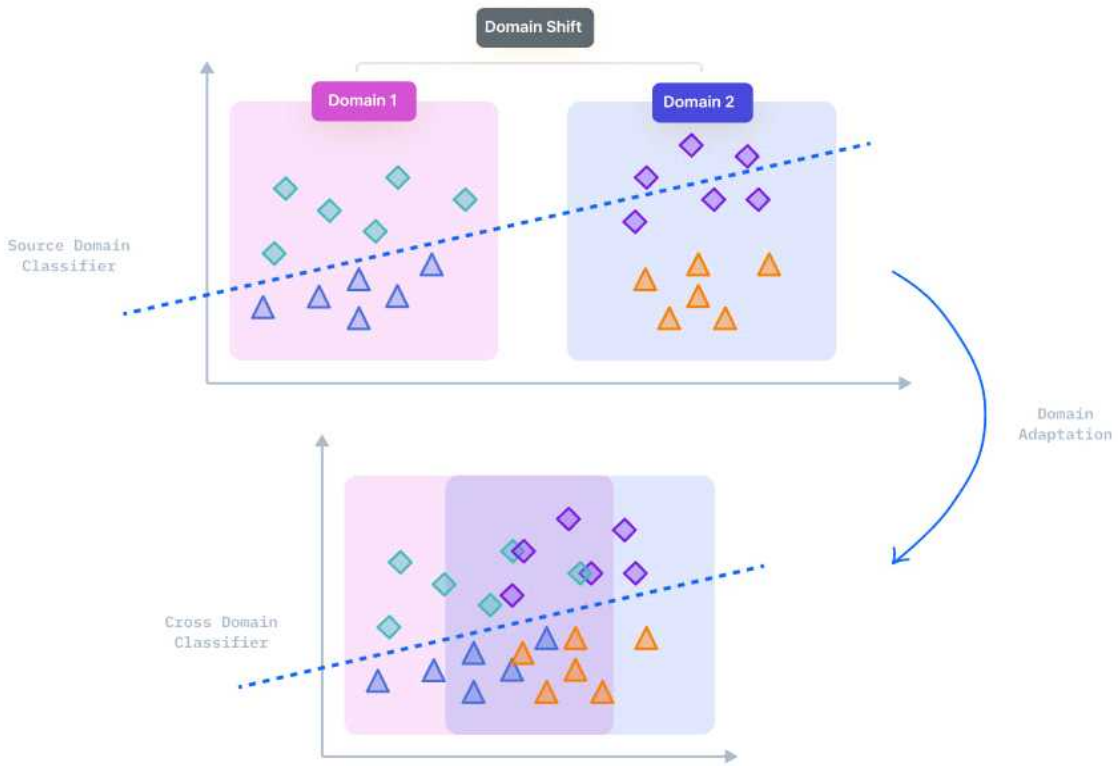
제3장

RoBERTa 모델의 도메인 적응 및 미세조정을 통한 예측

제1절 도메인 적응적 사전훈련(Domain Adaptive Pretraining)

Ⅰ. 개요

도메인 적응(domain adaptation)이란 기계학습 모델이 훈련된 기존 영역(source domain)과 다르지만 관련 있는 새로운 영역(target domain)에서의 예측을 위하여 사용될 때, 기존 영역의 정보를 새로운 영역에 적응(adapt)시켜 사용함으로써 새로운 환경에 대한 견고성(robustness)을 높이는 기법입니다. 본 과제에서는 특히 일반적인 한국어 코퍼스로 훈련된 언어모델에 법률 코퍼스를 신규 투입하여 법률문제의 예측에 특화된 모델을 만드는 수행하였습니다.



[그림 7] 도메인 적응의 개념도[20]

20) Rohit Kundu, Domain Adaptation in Computer Vision: Everything You Need to Know, https://www.v7labs.com/blog/domain-adaptation-guide (2022).

Ⅱ. 사전학습 언어모델

허깅페이스(Hugging Face)에서 klue/roberta-large 체크포인트[21]를 불러와 법률 도메인 코퍼스로 MLM (Masked Language Model) 학습을 추가적으로 진행하였습니다. KLUE-RoBERTa는 업스테이지, 네이버 AI랩스, KAIST 등의 연구자들이 공동 개발하였으며 총 1,024개의 임베딩 사이즈, 1,024개의 은닉 사이즈, 24개 레이어, 16개 헤드로 구성된 모델입니다.[22]

Ⅲ. 데이터

약 1.24GB의 레이블되어 있지 않은(unlabeled) 법률 도메인 텍스트 파일을 KLUE-RoBERTa의 추가 훈련에 투입하였습니다. 총 2,852,073 행의 텍스트가 투입된바, 그 종류는 이하와 같습니다.

1. 귀원으로부터 제공받은 자동차 사고 관련 판결문 (2015년~2020년)

본 연구진은 귀원으로부터 2015. 1. 1. ~ 2020. 12. 31. 기간 중 선고된 전국 지방법원 1심 민사 판결문 중 ① 사건명이 “손해배상(자)”인 판결문 전부와 ② 사건명이 “구상금”, “보험금”, 또는 “채무부존재확인”(“구상금등”, “보험금등”, “보험금지급채무부존재확인” 등과 같이 이들 단어가 포함되거나 조합된 사건명 포함)이면서 “차량” & “사고”라는 키워드로 검색되는 판결문 중 원고 일부 승소 판결문을 가명처리한 한글 파일(.hwp)로 제공받았고,[23] 그 구체적인 내역은 다음과 같습니다.

- 판결문(손해배상(자), 원고일부승): 4,336건
- 판결문(구상금,차량&사고,원고일부승): 1,652건

21) https://huggingface.co/klue/roberta-large.
22) Sungjoon et al., KLUE: Korean Language Understanding Evaluation, arXiv:2105.09680v4 (2021).
23) 서울대학교 산학협력단, 전게 보고서.

- 판결문(보험금,차량&사고,원고일부승): 243건
- 판결문(채무부존재확인,차량&사고,원고일부승): 659건

2. 손해보험협회 자동차사고 과실비율분쟁 심의사례집 (2020)[24]

손해보험사들이 과실비율 분쟁의 자율적인 해결을 위해 보험업법에 근거한 상호협정으로 인가받아 운영하는 "자동차사고 과실비율분쟁 심의위원회"의 심의사례를 모아놓은 자료입니다.

3. 엘박스 판결문 데이터[25]

엘박스가 허깅페이스에 공개한 총 301,328건의 판결문 데이터셋입니다.

4. 도로교통법 전문

법제처에서 입수한 도로교통법 전문입니다.

5. 찾기 쉬운 생활법령정보 백문백답[26]

법제처의 Q&A 데이터를 모아 정지우 씨가 허깅페이스에 올려놓은 코퍼스입니다.

6. LegalQA데이터[27]

프리로펌의 법률질의응답 데이터를 전희원 씨가 Github에 올려놓은 코퍼스입니다.

전체 데이터 중 텍스트 파일로 변환 중 오류가 발생한 데이터 3건은 제외되었습니다. 전처리는 한글, 영어 소문자 및 대문자, 숫자, 공백, 기본적인 문장부호만을 포함하도

24) 손해보험협회, "자동차사고 과실비율분쟁 심의 사례" (2020. 10.), https://accident.knia.or.kr/research-content?index=87957.
25) https://huggingface.co/datasets/lbox/lbox_open.
26) https://huggingface.co/datasets/jiwoochris/easylaw_kr.
27) https://github.com/haven-jeon/LegalQA.

록 정규표현식(regex) [^가-힣a-zA-Z0-9\s~!@#$%^&*()-.,_+|<>?:{}]+을 통해 처리하였고, line별로 split() 함수를 적용하여 6어절 이상인 경우만을 포함하였습니다. 또한 line의 총 음절이 510을 초과하는 경우 분할하였습니다.

