



# AI 텍스트 기반 다중감정분석 모델

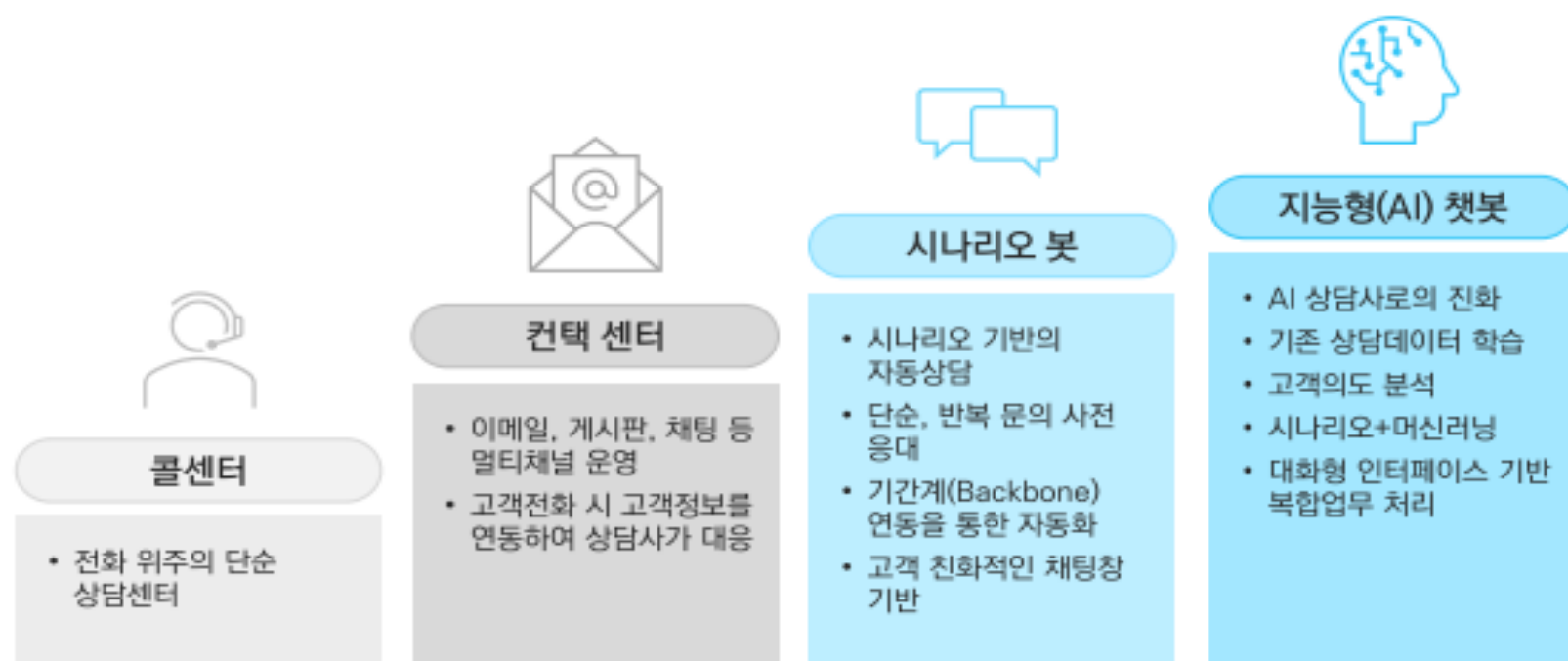
---

주제 설정과 동기\_#금융 #AI

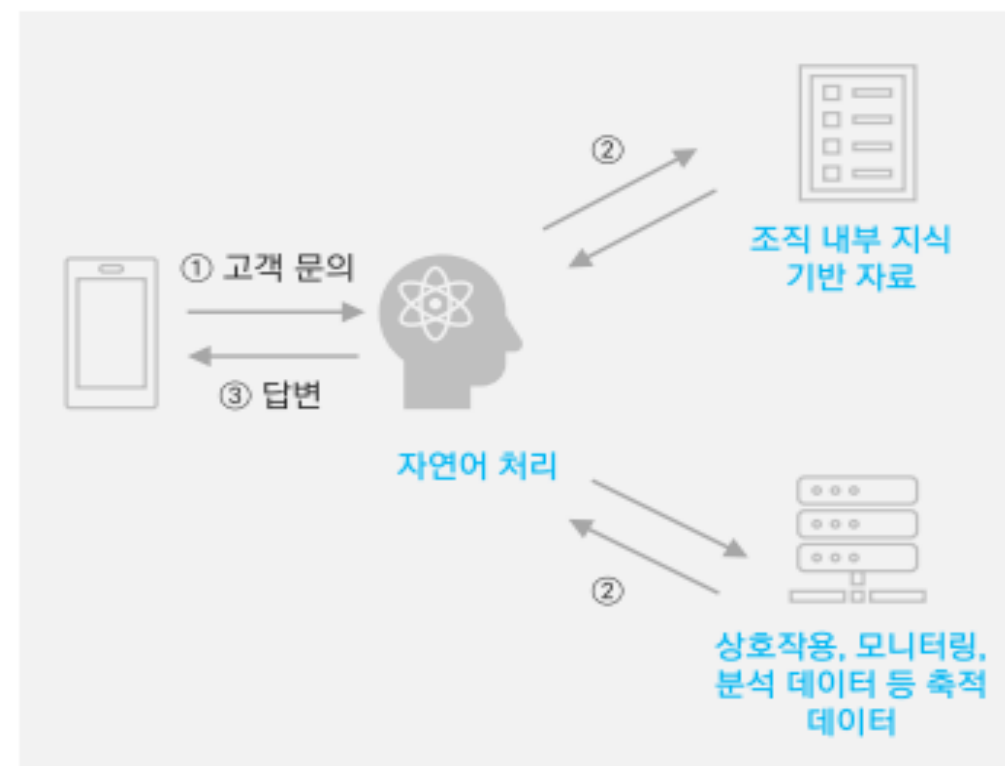
## 프론트 오피스 ① – 고객 상담에서 가상 비서로 진화 (1/2)

