필터버블을 해소하는 뉴스레터 서비스

글쓴이: YBIGTA 25기 최수현 (언론홍보영상학부/응용통계학과 20학번)

 상속세 개편에 관한 사설 2편

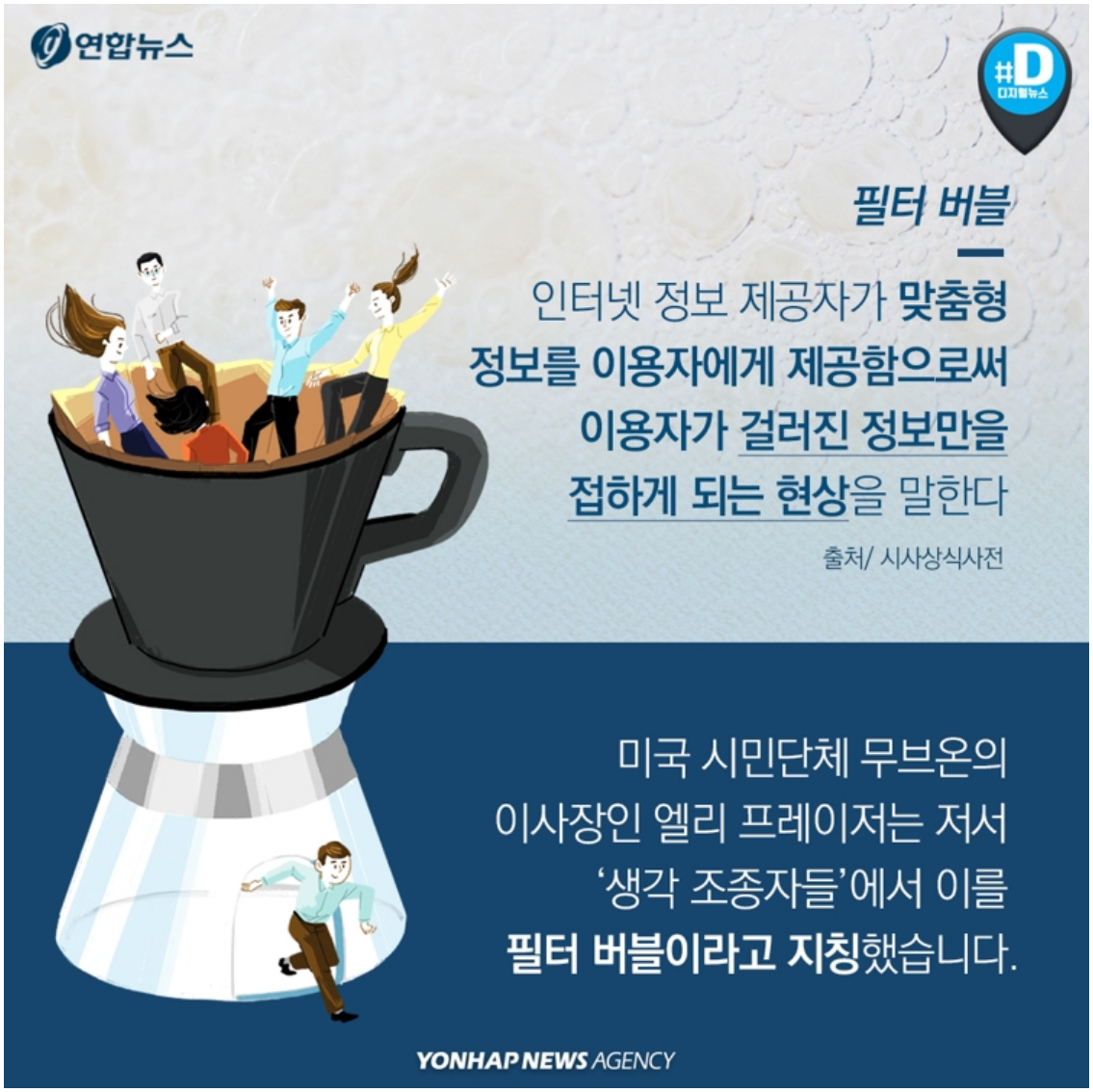
> 💡 " 우리 상속세 체계는 다른 선진국보다 지나치게 과중하다.
> ... <중략>
> 그 결과 서울에 아파트 한 채만 가져도 상속세 걱정을 해야 할 지경에 이르렀다.
> 이것은 상속세의 본뜻이 아니다. <후략> "