Ⅳ. 학습

허깅페이스에서 제공되는 MLM 학습 코드[28]를 활용하였습니다.

- 학습 하이퍼파라미터: epoch=3, batch size=32, gradient accumulation steps=1
- 사용한 GPU: A100 80GB * 3
- 총 학습 소요 시간: 약 95시간

이로써 법률 분야에 특화될 뿐 아니라 귀원이 제공한 판결문들을 통해 추가 훈련된 KLUE-RoBERTa 모델을 얻을 수 있었습니다.

```
epoch 0: perplexity: 1.92165443533486 eval_loss: 0.6531865000724792
epoch 1: perplexity: 1.7952154257546065 eval_loss: 0.5851250290870667
epoch 2: perplexity: 1.712825710978678 eval_loss: 0.5381444692611694
```

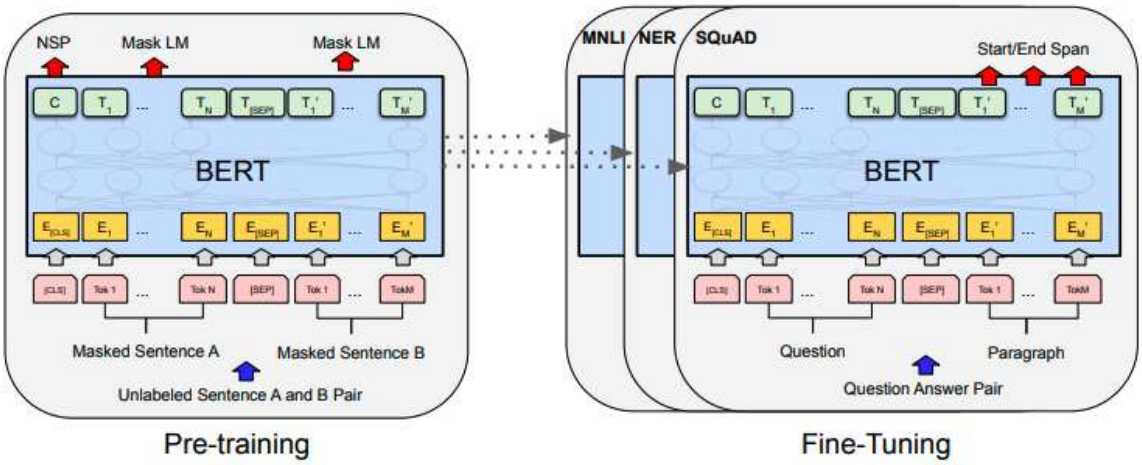
언어모델의 당황도(perplexity)란 새로운 토큰을 생성할 때의 불확실성의 정도, 즉 모델이 언어의 문법 패턴을 잘못 학습하여 새로운 예시에 당황하는 정도를 의미하며 0부터 무한대의 값을 가집니다. 평가손실(evaluation loss)은 시험 단계에서 측정한 모델 성과를 의미하는데 일반적으로 0.1과 1.0 사이는 우수한 성과로 평가됩니다. 따라서 이상의 지표는 모델이 상당히 안정적으로 훈련되었음을 시사합니다.

28) https://github.com/huggingface/transformers/blob/main/examples/pytorch/language-modeling/run_mlm_no_trainer.py

제2절 과실상계 비율 예측을 위한 미세조정

Ⅰ. 개요

미세조정(fine-tuning)이란 사전학습 언어모델을 특정 하류작업(downstream task)에 맞게 조정하는 작업을 의미합니다. 사전학습한 모든 파라미터와 더불어 하류작업을 위한 최소한의 파라미터를 추가해서 모델을 추가로 학습하게 됩니다.



[그림 8] BERT 모델의 미세조정의 개념도[29]

Ⅱ. 데이터

입력값(*X*)으로 투입할 판결문 중 과실상계 판단 부분의 자동추출은 1차 연구 때와 동일한 방식으로 수행하였으므로 1차 연구보고서를 참조하시기 바랍니다. 간략히 다시 설명하면, 판결문의 본문 중 어떠한 단락이 "책임.*제한|과실.*상계|원고.*과실"이라는 정규표현식을 최초로 충족할 경우 이 단락을 과실상계 판단 부분으로 보고, 이를 판결문 특유의 목차의 구조(1, 2, 3, … / 가. 나. 다., … / (1), (2), (3))를 활용하여

29) Jacob Devlin et al., BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, arXiv:1810.04805v2 (2018).

추출해 냈습니다.[30] 다만, 이 때 예컨대 “3. 손해배상 책임의 발생과 과실 상계” 등 목차로 시작하는 단락의 경우 “가. 손해배상 책임의 발생”, “나. 과실 상계” 등의 하위 단락을 포함하는데, 이 중 필요한 것은 가. 나. 전부가 아니라 나. 뿐이므로, 3항 전부가 추출되지 않도록 할 필요가 있으므로, 위 정규표현식을 충족하는 경우에도, "책임.*근거|책임.*인정|책임.*발생|면책|인정사실"이라는 정규표현식을 충족하는 단락의 경우 추출되지 않도록 함으로써, 뒤에 오는 “나. 과실 상계”로 시작되는 하위 단락만 추출되도록 하였습니다.[31]

레이블(y)로 투입할 과실상계비율을 자동으로 추출하면서 동시에 이러한 과실상계비율이 추출되지 않는 판결문을 분석에서 제외하는 절차는 다음 순서로 진행 하였습니다:[32]

- “피재자”나 “피용자”가 포함되어 있을 경우 산업재해 관련 사건이므로 제외
- “주장한다”, “주장은 이유 없다”, “주장은 받아들일 수 없다”, “증거가 없다”로 끝나는 문장에 적시된 비율은 판시된 비율이 아닌 당사자가 주장한 비율이므로 제외
- "책임.*?\d*%.*?로 제한", "책임.*?\d*%.*?로 판단", "손해액.*?\d*%.*?로 제한", "손해의.*?\d*%를 배상할 책임", "책임\s?비율.*?\d*%"의 정규표현식을 이용하여 피고 책임비율을 추출한 후, 100으로부터 피고 책임비율을 빼서 원고 과실비율을 산정.
- 위 방식으로 추출되지 않을 경우, "과실\s?비율.*?\d*%", "감액\s?비율.*?\d*%", "과실상계.*?비율.*?\d*%"의 정규표현식을 이용하여 원고 과실비율을 곧바로 추출.

단, 과실상계비율을 연속값으로 투입하지 않고 삼분위(tertile)([0, 1/3] 구간은 0, (1/3, 2/3] 구간은 1, (2/3, 1] 구간은 2로 범주화)를 레이블(label)로 하여 분류 태스크를 수행하였습니다.

위와 같은 방식으로 추출한 전체 사고 판결문 총 1,944건과 이 중 보행자 사고 판결문 총 725건 두 종류의 데이터셋으로 각각 실험을 진행 하였습니다. 원래 기존 데이터는

30) 서울대학교 산학협력단, 전게 보고서.
31) 전게 보고서.
32) 전게 보고서.

각각 1,948건과 727건이었으나, 사건번호는 같되 판시내용이 동일한 데이터들이 일부 존재하여 실험에서는 중복되는 두 데이터 중 한 가지만 포함하였습니다. 보행자 사고 뿐 아니라 차량, 오토바이, 자전거, 기타 사고 판결을 포함한 1,948건을 모두 분석한 것이 지난 1차 연구와의 큰 차이점입니다. 두 종류의 데이터를 각각 60:20:20의 비율로 훈련:검증:시험 데이터로 랜덤하게 분할하였는데, 구체적으로 전체 사고 판결문은 1,166건:389건:389건, 보행자 사고 판결문은 435건:145건:145건으로 분할하였습니다.