챗봇은 고객 상담 분야에서 가장 많이 활용되고 있으며, 고객 상담은 초기에 전화로 대응하는 콜센터에서 시작하여 컨택 센터-시나리오 봇-AI 챗봇으로 진화. 생성형 AI 접목으로 고객 질의에 맞춤형 대처가 가능해져 보다 유연하게 최적의 답변을 고객에게 제공할 수 있을 것으로 기대

### 고객상담 분야 변화 단계






### AI 챗봇 작동 구조



## 프론트 오피스 ① – 고객 상담에서 가상 비서로 진화 (2/2)

AI 기술은 지속적으로 변화하는 고객의 정보를 자동으로 업데이트할 수 있기 때문에, 다양한 채널에서 발생하는 취업, 결혼, 퇴직 등 생활사건(Life event) 데이터 분석을 기반으로 고객 니즈에 부합하는 맞춤형 상품을 적시에 추천이 가능. 이에 따라 글로벌 금융사들은 AI 가상비서를 통한 초개인화 서비스를 제공

### 글로벌 주요 금융사의 AI 가상 비서 서비스 현황

기업	서비스명	주요 특성 및 기능
 BANK OF AMERICA 뱅크오브아메리카(BoA)	에리카 (Erica)	<ul style="list-style-type: none"> <li>2018년 Apple의 Siri를 본떠 출시한 AI 기반 금융 비서로, 365일 실시간 고객 응대가 가능</li> <li>문자와 음성 대화를 통해 계좌조회, 카드관리, 개인송금, 거래보고, 투자조언 등 다양한 유형의 금융서비스를 지원하고 있으며, 사용자 수 또한 지속 성장세</li> </ul>
 WELLS FARGO 웰스파고	파고 (Fargo)	<ul style="list-style-type: none"> <li>구글 대화형 AI 플랫폼 다이얼로그플로우(Dialogflow) 기반 가상 비서로, 언어 처리 기능을 통해 고객의 의도를 이해하고 맞춤형 응답 제공</li> <li>AI, 클라우드를 이용해 언제 어디서나 금융업무와 서비스를 스마트하게 수행할 수 있도록 지원, 고객에게 편리하고 직관적인 बैंकिंग 경험을 제공하고 평범한 언어로 대부분 은행업무에 대한 도움을 주고 있음</li> </ul>
	밴티지 (Vantage)	<ul style="list-style-type: none"> <li>단일 플랫폼에서 중소기업 및 대기업의 금융, 비금융 니즈를 충족시킬 수 있도록 구현한 원스톱숍으로, 기업고객의 디지털 경험 향상을 위한 AI 기반 플랫폼</li> <li>AI/ML 기반으로 기업 성장에 따라 진화하는 금융 니즈에 맞춰 계정을 커스터마이징하는 기업고객 개인화 기능</li> </ul>
 RBC 캐나다 왕립은행 (Royal Bank of Canada, RBC)	노미 (NOMI)	<ul style="list-style-type: none"> <li>RBC 모바일 애플리케이션에 통합된 디지털 비서 서비스로, AI를 기반으로 개인화된 인사이트를 제공하여 고객의 저축, 지출관리 등을 돕고 있음</li> <li>가장 최근에 추가된 NOMI Forecast 서비스는 반복되는 청구서 지불을 추적하여 고객에게 향후 현금 흐름에 대한 예측을 제공하며, AI를 고객 경험에 가장 잘 활용한 것으로 인정 받기도 함</li> </ul>
<b>BlackRock</b> 블랙록	알라딘(Aladdin) 내 서비스	<ul style="list-style-type: none"> <li>알라딘과 eFront 리스크 관리 시스템을 위한 코파일럿 구축에 생성형 AI를 적용함으로써, 고객들은 알라딘에서 정보 추출 시 블랙록의 LLM 기술을 사용할 수 있음</li> <li>*알라딘은 리서치, 리스크 분석, 포트폴리오 관리, 트레이딩 등 투자관리 전반을 아우르는 블랙록의 종합서비스 플랫폼</li> </ul>