> 💡 "내수경기 악화에 고통을 겪는 민심을 달래기보다는
> '부자감세를 완성하겠다'는 의지를 명확히 한 것으로 해석된다.
> ... <중략>
> 그런 처지임에도 부자감세를 이렇게 노골적으로 밀고 가는 건
> 민심은 아랑곳하지 않는 오만, 오기의 국정이 아닐 수 없다."

과연 여러분의 선택은? 어떤 것이 정답일까요?

> **정답은 없습니다 여러분!!**
>
> 그저 그럴듯한 의견만 있을 뿐이죠. 여러분들은 어떤 의견이신가요?
> 여러분이 가지고 계신 의견 다 존중 받아야 마땅합니다.
>
> ...
> ...
> **그런데 ...**
>
> 어쩌다가 그런 의견을 갖게 되었는지 생각해보셨나요?
> 물론, 정말 뉴스를 균형적으로 읽고, 비판적으로 글 읽기에 성공해서
> 자신의 의견을 반대편의 의견과 수도 없이 비교해본 뒤, 얻게 된 의견일 수도 있습니다.
>
> 우리는 이걸 미디어 리터러시라고 부르고,
> 현대 사회에서 많은 사람들이 가지기 힘든 역량입니다.

> 왜냐구요??
>
> **바로 필터버블 때문입니다.**



출처: 연합뉴스

> 내가 스마트폰으로 보는 뉴스, 인스타그램, 유튜브, 넷플릭스, 지도
> 스마트폰 안에 있는 모든 것이
> **다 여러분을 지켜보고 있습니다.**
>
> 너무 편리하죠. 내가 생각한 것만 쏙쏙 전달해주고, 내가 먹으려 했던 거, 내가 보려 했던 거
> 내가 들으려고 했던 거, 찾기 전에 먼저 가져다 주기 때문이죠.

> 편리하다는 말은 힘을 들이지 않아도 된다는 뜻이고,
> 힘이 들지 않는다는 말은 복잡하게 생각할 필요 없다는 말입니다.
>
> 그런데 뉴스를 힘 안 들이고 읽고 보다가는 나도 모르게 큰 코를 다치게 됩니다.
> 뉴스는
> **'사실'만** 전달하는 것이 아니거든요.
>
> 아주 교묘하고, 아주 조심스럽게 의식 공동체를 형성하게 됩니다.
> 댓글까지 보게 된다면,
> **나도 모르게 그들과 생각을 공유하게 될지도 모르죠.**
>
> 그래서 어느 날 생각했습니다.
> 의도적으로 자신과 맞지 않는 뉴스를 계속해서 접하게 된다면
>
> **자신의 성향이 바뀌게 될까? 아니면 그대로일까?**
>
> 그래서 기존의 추천 알고리즘과 반대인 역추천 알고리즘을
> 뉴스 추천에 사용해보는 것을 제안하게 되었습니다!!
>
> 사실 여기까지는 아직 시작이고, 해당 서비스가 시장성도 갖춘 서비스임을
> 입증하는 자료도
> **이미 모두 구비하고 있습니다!**
> 만일 팀에 참여하신다면, 이 부분에 대해서는 크게 준비하지 않아도 될 것 같네요 :)

🛠️ 그래서 구현은 어떻게??