Ⅲ. 실험

다음 코드를 기본으로 하여 실험을 진행하였습니다. 모든 실험에 공통적으로 epoch는 10, batch size는 32, weight decay는 1e−5, learning rate는 5e−5으로 설정하였습니다. 또한 훈련을 진행하며 valid loss가 이전 valid loss보다 클 경우 early_stop_loss 리스트에 담는데, 이 길이가 4에 도달하면 학습을 멈추었습니다. Optimizer로 AdamW를, learning rate scheduler로 ExponentialLR을 사용하였습니다.

```
import os
import time
import random
import datetime
import numpy as np
import pandas as pd
from copy import deepcopy
from argparse import Namespace

import torch
import torch.nn as nn
import torch.nn.functional as F
from torch.optim import AdamW
from torch.optim.lr_scheduler import ExponentialLR
from torch.utils.data import Dataset, DataLoader
from transformers import AutoTokenizer, AutoModel
```

```
def read_file(infile):
    data = pd.read_csv(infile)
    label_lst = list(data['ratio'])
    query_lst = [query.strip() for query in list(data['judge'])]

    return {'judge':query_lst,
            'ratio':label_lst}

class LawDataset(Dataset):

    def __init__(self,infile):

        super(LawDataset,self).__init__()

        corpus = read_file(infile)
        self.query_lst = corpus['judge']
        self.label_lst = corpus['ratio']

        self.label_id = {topic: idx for idx, topic in enumerate
(list(set(self.label_lst)))}

    def __len__(self):
        assert len(self.query_lst) == len(self.label_lst)
        return len(self.query_lst)

    def __getitem__(self, index):
        query = self.query_lst[index]
        label = self.label_id[self.label_lst[index]]

        return {'query': query,
                'label': label}

class SampleCollate():
```

```
    def __init__(self,
                 tokenizer,
                 args):

        self.tokenizer = tokenizer
        self.padding = args.padding
        self.max_length = args.max_length
        self.truncation = args.truncation

    def __call__(self, samples):

        query_lst, label_lst = [], []

        for sample in samples:
           query_lst.append(sample['query'])
           label_lst.append(sample['label'])

        query_encode = self.tokenizer(
            query_lst,
            padding = self.padding,
            truncation = self.truncation,
            max_length = self.max_length
            )

        batch = {
            'input_ids':torch.tensor(query_encode['input_ids']),

'attention_mask':torch.tensor(query_encode['attention_mask']),

'token_type_ids':torch.tensor(query_encode['token_type_ids']),
            'label': torch.tensor(label_lst)
        }
        return batch
class LawModel(nn.Module):
```

```
    def __init__(self,
                 num_labels,
                 len_tokenizer,
                 args):

        super(LawModel,self).__init__()

        self.num_labels = num_labels
        self.len_tokenizer = len_tokenizer
        self.dropout = args.dropout
        self.hidden_size = args.hidden_size

        self.mod = AutoModel.from_pretrained(args.model)
        self.mod.resize_token_embeddings(self.len_tokenizer)

        self.bn = nn.BatchNorm1d(self.hidden_size)
        self.dropout = nn.Dropout(self.dropout)
        self.fc = nn.Linear(self.hidden_size, self.hidden_size)
        self.gelu = nn.GELU()
        self.output = nn.Linear(self.hidden_size, self.num_labels)

    def  forward(self,  input_ids,  attention_mask,  token_type_ids,
return_embedding=False):
        outputs   =   self.mod(input_ids=input_ids,   attention_mask=
attention_mask, token_type_ids=token_type_ids)
        last_hidden_state = outputs[0]

        cls = last_hidden_state[:,0,:]

        if return_embedding: return cls

        cls = self.bn(cls)
        cls = self.dropout(cls)
        cls = self.fc(cls)
```

```
        cls = self.gelu(cls)

        logits = self.output(cls)

        return logits

def create_adamw_optimizer(model, args):

    if args.no_decay:
        no_decay = args.no_decay
        optimizer_grouped_parameters = [
            {'params': [p for n, p in model.named_parameters() if not
any(nd in n for nd in no_decay)], 'weight_decay': args.weight_decay},
            {'params': [p for n, p in model.named_parameters() if any(nd
in n for nd in no_decay)], 'weight_decay': 0.0}
        ]
    else:
        optimizer_grouped_parameters = model.parameters()

    optimizer = AdamW(optimizer_grouped_parameters, lr = args.
learning_rate, eps = args.eps)

    return optimizer


def create_explr_scheduler(optimizer, args):

    scheduler = ExponentialLR(optimizer, gamma = args.gamma, last_
epoch=-1,verbose=False)

    return scheduler
def format_time(elapsed):
    elapsed_rounded = int(round((elapsed)))
    return str(datetime.timedelta(seconds=elapsed_rounded))
```

```
def get_accuracy(logits, labels):
    labels_flat = labels.flatten()
    return (np.sum(logits == labels_flat) / len(labels_flat))*100


def  train(model,  train_dataloader,  valid_dataloader,  optimizer,
scheduler, args):

    epoch_step = 0
    early_stop_loss = list()
    total_t0 = time.time()
    best_loss, best_model = None, None

    for epoch_i in range(args.epochs):

        print()
        print('======== Epoch {:} / {:} ========'.format(epoch_i + 1,
args.epochs))

        t0 = time.time()

        total_train_loss = 0

        model.train()
        for _, batch in enumerate(train_dataloader):
            input_ids = batch['input_ids'].to(args.device)
            attention_mask = batch['attention_mask'].to(args.device)
            token_type_ids = batch['token_type_ids'].to(args.device)
            labels = batch['label'].to(args.device)
            model.zero_grad()

            logits = model(input_ids, attention_mask, token_type_ids)

            train_loss = F.cross_entropy(logits, labels.long())
```

```
            total_train_loss += train_loss.item()

            train_loss.backward()

            torch.nn.utils.clip_grad_norm_(model.parameters(),
max_norm=5.0)

            optimizer.step()
            scheduler.step()

        avg_train_loss = total_train_loss / len(train_dataloader)

        training_time = format_time(time.time() - t0)

        print("Average Train Loss :   {}".format(avg_train_loss))
        print("Training epoch took:   {}".format(training_time))
        print()

        t0 = time.time()

        total_valid_loss = 0
        logit_lst = np.array([])
        label_lst = np.array([])

        model.eval()
        for _, batch in enumerate(valid_dataloader):

            input_ids = batch['input_ids'].to(args.device)
            attention_mask = batch['attention_mask'].to(args.device)
            token_type_ids = batch['token_type_ids'].to(args.device)
            labels = batch['label'].to(args.device)

            with torch.no_grad():
                logits = model(input_ids, attention_mask, token_type_ids)
```

```
                valid_loss =  F.cross_entropy(logits, labels.long())
                total_valid_loss += valid_loss.item()

            logits = logits.detach().cpu().numpy()
            logits_flat = np.argmax(logits, axis=1).flatten()
            labels = labels.to('cpu').numpy()

            logit_lst = np.concatenate((logit_lst, logits_flat))
            label_lst = np.concatenate((label_lst, labels))

        avg_valid_loss = total_valid_loss / len(valid_dataloader)

        if not best_loss or avg_valid_loss < best_loss:
            best_loss = avg_valid_loss
            best_model = deepcopy(model)
            epoch_step = epoch_i + 1

        valididation_time = format_time(time.time() - t0)

        print("Average Valid Loss    :{}".format(avg_valid_loss))
        print("Epoch Valid Accuracy  :{}".format(get_accuracy(logit_lst,
label_lst)))
        print("Validation epoch took :{}".format(valididation_time))

        if len(early_stop_loss) == 0 or avg_valid_loss > early_stop_loss[-1]:
            early_stop_loss.append(avg_valid_loss)
            if len(early_stop_loss) == args.early_stop:break
        else: early_stop_loss = list()
    model_saved_path = args.path + '_epochs_' + str(epoch_step) + '.pt'
    torch.save(best_model.state_dict(), model_saved_path)
    print('TRAIN DONE.')
    print("\nTotal  training  took  {:}  (h:mm:ss)".format(format_time
(time.time()-total_t0)))
```

```
    return model_saved_path


def test(model, test_dataloader, args):

    logit_lst = np.array([])
    label_lst = np.array([])

    for _, batch in enumerate(test_dataloader):

        input_ids = batch['input_ids'].to(args.device)
        attention_mask = batch['attention_mask'].to(args.device)
        token_type_ids = batch['token_type_ids'].to(args.device)
        labels = batch['label'].to(args.device)

        with torch.no_grad():
            logits = model(input_ids, attention_mask, token_type_ids)

        logits = logits.detach().cpu().numpy()
        logits_flat = np.argmax(logits, axis=1).flatten()
        labels = labels.to('cpu').numpy()

        logit_lst = np.concatenate((logit_lst, logits_flat))
        label_lst = np.concatenate((label_lst, labels))

    print('Accuracy: {}(%)'.format(get_accuracy(logit_lst, label_lst)))

def trainer(args):
    train_dataset = LawDataset(args.train_data)
    valid_dataset = LawDataset(args.valid_data)

    tokenizer = AutoTokenizer.from_pretrained(args.model)
    collator = SampleCollate(tokenizer, args)
```

```
    train_dataloader = DataLoader(train_dataset,
                                  batch_size=args.batch_size,
                                  shuffle=args.shuffle,
                                  collate_fn=collator)

    valid_dataloader = DataLoader(valid_dataset,
                                  batch_size=args.batch_size,
                                  shuffle=args.shuffle,
                                  collate_fn=collator)

    model=    LawModel(len(train_dataset.label_id),    len(tokenizer),
args).to(args.device)
    optimizer = create_adamw_optimizer(model, args)
    scheduler = create_explr_scheduler(optimizer, args)

    saved_path  =  train(model,  train_dataloader,  valid_dataloader,
optimizer, scheduler, args)

    return saved_path


def tester(model_path, args):

    test_dataset = LawDataset(args.test_data)

    tokenizer = AutoTokenizer.from_pretrained(args.model)
    collator = SampleCollate(tokenizer, args)
    test_dataloader = DataLoader(test_dataset,
                              batch_size=args.batch_size,
                              shuffle=args.shuffle,
                              collate_fn=collator)

    model=    LawModel(len(test_dataset.label_id),    len(tokenizer),
args).to(args.device)
```

```
    model.load_state_dict(torch.load(model_path))

    test(model, test_dataloader, args)

def seed_everything(args):
    random.seed(args.random_seed)
    np.random.seed(args.random_seed)
    os.environ["PYTHONHASHSEED"] = str(args.random_seed)
    torch.manual_seed(args.random_seed)
    torch.cuda.manual_seed(args.random_seed)
    torch.backends.cudnn.deterministic = True
    torch.backends.cudnn.benchmark = False

def argument_parser():
    args = Namespace(
        train_data = 'mini_ratio_train_dup.csv',
        valid_data = 'mini_ratio_valid_dup.csv',
        test_data = 'mini_ratio_test_dup.csv',
        path = 'klue-roberta-large-law-mini-best',
        model = 'klue-roberta-large-law',
        dropout = 0.5,
        hidden_size = 1024,
        epochs = 10,
        early_stop = 4,
        max_length = 512,
        batch_size = 32,
        padding = True,
        truncation = True,
        shuffle = True,
        no_decay = None,
        weight_decay = 1e-5,
        learning_rate = 5e-5,
        gamma = 0.996,
        eps = 1e-8,
```

```
        random_seed = 42,
        device = 'cuda' if torch.cuda.is_available() else 'cpu'
    )
    return args

def main(args):
    torch.cuda.empty_cache()
    saved_path = trainer(args)
    tester(saved_path, args)

if __name__ == '__main__':
    args = argument_parser()
    seed_everything(args)
    main(args)
```