# # 상담업무에서의 AI



## 챗봇(ChatBots)

### • 자동응답 및 간단한 문제 해결

- 고객의 기본적인 문의 즉각적으로 답변 제공
- 반복적인 질문에 대한 응답 자동화

\* 24시간 운영되므로 언제든지 고객 지원이 가능하며,  
특히 시간대가 다른 글로벌 고객을 지원하는 데 유용함



## 자연어 처리(NLP)

### • 문의 분류 및 라우팅

- AI는 고객의 질문을 분석하고, 이를 적절한 부서나 상담사에게 자동으로 전달

### • 감정분석

- NLP를 통해 고객의 감정을 분석하여 상담사가 긴급하게 대응해야 할 상황을 식별가능

## 그 외 적용사례

### • 고객 응답 예측

고객의 문의에 대한 적절한 답변을 제안

### • 고객 행동 분석

고객 피드백을 분석하여 제품이나 서비스 개선에  
필요한 인사이트를 제공

### • 콜센터 통화 분석:

AI를 통해 고객과의 통화를 실시간으로 분석하고,  
중요한 정보를 추출하거나 고객 만족도를 평가

### • 자동화된 음성 지원 시스템:

고객이 음성으로 문의하면 AI가 이를 인식하여  
적절한 정보를 제공

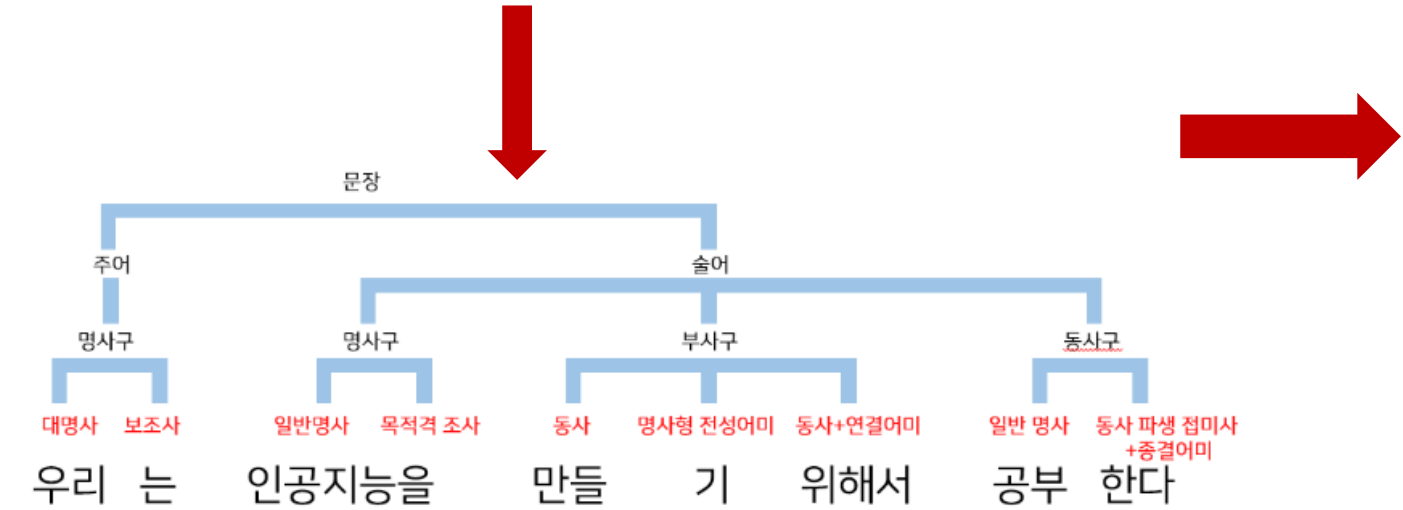
# NLP(Natural Language Processing)\_#자연어처리

**자연어(Natural Language):** 사람이 일상적으로 사용하는 언어  
**처리(Processing):** 컴퓨터가 이 언어를 이해하고 처리하는 과정  
↳ 즉. 자연어처리란 컴퓨터가 사람의 말을 이해하고, 해석하고, 처리하는 기술

## ▶ 자연어처리의 작동 방식

- 형태소 분석: 문장을 단어로 분해하는 과정
- 구문분석: 단어들 간의 관계를 파악
- 벡터공간: 의미를 분석(문장의 특징을 수학적인 처리 및 코사인유사도를 통해 문장을 비교
- 문장 내 품사 분석
- 단어 필터링
- 문서를 단어 벡터로 변환
- 단어벡터 가중치
- 문서 분류

우리는 인공지능을 만들기 위해서 공부한다.  
우리/는/인공지능/을/만들/기/위해서/공부/한다.



예) N=3  
우리는 / 리는□ / 는□인 / □인공 / 인공지 / 공지능 / 지능을 / 능을□ / 을□만 / ...