전체적인 파이프라인 설계

1. 데이터 확보 및 저장

1.1 크롤링

- **크롤링 도구:** BeautifulSoup, Requests, Selenium (동적 페이지의 경우)
- **크롤링 대상:** 조선일보, 중앙일보, 동아일보, 한겨레, 경향신문 등의 사설
- **크롤링 주기:** 일일 또는 주간 크롤링 (크론 작업을 통해 자동화)

  데이터의 경우에는 언론사의 논조가 확연하게 돋보이는 **'사설'을 크롤링**할 예정입니다.

  대표적인 **보수 언론사 3사(조중동)**의 사설과 **진보 언론사(한겨레, 경향신문)**등의
  언론사 포털에서 열심히 글을 긁어올 예정입니다.

  우선 프로토타입을 3주라는 시간 안에 만드는 것이 가장 중요하기 때문에,
  label은

0(중도), 1(보수), 2(진보)로 나누거나,
만일 이진분류모델을 사용하게 된다면
0(보수), 1(진보)로만 나눠서
학습을 진행할 예정입니다.

1.2 데이터 저장

- **저장 형식:** csv, json 또는 SQL Database
- **데이터베이스:** SQLite (프로토타입), PostgreSQL/MySQL (확장성 고려)

  크롤링을 한 데이터를 바탕으로 DB를 만들 예정입니다.

  SQL이 아니더라도 **Dataframe** 형태로 사용할 수 있도록

  csv와 json 자료로 정리한다고 생각하시면 됩니다.

  기사 제목, 기사 본문, 논설위원, 언론사 등의 feature를 갖게 될 것이고,
  모델 학습에 이용될 겁니다!

  추후에, DB를 바탕으로 **콘텐츠 기반 필터링 혹은 협업 필터링**을 진행하게 될 수 있으므로
  데이터 수집 시에 최대한 관련 있는 feature를 모두 모을 생각입니다.

2. 데이터 전처리 및 라벨링

2.1 데이터 전처리

- **텍스트 전처리:** 불용어 제거, 토큰화, 표제어 추출
- **라벨링:** 언론사 별로 레이블 할당 (0: 중도, 1: 보수, 2: 진보)

  크롤링 후 해야하는 건 **데이터 전처리와 데이터 라벨링입니다!!**

  필요 없는 불용어를 제거하고, 토픽 별 사설을 제시해야하기 때문에, 표제어도 추출합니다.

  또한, **직접 눈으로 하나 하나 보면서 진보, 보수 라벨링을 붙이게 됩니다.**

  그런데 사실, 언론사의 성향이 너무 뚜렷하기 때문에 언론사 별로 레이블을 미리 지정해서 붙인 다음 눈으로 빠르게 스크리닝하면 금방할 것으로 예상됩니다.

  Ex) 조중동 → 보수 / 한겨레 → 진보 / 한국일보 → 중도

3. 모델 학습 및 파인튜닝

3.1 Kpf-BERT 파인튜닝

- **모델:** Kpf-BERT https://github.com/KPFBERT/kpfbert
- **목적:** 기사 분류 (보수/진보), 유저 성향 분류(보수/진보)
- **라이브러리:** Hugging Face Transformers, PyTorch

  보수/진보 언론사에서 크롤링한 데이터를 바탕으로 label(보수/진보)를 예측하는
  모델을 구축하기 위해
  fine-tuning을 진행합니다.

  train_dataset : val_dataset : test_dataset = 8: 1: 1로 구성해 파인튜닝 후 MLflow를 이용해 로깅하고 confusion matrix등을 확인하며 **최적의 학습 checkpoint를 찾아 모델로 채택합니다.**

4. 예측 및 사용자 성향 분석

4.1 사용자 데이터 입력 및 예측

- **입력:** 사용자 선택 기사 (10-15개) : 진보/보수 기사문 쌍으로 주어져서 이진 선택
- **예측:** 사용자 성향 (보수/진보) 및 예측 확률

  유저가 해당 서비스를 사용하기 전, **자신의 성향과 맞는 기사 문장을 10-15개 고르게 됩니다.**

  해당 기사 문장은 구축된 데이터로 fine-tuning된 Kpf-BERT의 input data로 들어가게 되고, 해당 **유저의 label(보수, 진보)와 예측 확률을 반환하게 됩니다.**