Ⅳ. 결과

그 결과 전체 테스트셋에 대하여 다음과 같은 혼동행렬(confusion matrix)을 얻을 수 있었습니다.

actual \ predicted	[0, 1/3]	(1/3, 2/3]	(2/3, 1]
[0, 1/3]	67	12	2
(1/3, 2/3]	12	42	2
(2/3, 1]	1	6	1

[그림 9] 혼동행렬: 보행자 사고의 경우 (분류정확도 75.86%)

actual \ predicted	[0, 1/3]	(1/3, 2/3]	(2/3, 1]
[0, 1/3]	217	28	1
(1/3, 2/3]	65	55	0
(2/3, 1]	12	10	1

[그림 10] 혼동행렬: 전체 사고의 경우 (분류정확도 70.18%)

결국 3분위 분류 정확도는 보행자 사고에 대해서는 75.86%를, 전체 사고에 대해서는 70.18%를 얻을 수 있었습니다. 1차 연구결과와 비교하면, 1차에서 BERT 모델을 사용하여 보행자 판결문에 대해서만 69.9%를 얻은 것에 비해 성능향상을 보이고 있고, 비록 1차에서 매뉴얼한 레이블링을 했을 때 얻은 알고리듬 별 75.4~80.8%의 예측정확도보다는 약간 떨어지나, 레이블링 작업을 전혀 하지 않고 그에 비견할 만한 예측정확도를 확보했다고 평가할 수 있습니다.

제3절 평가

여전히 이러한 정도의 3분위 분류 정확도만으로 법관의 과실상계 비율에 대한 판단을 신뢰성 있게 보조할 수 있는 수준이라고 하기는 어렵습니다. 그러나 1차 연구에서와 같은 데이터 특유적인 레이블링 방식을 동원하지 않고, 최근의 추세에 맞게 다양한 판결문 데이터에 적용할 수 있는 언어모델의 도메인 적응과 미세조정만으로 성능향상을 얻었다는 점은 의미가 있습니다. 이러한 방법을 통해 기존에 분석했던 보행자 사고 뿐 아니라 모든 종류의 사고에 대한 예측을 할 수 있다는 점 또한 뚜렷한 개선점입니다.

향후 이러한 모델을 추가적인 판결문 데이터셋의 투입을 통해 계속적으로 고도화하고 다양한 종류의 손해배상 사건을 비롯한 민사사건에 대한 예측작업에 적용해 나갈 수 있을 것으로 기대됩니다.

제4장

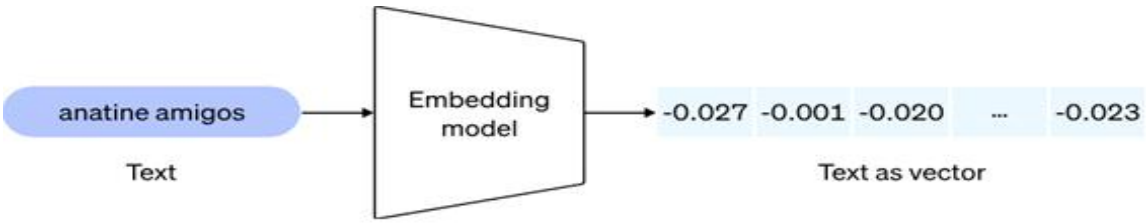
RoBERTa 임베딩의 코사인 유사도를 활용한 예측

제1절 유사 판시 검색에 의한 예측모델의 검토

Ⅰ. 개요

앞 장에서 한 실험으로써 예측정확도를 통해 모델의 성능을 측정할 수 있었지만, 이러한 삼분위 예측만으로 법관의 판단에 실질적인 도움을 주기는 어려울 수 있습니다. 실무적 도움을 위해서는 과거의 판결문 중 과실상계에 대해 유사한 판시를 하였던 판결문들을 검색하여 그 판결문에서 어떻게 판시했는지를 일목요연하게 보여줄 필요도 있을 것입니다.