# 텍스트 기반 다중 감정 분석 주요기능

## AiHub 감정 분류 대화 음성 데이터셋 활용

- 60가지의 세부 감정을 다루는 자연어 처리 말뭉치 포함

- 데이터 종류:음성 데이터(약 10,000개의 문장)/텍스트(약 27만 개의 문장)

## AI 모델 학습

KoBERT를 기반으로 파인튜닝을 통해 감정 분류 모델 학습

1

2

3

4

## 데이터 전처리

감정 레이블링된 텍스트로수집된 데이터를 전처리 (감정 분석의 기초 데이터)

## 감정 분류 및 예측(결과 시각화)

분석 결과를 실시간으로 시각화하여 사용자에게 제공  
(직관적인 그래프로 감정 변화를 표현)

# \* 개발 환경 및 기술 스택

## 1. NLP(자연어 처리) 관련 패키지

- transformers: Hugging Face의 트랜스포머 모델 라이브러리.
- kobert-tokenizer: SKT의 KoBERT 모델용 토큰나이저.
- kobert-transformers: KoBERT 모델을 Transformers와 함께 사용하기 위한 패키지.
- gluonnlp: NLP 도구 모음.
- sacremoses: 텍스트 전처리 라이브러리.
- sentencepiece: 텍스트를 토큰화하는 라이브러리.
- tokenizers: Hugging Face의 고성능 토큰나이저 라이브러리.

## 2. 데이터 과학 및 머신러닝 관련 패키지

- numpy: 다차원 배열 객체를 지원하는 과학 컴퓨팅 라이브러리
- pandas: 데이터프레임을 다루는 데이터 분석 라이브러리
- scikit-learn: 머신러닝을 위한 라이브러리
- scipy: 과학 및 기술 계산 라이브러리
- torch: PyTorch 딥러닝 프레임워크
- mxnet: 딥러닝 프레임워크
- onnxruntime: ONNX 모델 실행을 위한 런타임 라이브러리



# \* 개발 환경 및 기술 스택

## 3. 웹 개발 관련 패키지

- Flask: 경량 웹 애플리케이션 프레임워크
- Jinja2: Flask와 함께 사용하는 템플릿 엔진
- Werkzeug: Flask의 WSGI 유틸리티 라이브러리
- itsdangerous: 안전한 데이터 처리를 위한 라이브러리
- python-dotenv: 환경 변수 관리를 위한 라이브러리

## 5. 유틸리티 및 기타 패키지

- certifi: SSL 인증서를 관리하는 라이브러리
- chardet, charset-normalizer: 문자 인코딩 감지 라이브러리
- click: 명령 줄 인터페이스(CLI) 도구 생성을 위한 라이브러리
- colorama: 콘솔에 색상 출력을 지원하는 라이브러리
- Cython: Python과 C를 결합하여 성능을 높이는 라이브러리
- decorator: 데코레이터 패턴을 쉽게 구현할 수 있는 라이브러리
- dill: 객체 직렬화/역직렬화 라이브러리
- filelock: 파일 잠금 관리 라이브러리
- psutil: 시스템 프로세스 및 리소스 모니터링 라이브러리
- regex: 정규 표현식 라이브러리
- requests: HTTP 요청을 쉽게 보낼 수 있는 라이브러리
- sympy: 기호 수학 연산을 위한 라이브러리

## 4. 시각화 및 UI 관련 패키지

- matplotlib: 데이터 시각화를 위한 라이브러리
- contourpy: 등고선을 그리기 위한 라이브러리
- cyclert: matplotlib의 색상 및 스타일 순환을 위한 라이브러리
- fonttools: 폰트 파일을 다루는 도구
- pillow: 이미지 처리 라이브러리
- ipywidgets: Jupyter 노트북에서 대화형 위젯을 제공하는 라이브러리
- jupyterlab\_widgets: JupyterLab에서 위젯을 사용할 수 있게 해주는 확장

## 6. 개발 도구 및 환경 관리 패키지

- accelerate: 대규모 딥러닝 모델 학습 및 배포를 가속화하는 라이브러리
- pip: Python 패키지 설치 및 관리 도구
- setuptools: Python 패키지의 설치, 업그레이드 등을 돕는 도구
- importlib\_metadata,  
importlib\_resources : Python의 importlib 모듈을 보완하는 라이브러리

# Train Model

1

## BERT

구글에서 개발한 자연어 처리 모델  
문맥 이해와 다양한 NLP 작업에서 뛰어난 성능을 가지고 있음  
그러나 BERT는 주로 영어 데이터를 기반으로 학습되어 있어  
한국어 텍스트에 대한 성능이 제한적

2

## KoBERT

SKTBrain에서 개발한 한국어 특화 BERT 모델  
약 5백만 개의 한국어 위키피디아 문장과  
약 2천만 개의 뉴스 데이터를 학습하여 한국어에 최적화된 모델로 개발

3

## TensorFlow

구글이 개발한 오픈소스 머신러닝 라이브러리  
주로 딥러닝 모델을 만들고 학습시키는 데 사용되며, 다양한 수학적  
연산을 효율적으로 처리할 수 있도록 설계되었습니다.  
TensorFlow는 신경망을 구성하고 훈련시키는 데 필요한 도구들을  
제공

The BERT logo is displayed in a stylized, italicized white font against a dark background. The letters are bold and modern, with a slight shadow effect.The KoBERT logo features the Korean characters '코보트' (KoBot) in a white, blocky font above the word 'KOBERT' in a similar white, blocky font. The entire logo is set against a dark background.