  예측 확률에 따른 가중치를 부여해 label들의 평균을 내서 유저의 성향 data를 구축하게 되며,
  이를 바탕으로 유저에게 유사도가 낮은 사설을 추천해주는 것을 목표로 합니다.
  유저 성향은 2-3차원 벡터로 이뤄질 예정. Ex) [보수확률 | 진보확률 | 중도 확률]

  추후 목표) 유저가 서비스를 이용하면서 쌓이는 데이터 베이스를 바탕으로 유저의 성향을 업데이트하며 기사를 추천하도록..!

5. 유사도 낮은 사설 추천

5.1 유사도 계산 및 추천

- **기술:** 코사인 유사도
- **목적:** 사용자 성향과 반대되는 기사 추천

1) 사용자 성향 벡터화

먼저, 유저가 선택한 기사들에 대한 Kpf-BERT의 예측 확률을 이용하여
사용자 성향을 벡터로 표현합니다.

[보수 확률 | 진보 확률 | or 중도확률] - 2-3차원 벡터

예측 확률은 각 기사에 대해 보수, 중도, 진보의 확률을 반환하여,
이를 유저가 선택한 기사 수만큼 평균 내서 유저 성향 벡터로 저장하게 됩니다.

2) 문서 성향 벡터화

마찬가지로, 모델 구축 이후 자동 크롤링할 모든 기사에 대해서도
Kpf-BERT의 예측 확률을 계산하여 각 기사의 성향을 벡터로 표현하고 저장합니다.

[보수 확률 | 진보 확률 | or 중도확률] - 2-3차원 벡터

이 두 벡터 간의 코사인 유사도를 계산해서 가장 적은 예측 확률을
반환하는 기사를 추천하는 방식으로 진행할 예정입니다.

6. 서비스 구현

6.1 백엔드 (Backend)

- **프레임워크:** Flask, Django
- **API:** RESTful API for handling requests (user data input, recommendation fetching)
- **Database Integration:** SQLAlchemy, Django ORM

사용자 입력을 받아 예측 및 추천을 수행하는 API를 구현합니다.

6.2 프론트엔드 (Frontend)

- **프레임워크:** React, Vue.js
- **기능:** 사용자 입력 인터페이스, 추천 결과 출력
- **통신:** Axios 등으로 백엔드 API와 통신

사용자 입력 인터페이스와 추천 결과를 출력하는 UI를 구현합니다.

7. 배포 및 유지보수

7.1 배포

- **클라우드 서비스:** AWS
- **컨테이너화:** Docker, Kubernetes

7.2 유지보수

- **모니터링:** New Relic, Grafana
- **로깅:** ELK Stack (Elasticsearch, Logstash, Kibana)

팀원 역량

> 💡 사실 이 아이템을 생각한 지는 꽤 오래 됐습니다.
> 하지만, 늘 구현 과정에서 막혀서 이번에도 가지고 오게 됐습니다.
> 그 당시에는 끝내 구현하지 못할 것 같았는데, 이렇게 정리하고 보니까
> 정말 구현할 수 있을 것 같다는 생각을 하게 되었습니다!

그래서 누구??

- 그래서 구현에 초점을 맞추고, 신기플 발표 때까지 기능을 하는
  프로토타입을 만들고 싶으신 분!
- FE, BE 경험이 있으신 분!! - 사실 제일 중요.. (제가 해본 적이 없습니다 ..ㅜ)
- 지겨워서 하품 나올 때까지 크롤링하실 수 있는 분!!
- 모든 걸 떠나서 정말 열심히 방학 마무리하실 수 있는 분들!! 모두 환영입니다!!

왼쪽 같은 플젝이 되지 말고, 열심히 좋은 성과 이뤄서 회식하는 플젝이 되..

많은 관심 부탁드립니다!!!!