언어모델의 임베딩(embedding)은 개념의 표상(representation)을 숫자의 시퀀스(sequence)로 변환된 것을 의미하며, 이를 통해 컴퓨터는 이들 개념 간의 관계를 쉽게 이해할 수 있습니다. 특히 언어모델의 임베딩에서의 각 토큰(token)의 거리를 측정하면 그 토큰 간의 의미론적 연관도를 파악할 수 있습니다.



[그림 11] 임베딩의 개념도[33]

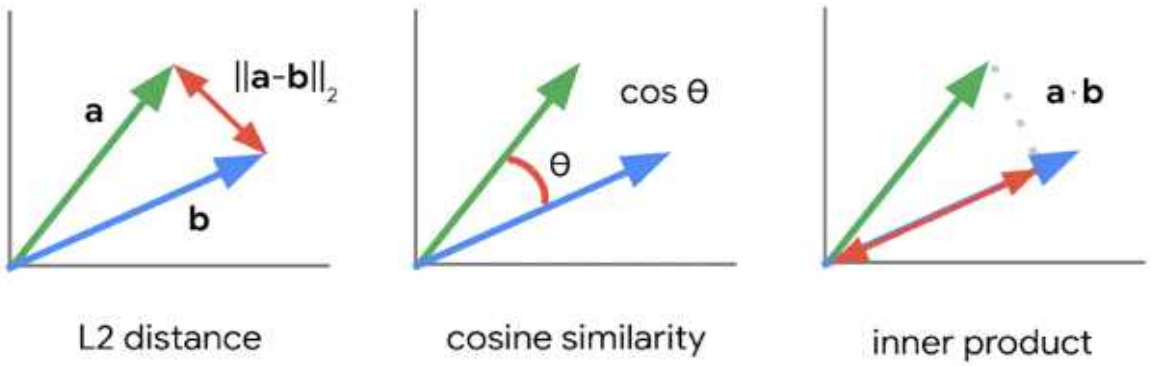
특히 그 거리 측정의 지표로 가장 많이 활용되는 것은 코사인 유사도(cosine similarity)로서, 벡터 A, B에 대한 코사인 유사도는 다음과 같이 측정합니다.

$$\text{similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \times \sqrt{\sum_{i=1}^{n} (B_i)^2}}$$

[그림 12] 코사인 유사도의 산식[34]

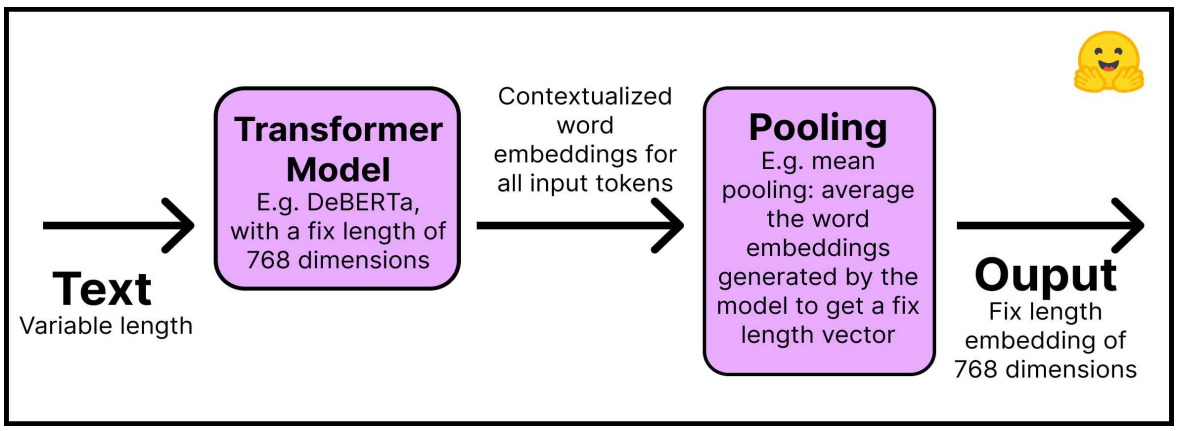
33) OpenAI, New and Improved Embedding Model, https://openai.com/blog/new-and-improved-embedding-model.

34) 위키백과, 코사인 유사도, https://ko.wikipedia.org/wiki/%EC%BD%94%EC%82%AC%EC%9D%B8_%EC%9C%A0%EC%82%AC%EB%8F%84.



[그림 13] 코사인 유사도의 개념도와 유사 측도와의 관계[35]

이러한 의미론적 유사도를 비단 단어 형태의 토큰에 한하지 않고 문장에 대해서도 측정할 수 있는 문장 임베딩(sentence embedding)이 과거의 word2vec, glove 등 단어 임베딩 방식과 구분되는 현대의 트랜스포머 기반 언어모델의 강점입니다. 엄밀히 말하면, 이 때 문장 자체에 대한 임베딩을 구한다기보다는, 문장을 구성하는 각 토큰(주로 단어)들의 문맥에 따른 임베딩을 먼저 구한 후, 이들의 평균값을 취하는 평균 풀링(mean pooling) 방식으로 문장의 임베딩을 산출하고 이에 기해 문장 간의 의미론적 연관성도 파악하게 됩니다.



[그림 14] 문장 임베딩의 산출 방식[36]

35) Google Cloud, Finding Anything Blazingly Fast with Google's Vector Search Technology (2021), https://cloud.google.com/blog/topics/developers-practitioners/find-anything-blazingly-fast-googles-vector-search-technology?hl=en.
36) Hugging Face, Train and Fine-Tune Sentence Transformers Models (2022), https://huggingface.co/blog/how-to-train-sentence-transformers.

Ⅱ. 임베딩

본 연구에서는 제3장에서 RoBERTa 모델을 도메인 적응시킨 법률 맞춤형 모델의 임베딩을 그대로 SentenceTransformer로 불러와 활용하였습니다. 이와 같이 한국어 법률 코퍼스를 투입하여 맞춤형으로 조정된 임베딩을 통해 판결문에 사용되는 문장의 유사도를 최대한 정확하게 파악해 보고자 하였습니다.

Ⅲ. 실험

이러한 벡터공간에 제3장에서 설명 드린 방법으로 추출한 과실상계 판단의 근거가 되는 판시내용(단, 과실상계비율은 삭제)을 배치(embed)시켜 그 좌표를 파악하였습니다. 이러한 좌표 간의 코사인 유사도를 구함으로써, 각각의 판결문마다 판시내용이 가장 유사한 다른 판결문이 무엇인지 검색한 후 코사인 유사도 순위로 상위 10건을 선정하였습니다. 코사인 유사도가 0.95가 넘지 않는 판결은 상위랭킹에서 탈락시키되, 그 어떠한 판결도 코사인 유사도가 0.95에 미치지 못하면 최상위랭킹의 판결문을 검색하였습니다.

이와 같이 상위의 유사도를 가진 판결문들을 검색한 후, 이 판결문들이 판시한 과실상계비율들을 다음과 같이 취합하였습니다.

- 연속값의 예측: 상위 판결문들이 판시한 과실상계비율의 평균을 사용
- 삼분위(이산값)의 예측: 상위 판결문들이 판시한 과실상계비율 중 최빈값을 사용. 일종의 다수결(majority voting) 방식이라고 할 수 있음.

이를 구현하기 위한 코드는 다음과 같습니다.