# data\_processing.py

```
# 감정 라벨 매핑 (한국어 명칭에 맞춤)
emotion_mapping = {
    '분노': 0,
    '기쁨': 1,
    '당황': 2,
    '불안': 3,
    '슬픔': 4,
    '상처': 5
}

# 텍스트 정제 함수
def clean_text(text):
    # 기본적인 정제 (소문자 변환, 불필요한 특수문자 제거 등)
    text = text.lower().strip()
    return text

# JSON 데이터 로드 및 전처리 함수 (필요에 따라 수정 가능)
def load_json_data(directory):
    data = {
        'Text': [], # 텍스트 데이터를 저장할 리스트
        'Emotion': [] # 감정 라벨을 저장할 리스트
    }

    for filename in os.listdir(directory):
        if filename.endswith('.json'): # 확장자가 .json인 파일만 처리
            with open(os.path.join(directory, filename), 'r', encoding='utf-8') as f:
                json_data = json.load(f) # JSON 파일을 로드
                for entry in json_data: # JSON의 각 엔트리(데이터) 처리
                    emotion_type = entry['profile']['emotion']['type'] # 감정 유형 추출
                    if emotion_type in emotion_mapping: # 감정 유형이 매핑된 경우에만 처리
                        label = emotion_mapping[emotion_type] # 해당 감정 유형의 라벨을 가져옴
                        talk_content = entry['talk']['content'] # 대화 내용을 가져옴
                        for key in ['HS01', 'HS02', 'HS03']: # 각 대화 내용 필드를 순회
                            if talk_content[key]: # 텍스트가 비어 있지 않으면
                                cleaned_text = clean_text(talk_content[key]) # 텍스트 정제
                                data['Text'].append(cleaned_text) # 텍스트 추가
                                data['Emotion'].append(label) # 라벨 추가

    return pd.DataFrame(data) # DataFrame 형태로 반환
```

```
# 엑셀 데이터 로드 및 전처리 함수
def load_excel_data(file_path):
    df = pd.read_excel(file_path) # 엑셀 파일 로드

    # '감정_대분류'를 사용하여 감정 라벨을 매핑
    df['Emotion'] = df['감정_대분류'].map(emotion_mapping) # 감정 라벨 매핑

    data = {
        'Text': [], # 텍스트 데이터를 저장할 리스트
        'Emotion': [] # 감정 라벨을 저장할 리스트
    }

    for index, row in df.iterrows():
        for col in ['사람문장1', '사람문장2', '사람문장3']: # 각 텍스트 열을 순회
            if pd.notna(row[col]): # 텍스트가 존재하는 경우만 추가
                cleaned_text = clean_text(row[col]) # 텍스트 정제
                data['Text'].append(cleaned_text) # 텍스트 추가
                data['Emotion'].append(row['Emotion']) # 라벨 추가

    final_df = pd.DataFrame(data).dropna() # DataFrame으로 변환 후 NaN 값 제거
    return final_df

# 데이터 불균형 처리 (업샘플링)
def balance_dataset(df):
    balanced_df = pd.DataFrame()

    for label in df['Emotion'].unique():
        label_df = df[df['Emotion'] == label]
        if len(label_df) < 1000: # 예시: 최소 1000개 이상으로 업샘플링
            label_df = resample(label_df, replace=True, n_samples=1000, random_state=42)
        balanced_df = pd.concat([balanced_df, label_df])

    return balanced_df.sample(frac=1, random_state=42).reset_index(drop=True) # 셔플 및 인덱스 리셋
```

# train\_model.py

```
# 설정
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
max_len = 32
batch_size = 32 # 배치 크기를 64로 조정하여 메모리 효율을 높임
num_epochs = 3
learning_rate = 3e-5 # 학습률을 높여서 초기 학습 속도를 향상시킴
fp16 = True if torch.cuda.is_available() else False

# 데이터셋 클래스 정의
class EmotionDataset(Dataset):
    def __init__(self, encodings, labels):
        self.encodings = {key: torch.tensor(val) for key, val in encodings.items()}
        self.labels = torch.tensor(labels).long()

    def __len__(self):
        return len(self.labels)

    def __getitem__(self, idx):
        item = {key: val[idx] for key, val in self.encodings.items()}
        item['labels'] = self.labels[idx]
        return item

# 데이터 로드 및 크기 줄이기 (훈련 데이터의 50%만 사용하여 학습 시간 단축)
train_df = pd.read_csv('./data/train.csv').sample(frac=0.5)
val_df = pd.read_csv('./data/validation.csv')

# KoBERT 토큰라이저 로드 (BPE-dropout 적용)
sp_model_kwargs = {'nbest_size': -1, 'alpha': 0.6, 'enable_sampling': True}
tokenizer = KoBERTTokenizer.from_pretrained('skt/kobert-base-v1', sp_model_kwargs=sp_model_kwargs)

# 모델 로드
model = BertForSequenceClassification.from_pretrained("skt/kobert-base-v1", num_labels=6)
model.to(device)

# 데이터셋 준비
train_encodings = tokenizer(train_df['Text'].tolist(), truncation=True, padding=True, max_length=max_len, return_tensors="pt")
val_encodings = tokenizer(val_df['Text'].tolist(), truncation=True, padding=True, max_length=max_len, return_tensors="pt")

train_dataset = EmotionDataset(train_encodings, train_df['Emotion'].tolist())
val_dataset = EmotionDataset(val_encodings, val_df['Emotion'].tolist())

# DataLoader 사용하여 데이터 로딩 속도 개선
train_loader = DataLoader(train_dataset, batch_size=batch_size, shuffle=True, num_workers=4)
val_loader = DataLoader(val_dataset, batch_size=batch_size, shuffle=False, num_workers=4)
```