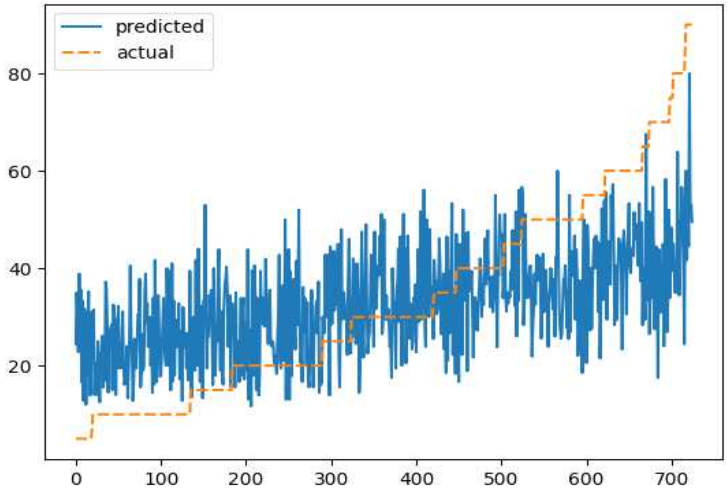
```
from sentence_transformers import SentenceTransformer, models, util
import numpy as np
import pandas as pd
import statistics
from collections import Counter
import seaborn as sns
import matplotlib.pyplot as plt
import sklearn as sk
```

```
word_embedding_model = models.Transformer("klue-roberta-large-law", max_seq_length=256)
pooling_model = models.Pooling(word_embedding_model.get_word_embedding_dimension())
embedder = SentenceTransformer(modules=[word_embedding_model, pooling_model])

data = pd.read_excel('dataset.xlsx', na_filter=False)
corpus, holdings, tertiles, rates = [], [], [], []
for id, row in data.iterrows():
  corpus.append(row['판시내용'].replace(str(row['비율']) + '%', '').replace(str(100 - row['
비율']) + '%', ''))
  holdings.append(row['판시내용']
  rates.append(int(row['비율']))
  tertiles.append(int(row['비율'] / 100 * 3))

pool = embedder.start_multi_process_pool()
embeddings = embedder.encode_multi_process(corpus, pool)

top_k, threshold = 10, 0.95
predicted_rates, predicted_tertiles = [], []
for i in range(len(data)):
  cos_scores = util.pytorch_cos_sim(embedder.encode(corpus[i]), embeddings)[0].cpu()
  top_results = np.argpartition(-cos_scores, range(top_k))[0:top_k + 1]
  predicted_rate, predicted_tertile = [], []
  for idx in top_results[1:top_k + 1]:
    if cos_scores[idx] > threshold:
      predicted_rate.append(rates[idx])
      predicted_tertile.append(tertiles[idx])
  if not predicted_rate:
    predicted_rate.append(rates[top_results[1]])
    predicted_tertile.append(tertiles[top_results[1]])
  predicted_rates.append(statistics.mean(predicted_rate))
  predicted_tertiles.append(Counter(predicted_tertile).most_common()[0][0])

sns.lineplot(data=pd.DataFrame([[predicted_rates[i], rates[i]] for i in np.arg
sort(rates)], columns=['predicted', 'actual']))
plt.show()
errors = 0
for i in range(len(rates)):
  errors += (predicted_rates[i] - rates[i]) ** 2
print((errors / len(rates)) ** 0.5)

cm = sk.metrics.confusion_matrix(tertiles, predicted_tertiles)
disp = sk.metrics.ConfusionMatrixDisplay(cm)
disp.plot()
plt.show()
```

Ⅳ. 결과

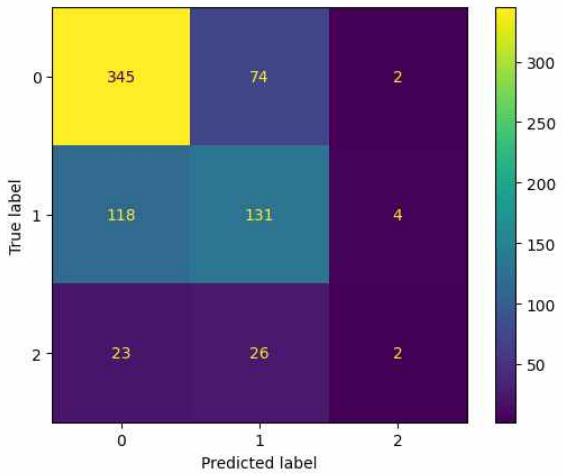
1. 보행자 사고 판결 총 725건

연속값으로서의 과실상계비율의 예측 결과는 이하와 같으며, 평균제곱오차(mean squared error; MSE)의 제곱근은 16.6%로 측정됩니다.



[그림 15] 실제 과실상계비율과 유사 판결문들의 판시를 평균한 예측치의 비교

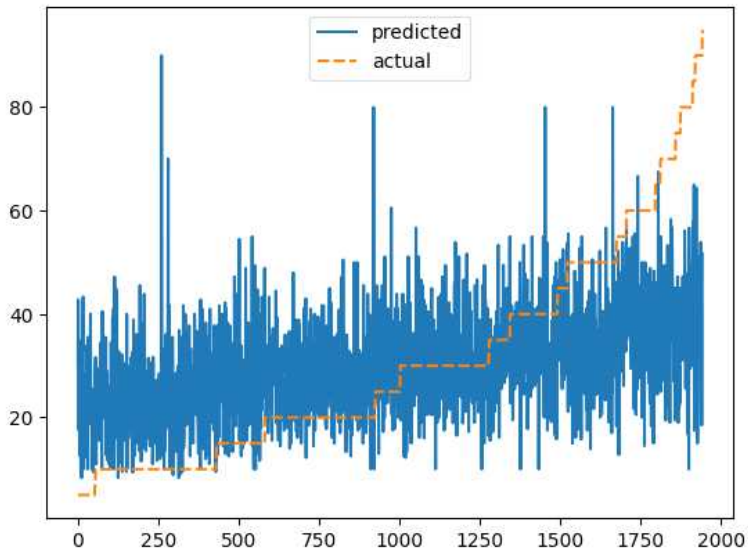
과실상계비율의 삼분위 범주의 예측 결과는 이하와 같으며, 결국 예측정확도(predicted accuracy)는 65.9%로 산출됩니다.



[그림 16] 혼동행렬

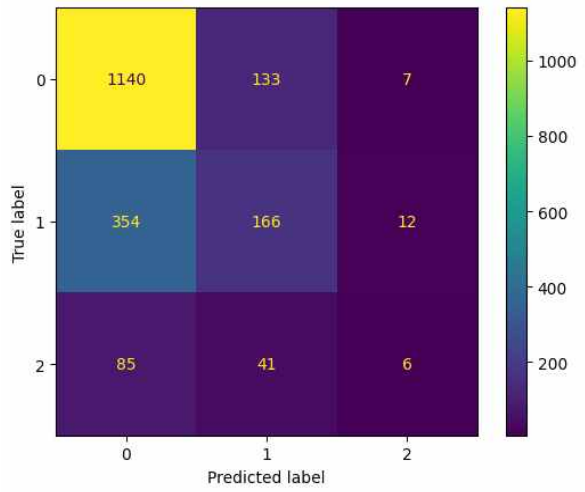
2. 전체 사고 판결 총 1,944건

연속값으로서의 과실상계비율의 예측 결과는 이하와 같으며, 평균제곱오차(mean squared error; MSE)의 제곱근은 18.1%로 측정됩니다.



[그림 17] 실제 과실상계비율과 유사 판결문들의 판시를 평균한 예측치의 비교

과실상계비율의 삼분위 범주의 예측 결과는 이하와 같으며, 결국 예측정확도는 67.5%로 산출됩니다.



[그림 18] 혼동행렬

제2절 실무에서 활용 가능한 애플리케이션의 가능성

이상의 모델은 실무에서 활용 가능한 애플리케이션으로 그대로 활용될 수 있습니다. 예컨대 보행자 사고 관련 데이터를 불러온 후 위 코드에 단지 다음 코드만 추가하여 실행하면 애플리케이션으로 활용할 수 있습니다.

```
top_k = 10
fact = input("사실관계를 입력하십시오:\n")
print("\n")
cos_scores = util.pytorch_cos_sim(embedder.encode(fact), embeddings)[0].cpu( )
top_results = np.argpartition(-cos_scores, range(top_k))[0:top_k + 1]
no = 1
for idx in top_results[1:top_k + 1]:
    print(str(no) + ". " + holdings[idx].strip() + " (유사도: " +
str(cos_score
s[idx].item()) + ", 과실비율: " + str(rates[idx]) + "%)")
    no += 1
```

임의의 사실관계를 입력할 때마다 다음과 같은 상위 10건의 의미론적 검색결과를 얻을 수 있음을 확인할 수 있습니다.

```
사실관계를 입력하십시오:
밤에 빨간 불에 횡단보도 무단횡단

1. 앞서 본 사고경위에 비춰 망인은 고령에 신호를 위반하여 횡단보도 부근에서 도로를
   건너려다가 이 사건 사고가 발생한 점에서 피고의 책임을 50%로 제한한다.
   (유사도: 0.8284025192260742, 과실비율: 50%)
2. 이 사건 사고 당시 망인도 어두운 새벽에 차량진행신호에 도로를 무단횡단한 잘못이
   있으므로, 이러한 사정을 참작하여 피고의 책임을 55%로 제한한다.
   (유사도: 0.8270647525787354, 과실비율: 45%)
3. 망인도 보행신호등의 녹색등화가 점멸하는 상태에서 횡단보도를 보행하기 시작하여
```

보행신호등이 적색등화로 바뀐 후에도 횡단보도를 보행한 잘못이 있으므로, 이러한 사정을 참작하여 피고들의 책임을 65%로 제한한다.
(유사도: 0.8205572366714478, 과실비율: 35%)

4. 원고의 과실비율 10%(야간에 신호등이 없는 횡단보도를 건널 때 좌우측 차량의 진입 여부를 잘 살피고 건너야 할 의무가 있음에도 이를 게을리 한 점).
(유사도: 0.8169166445732117, 과실비율: 10%)

5. 다만, 망인으로서도 야간에 편도 5차로 도로에서 보행자 적색신호에 횡단보도를 횡단한 잘못이 있으므로 망인의 과실을 60%로 보고, 피고의 책임을 40%로 제한한다.
(유사도: 0.8137151002883911, 과실비율: 60%)

6. 앞서 든 증거에 의하면, 이 사건 사고는 원고 A이 술에 취한 상태에서 야간에 검은색 점퍼를 입고 보행자 적색신호에 횡단보도를 횡단하던 과정에서 발생하였는바, 원고 A의 이러한 과실을 참작하여 피고의 책임을 55%로 제한한다.
(유사도: 0.8130747079849243, 과실비율: 45%)

7. 망인은 보행신호등이 있는 횡단보도가 인근에 설치되어 있었음에도 시야가 어두운 심야에 위 횡단보도에 이르기 전 적색의 보행자신호에 도로를 무단 횡단한 잘못이 있으므로, 이러한 사정을 피고가 배상할 손해액을 산정함에 있어 참작하기로 하되 망인의 과실을 60%로 보고 피고의 책임을 40%로 제한한다.
(유사도: 0.8117626905441284, 과실비율: 60%)

8. 한편, 새벽 시간에 운전자의 눈에 잘 띄지 않는 짙은 색의 옷을 입은 채 횡단보도 신호를 무시하고 무단횡단을 한 원고의 과실 또한 이 사건 교통사고 발생과 손해 확대의 원인이 되었다고 인정되므로, 이를 참작하여 피고의 책임을 50%로 제한한다.
(유사도: 0.8063555955886841, 과실비율: 50%)

9. 다만, 망인으로서도 야간에 편도 4차로의 도로에서 보행자 적색신호에 횡단보도를 횡단한 잘못이 있고, 이와 같은 망인의 과실을 참작하여 피고들의 책임을 40%로 제한한다.
(유사도: 0.8060821890830994, 과실비율: 60%)

10. 원고가 대학교 구내 도로를 횡단함에 있어 차량 통행 등을 살펴 안전함을 확인한 후 횡단하여야 했음에도 이를 게을리 한 부주의가 이 사건 사고로 인한 손해의 발생 및 확대에 기여하였으므로, 이를 손해배상액의 산정에 참작하기로 하여, 피고의 책임을 80%로 제한한다.
(유사도: 0.7377910017967224, 과실비율: 20%)

사실관계를 입력하십시오:
밤에 도로 위에 서 있다가 잘 보이지 않아 사고가 남

1. 이 사건 사고는 야간에 어두운 색깔의 옷을 입고 옆에 있는 횡단보도를 놔둔 채 도로를 무단횡단한 원고의 잘못과 전방 주시의무를 소홀히 하고 급하게 차선을 변경하여 원고를 들이받은 C의 잘못이 경합하여 발생하였으므로 앞서 본 사고경위에 비춰 피고의 책임을 50%로 제한한다.
   (유사도: 0.8503914475440979, 과실비율: 50%)
2. 한편, 원고는 일출 전 날이 어두운 시각에 삼거리 교차로의 모서리 지점 도로에 서서 택시를 잡으려다가 이 사건 사고를 당한 점, 그 곳은 인도가 아니라 가로수 등이 심어져 있어 차량 운전자가 위 지점에 사람이 서있으리라고 예상하기 어려운 곳인 점, 이 사건 사고 후 응급실에 이송되었을 때 원고에게서 술 냄새가 났던 점, 그 밖에 기록에 나타난 이 사건 사고의 경위 등에 비추어 원고에게도 인도가 아닌 곳에서 택시를 잡으면서 본인의 안전을 소홀히 한 과실이 있고, 위와 같은 과실 역시 이 사건 사고로 인한 손해의 발생 및 확대에 기여하였다고 보이므로, 원고의 과실 비율을 20%로 보고 피고의 책임을 80%로 제한한다.
   (유사도: 0.8489103317260742, 과실비율: 20%)
3. 다만 사고 당시 그곳은 야간에 차량에 대한 황색 점멸 신호 상태의 횡단보도였으므로, 망인에게도 진행하는 차량이 있는지를 잘 살펴 스스로 안전을 도모할 의무가 있음에도 이를 그르쳐 스스로 안전을 도모하지 못한 잘못이 있으므로, 피고의 책임을 90%로 제한한다.
   (유사도: 0.8444902896881104, 과실비율: 10%)
4. 이 사건 사고시각이 저녁으로 주위가 어두웠던 점, 이 사건 사고장소는 신호가 없는 혼잡한 사거리 교차로였던 점 등을 고려하면, 원고로서도 주위를 잘 살펴 안전하게 횡단할 의무가 있으므로, 피고의 책임을 90%로 제한한다.
   (유사도: 0.8437597751617432, 과실비율: 10%)
5. 다만, 망인으로서도 야간에 가로등이 설치되어 있지 않은 도로 위를 보행한 잘못이 있고 사고 태양에 비추어 볼 때 무단횡단에 준하는 정도의 잘못이므로 망인의 과실을 35%로 보고, 피고의 책임을 65%로 제한한다.
   (유사도: 0.8431017398834229, 과실비율: 35%)