# model.py

```
src > model.py > predict_emotions
1  import torch
2  from transformers import BertForSequenceClassification
3  from kobert_tokenizer import KoBERTTokenizer
4
5  def load_model():
6      model = BertForSequenceClassification.from_pretrained('./models/best_model', num_labels=6)
7      tokenizer = KoBERTTokenizer.from_pretrained('./models/best_model')
8      return model, tokenizer
9
10 def predict_emotions(sentence, model, tokenizer):
11     inputs = tokenizer.encode_plus(
12         sentence,
13         return_tensors="pt",
14         truncation=True,
15         padding=True,
16         max_length=32
17     )
18     input_ids = inputs['input_ids']
19     attention_mask = inputs['attention_mask']
20
21     with torch.no_grad():
22         outputs = model(input_ids, attention_mask=attention_mask)
23         logits = outputs.logits
24         probabilities = torch.softmax(logits, dim=-1).flatten().tolist()
25
26     return probabilities
```



```

sentences = eval(sentences[0]) # ast.literal_eval 대신 eval 사용

# 문장을 하나로 통합하여 감정을 분석
combined_sentence = " ".join(sentences)

# 감정 예측
probabilities = predict_emotions(combined_sentence, model, tokenizer)
logging.info(f"Combined sentence: {combined_sentence}")
logging.info(f"Predicted probabilities: {probabilities}")

emotions = ["분노", "기쁨", "당황", "불안", "슬픔", "상처"]

# Jinja2에서 min 함수 사용 문제 해결
emotion_dict = {emotions[j]: min(probabilities[j] * 100, 100) for j in range(len(probabilities))}

# 가장 높은 감정을 찾고 해당하는 캐릭터 이미지 선택
dominant_emotion = max(emotion_dict, key=emotion_dict.get)
emotion_image = emotion_images[dominant_emotion]
emotion_message = emotion_messages[dominant_emotion]

logging.info(f"Emotion dict: {emotion_dict}")
logging.info(f"Dominant emotion: {dominant_emotion}, Emotion image: {emotion_image}, Message: {emotion_message}")

# 그래프 생성
fig, ax = plt.subplots()
ax.barh(list(emotion_dict.keys()), list(emotion_dict.values()), color=['#FF9999', '#66B2FF', '#99FF99', '#FFCC99', '#FFD700', '#FF6347'])
ax.set_xlim(0, 100) # x축 최대 범위를 100으로 설정
ax.set_xlabel('Total Probability')
ax.set_title("Overall Emotion Analysis")
plt.tight_layout()

# 그래프 파일 저장
graph_filename = f"static/overall_emotion_graph.png"
plt.savefig(graph_filename)
plt.close()

```

# 시연\_이미지

오늘 하루는 어땠어?

하고싶은 말을 입력해주세요

입력

Emotion Analysis

하고싶은 말을 입력해주세요

입력

입력된 문장:

짜증나

화나

열받아

감정 분석

오늘의 기분은..



분노

기쁨

당황

불안

슬픔

상처

최종 감정 분석

기분전환을 위해 산책을 나가보세요.

다시 분석하기

# 상담사에게 실시간 코칭

## 대응 전략 제안

AI가 고객의 감정 상태에 맞는  
최적의 대응 전략을 제안하면  
상담사는 의사결정으로 응대 가능

## 감정 완화 표현 추천

고객의 부정적 감정을 완화할 수  
있는 적절한 표현을 추천.  
학습을 통해 공감과 이해를 표현  
하는 문구를 상담사에게 제시

## 실시간 피드백

상담사의 대응에 대한  
실시간 피드백을 제공  
통화 내용을 텍스트로 변환하고  
감정변화를 상담사 및 시스템에  
전달함으로써 더 나은 대화 방향을  
제시가능하도록 지원

# 감정 상태 시각화

## 실시간 대시보드 활용

## 고객의 감정 상태를 색상 코드로 시각화 처리

- 직관적인 UI로 상담사의 빠른 인지가 가능

## 감정 변화 그래프

시간에 따른 감정 변화를 그래프로 표시

- 상담 진행 상황에 따른 고객의 감정변화를 모니터링 가능

## 주요 감정 키워드

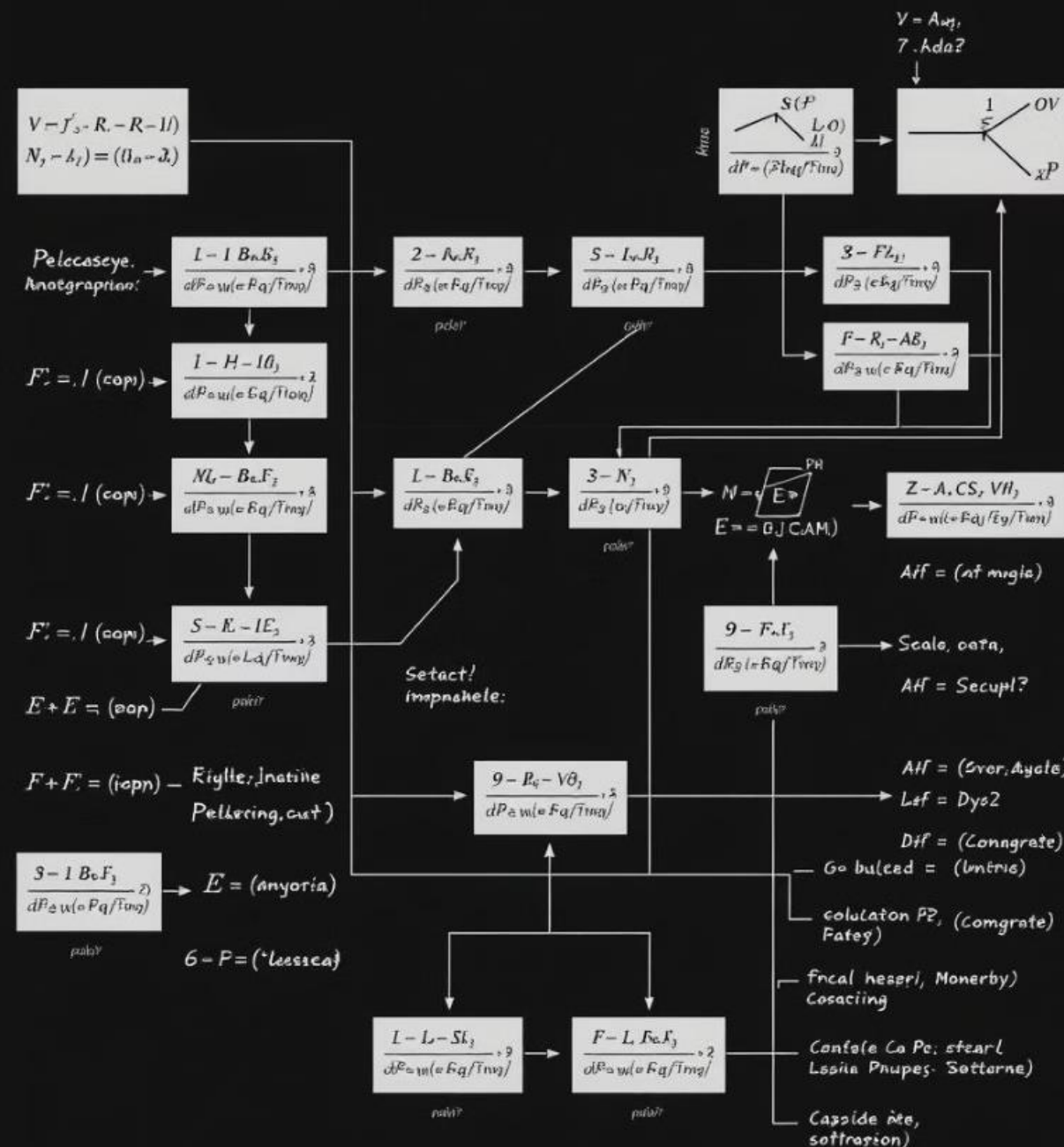
감정과 관련된 주요 키워드를 시각적으로 강조

- 중요한 감정 포인트를 쉽게 인식하여 대처가 가능하도록 유도

## 경고 알림 시스템

감정 상태가 급격히 악화될 경우 경고 알림을 제공

- 상담사가 즉각 대응 가능



# 실시간 대화 기록

1

## 음성 인식

고객과 상담사의 대화를 실시간으로 음성 인식

2

## 텍스트 변환

인식된 음성을 텍스트로 변환합니다. 빠른 속도와 높은 정확도로 실시간 기록이 가능

3

## 대화 내용 분석

변환된 텍스트를 AI가 분석하고 주요 키워드와 문맥을 파악하여 중요 정보를 추출함

4

## 데이터베이스 저장

분석된 대화 내용을 데이터베이스에 저장하여 추후 분석과 보고서 작성에 활용



# 대화 요약 및 분석



## 주요 이슈 식별

AI가 대화 내용에서 주요 이슈를 자동으로 식별하여 고객의 핵심 요구사항을 파악가능하도록 함



## 요약 생성

복잡한 대화 내용을 간결하게 요약처리  
- 상담사가 응대 업무 시 핵심을 빠른 파악이 가능



## 감정-내용 연계 분석

대화 내용과 감정 변화를 연계하여 분석  
- 특정 발언이 감정에 미치는 영향을 파악



## 인사이트 도출

분석 결과를 바탕으로 유용한 인사이트를 도출  
- 향후 상담 개선에 활용

# 상담 종료 후 보고서 작성 지원

1