6. 다만, 이 사건 사고가 발생한 횡단보도에는 시작 지점과 종료 지점 사이에 보행자가 안전하게 머물러 있을 수 있는 넓은 공간이 설치되어 있는데, 원고가 보행자 신호를 위반하여 무리하게 횡단하다가 이 사건 사고가 발생하게 되었고, 이 사건 사고 발생 시각이 야간이어서 피고 차량의 시야 확보에 어느 정도 어려움이 있었을 것으로 보이는 점 등을 참작하여 피고의 책임 비율을 50%로 제한한다.
(유사도: 0.8421773910522461, 과실비율: 50%)
7. 한편, 망인도 술에 취하여 도로에 누워 있으면 지나는 차량에 의해 충격이나 역과를 당할 위험이 높음에도 불구하고 위 아파트 내 도로에 술에 취해 누워 있다가 이 사건 사고를 당한 과실이 있고, 이 사건 사고 당시 한밤중이었던 점, 사고장소가 아파트 내 도로였던 점 등 제반 사정을 감안할 때 피고의 책임을 40%로 제한함이 상당하다.
(유사도: 0.8412953019142151, 과실비율: 60%)
8. 원고 A가 도로가에 주차된 차량 사이로 횡단보도 없는 도로를 횡단하다가 이 사건 사고가 발생한 경위를 참작하여 피고 책임을 80%로 제한한다.
(유사도: 0.8411918878555298, 과실비율: 20%)
9. 앞서 본 사고경위에 비춰 망인은 고령에 신호를 위반하여 횡단보도 부근에서 도로를 건너려다가 이 사건 사고가 발생한 점에서 피고의 책임을 50%로 제한한다.
(유사도: 0.8410850167274475, 과실비율: 50%)
10. 원고가 대학교 구내 도로를 횡단함에 있어 차량 통행 등을 살펴 안전함을 확인한 후 횡단하여야 했음에도 이를 게을리 한 부주의가 이 사건 사고로 인한 손해의 발생 및 확대에 기여하였으므로, 이를 손해배상액의 산정에 참작하기로 하여, 피고의 책임을 80%로 제한한다.
(유사도: 0.7862423062324524, 과실비율: 20%)

제3절 평가

이러한 검색(retrieval) 방식은 언어모델 자체를 도메인 적응하고 미세조정하여 곧바로 예측치를 얻어내는 것만큼의 정확도를 보이지는 않고 있지만, 유사한 판시내용들에 대한 정보를 담고 있다는 점에서 실무상 활용성은 높을 수 있습니다. 또한 이러한 방식은 비단 손해배상 사건에 한하지 않고, 다양한 유형의 재판에서 과거의 유사한 판결을 참조함에 있어 간편하고 효율적인 툴을 제공할 수 있습니다.

제5장

결론

이번 후속연구에서는 1차 연구에서 주로 활용된 인력을 동원한 레이블링을 피하고, 대신 언어모델 자체에 법률 코퍼스를 투입하여 추가 훈련을 하고 하류작업에 맞게 미세조정하여 이를 가지고 예측하면서 성능의 향상 여부를 측정하였습니다. 이는 비지도학습을 근간으로 하는 현대의 언어모델의 발전 양상에도 부합하는 접근방식입니다.

다만, 몇 가지 실험적 연구 결과 당초 기대와 달리 1차 연구에서 활용한 레이블링 하에서의 지도학습만큼의 정확성은 아쉽게도 달성하지 못하였습니다. 바로 현장에 투입될 만큼의 신뢰할 수 있는 모델이라고 보기도 어렵습니다. 앞으로 좀 더 다양한 영역의 다량의 판결문 데이터의 투입 등을 통해 모델 성능의 향상이 절실하다는 점도 확인할 수 있었습니다.

그러나 기존의 범용 언어모델을 적절한 데이터로 추가 훈련할 경우 레이블링을 할 때에 비견될 만한 성능을 효율적으로 얻을 수 있다는 점은 확인할 수 있습니다. 바로 이 점으로 인해, 기존에 보행자에 연구를 한정했던 것과 달리 모든 교통사고에 대해서도 분석할 수 있는 등, 분석의 범위와 효율성이 크게 제고되었습니다. 이 점은 곧바로 다량의 판결문을 투입하는 등 코퍼스의 수를 늘리는 작업이 큰 추가비용 지출 없이 이루어질 수 있음을 시사하는 것입니다. 실제 언어모델의 여러 실증된 성능향상법칙(scaling law)상 GPT-4 등 대규모 언어모델에 비견되는 수준까지는 아니더라도 미공개 하급심 판결문을 이 연구에 투입된 것보다 다량 투입할 경우 예측모델의 성능이 크게 늘어날 것임은 충분히 예상할 수 있습니다.

나아가, 가능성과 동시에 많은 한계를 노정하였던 이번 후속 작업을 통해 다음과 같은 추가적인 시사점도 얻을 수 있었습니다.[37)]

첫째, 본 연구에 투입된 수준을 훨씬 뛰어넘는 수준의 다량의 하급심 판결문 코퍼스를 모델에 추가 투입하여 훈련하는 것이 향후 언어모델 성능 향상의 관건일 것으로 생각되며, 특히 GPT-4 등 폐쇄모델을 비롯하여 좀 더 다양한 첨단의 모델로 훈련을 하고 그 밖의 다양한 방법으로 민간의 역량을 낮은 비용으로 활용하기 위해서도, 판결문 공개 범위의 조정에 대한 논의가 조심스럽게 이어져야 할 것으로 생각됩니다. 현재와 같은 상태에서는 법원이 자체 서버와 GPU 등 장비를 구축하여 언어모델을 훈련시킬

37) 두 번째, 세 번째 시사점을 정리함에 있어 한국정보법학회, 사법정책연구원의 2023년 추계공동 정기학술세미나("AI와 사법제도")에서의 오세용 부장판사님의 토론 내용으로부터 큰 인사이트를 얻었음을 밝힙니다.

수밖에 없으나 향후 계속적인 업데이트 등에 필요한 비용까지 고려해 볼 때 이는 비용 효율적이지 않은 측면이 있습니다. 법률 플랫폼들은 이미 미공개 판결문들을 상당 수 확보해 가면서 인공지능 모델의 훈련에 돌입된 상태라 향후 민간의 법률 정보의 불균등한 보유가 오히려 심화될 우려도 있습니다.

둘째, 의미 있는 판결예측모델을 훈련하기 위해서는 1차 연구와 이번 후속 연구와 같이 판결문에 정리된 사실관계로부터 판결의 결과를 예측하기보다는 당사자가 제출한 서면으로부터 판결의 결과를 예측하는 모델의 개발로 이행되어야 할 것입니다. 향후 이미지와 텍스트 등 다양한 포맷의 데이터를 처리하는 멀타이 모달(multi-modal) 기법이 향상되면 교통사고 사건의 경우 특히 경찰관 작성 실황조사서의 이미지 분석이 판결의 결과에 대한 상당한 예측력을 가질 수 있지 않을까 조심스럽게 예측해 봅니다.

셋째, 나름 근래의 여러 기법을 동원해 보았으나 데이터 수를 크게 늘리지 않으면 바로 현장에 투입할 수 있는 판결예측모델의 훈련에는 많은 한계가 있는 것으로 생각됩니다. 아울러 언어모델의 특성상 과실상계비율 등 숫자의 예측에는 기대한 것만큼의 성능을 얻지 못할 수 있다는 가설도 얻게 되었습니다. 공개된 법률 코퍼스의 비약적 증가 등 전기가 마련되기 전에는, (본 연구의 연구자들이 이 연구의 제안을 할 때까지 그러하였듯이) "예측" 모델의 개발에 지나치게 집착하기보다는, 법관의 과중한 업무를 경감할 수 있는 "생성" 모델(생성의 결과의 정확성에 대한 지나친 고려를 하기 보다는 효율성의 관점에서)의 개발에 좀 더 방점을 두는 것이 바람직할 수 있다는 성찰을 하여 보았습니다.

손해배상 사건에서의 인공지능(AI)
활용방안 (후속연구)

발행일 | 2023년 11월
발행처 | 법원행정처
서울 서초구 서초대로 219
Tel:(02)3480-1247
연구자 | 서울대학교 산학협력단
인쇄처 | (주)현대아트컴
Tel:(02)2278-4482