## 데이터 수집

상담 중 수집된 모든 데이터를 종합하여 대화 내용, 감정 분석 결과, 주요 이슈 등을 파악

2

## AI 분석

수집된 데이터를 AI가 심층 분석  
- 상담의 전체적인 흐름과 결과를 평가

3

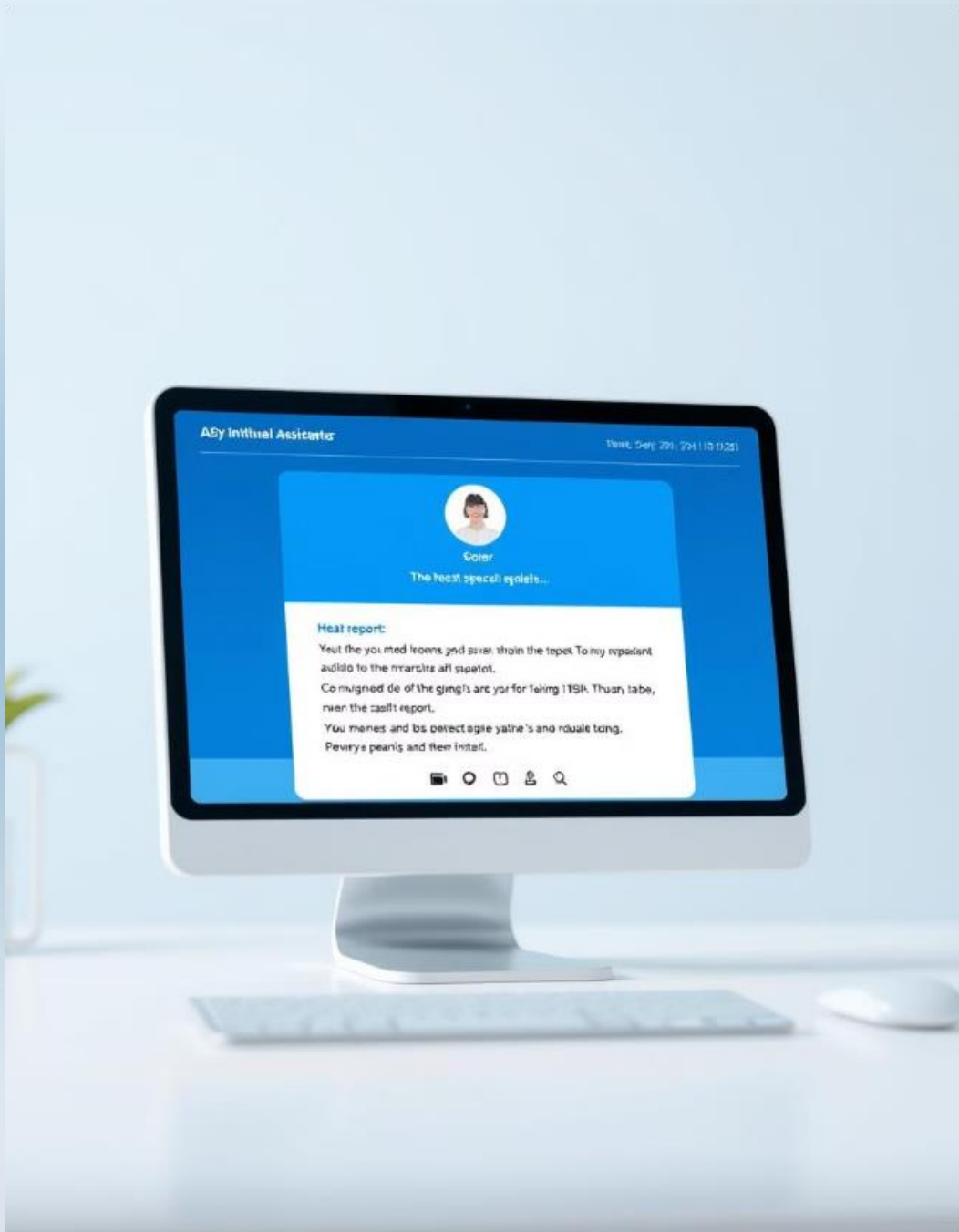
## 보고서 초안 생성

AI가 분석 결과를 바탕으로 보고서 초안을 자동으로 생성  
- 객관적이고 구조화된 보고서 작성이 가능

4

## 상담사 검토 및 수정

생성된 초안을 상담사가 검토하고 필요 시 수정 진행  
- 상담사의 경력 및 스킬에 따라 내용을 보완 가능하도록하여 최종 보고서를 완성



# 주요 기능



## 다기능 대시보드

감정 분석, 대화 내용, AI 제안 등을 한 화면에서 확인할 수 있는 통합 대시보드를 제공



## 보고서 작성 툴

AI가 생성한 초안을 바탕으로 상담사가 보고서를 쉽게 작성할 수 있는 인터랙티브 도구를 제공



# 음성 인식 및 감정 분석 기술

1

## Google Cloud Speech-to-Text API

고품질의 음성 인식 서비스를 제공하는 API  
(실시간으로 음성을 텍스트로 변환)

2

## Hugging Face Transformers

최신 NLP 모델을 쉽게 사용가능  
(\* BERT, GPT 등 다양한 모델을 활용)

3

## Custom Emotion Analysis Model

해당 기능에 특화된 감정 분석 모델을 개발  
- 업무의 특성에 맞게 최적화된 성능을 제공

4

## Real-time Processing Pipeline

음성 인식부터 감정 분석까지 실시간으로 처리하는 파이프라인을 구축

**Thank you**