Fashion-How 발표자료

Sub-Task 3

박상하

CONTENTS

1. 문제 정의

문제 정의

과제 목표

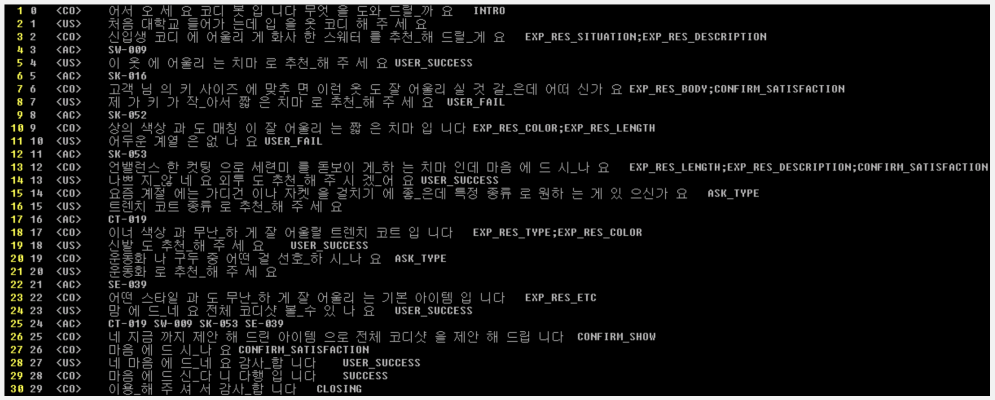
1. 문제 정의 및 과제 목표

1. 문제 정의
    1. 사용자의 발화에 따라 의상을 추천하는 방법을 학습하여 적합한 의상을 선택하도록 한다.
2. 과제 목표
    1. **사용자와 AI 코디네이터 간의 대화, 패션 추천 조합들이 입력이 되면 조합들의 순위를 매긴다.**
    2. **주어진 데이터를 순차적으로 학습 및 평가를 진행하면서 파괴적 망각 현상을 완화한다.**

2. 데이터 확인

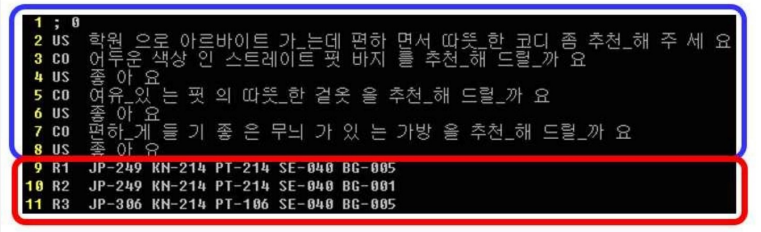
1. 데이터 확인

1. 학습 및 평가 데이터 확인



```
 1 0  <CO>  어서 오 세 요 코디 봇 입 니 다 무엇 을 도와 드릴_까 요  INTRO
 2 1  <US>  처음 대학교 들어가 는데 입 을 옷 코디 해 주 세 요
 3 2  <CO>  신입생 코디 에 어울리 게 화사 한 스웨터 를 추천_해 드릴_게 요  EXP_RES_SITUATION;EXP_RES_DESCRIPTION
 4 3  <AC>  SW-009
 5 4  <US>  이 옷 에 어울리 는 치마 로 추천_해 주 세 요 USER_SUCCESS
 6 5  <AC>  SK-016
 7 6  <CO>  고객 님 의 키 사이즈 에 맞추 면 이런 옷 도 잘 어울리 실 것 같_은데 어떠 신가 요 EXP_RES_BODY;CONFIRM_SATISFACTION
 8 7  <US>  제 가 키 가 작_아서 짧 은 치마 로 추천_해 주 세 요  USER_FAIL
 9 8  <AC>  SK-052
10 9  <CO>  상의 색상 과 도 매칭 이 잘 어울리 는 짧 은 치마 입 니 다 EXP_RES_COLOR;EXP_RES_LENGTH
11 10 <US>  어두운 계열 은 없 나 요 USER_FAIL
12 11 <AC>  SK-053
13 12 <CO>  언발런스 한 컷팅 으로 세련미 를 돋보이 게_하 는 치마 인데 마음 에 드 시_나 요  EXP_RES_LENGTH;EXP_RES_DESCRIPTION;CONFIRM_SATISFACTION
14 13 <US>  나쁘 지_않 네 요 외투 도 추천_해 주 시 겠_어 요 USER_SUCCESS
15 14 <CO>  요즘 계절 에는 가디건 이나 자켓 을 걸치기 에 좋_은데 특정 종류 로 원하 는 게 있 으신가 요  ASK_TYPE
16 15 <US>  트렌치 코트 종류 로 추천_해 주 세 요
17 16 <AC>  CT-019
18 17 <CO>  이너 색상 과 무난_하 게 잘 어울릴 트렌치 코트 입 니 다  EXP_RES_TYPE;EXP_RES_COLOR
19 18 <US>  신발 도 추천_해 주 세 요  USER_SUCCESS
20 19 <CO>  운동화 나 구두 중 어떤 걸 선호_하 시_나 요  ASK_TYPE
21 20 <US>  운동화 로 추천_해 주 세 요
22 21 <AC>  SE-039
23 22 <CO>  어떤 스타일 과 도 무난_하 게 잘 어울리 는 기본 아이템 입 니 다  EXP_RES_ETC
24 23 <US>  맘 에 드_네 요 전체 코디샷 볼_수 있 나 요  USER_SUCCESS
25 24 <AC>  CT-019 SW-009 SK-053 SE-039
26 25 <CO>  네 지금 까지 제안 해 드린 아이템 으로 전체 코디샷 을 제안 해 드립 니 다  CONFIRM_SHOW
27 26 <CO>  마음 에 드 시_나 요 CONFIRM_SATISFACTION
28 27 <US>  네 마음 에 드_네 요 감사_합 니 다  USER_SUCCESS
29 28 <CO>  마음 에 드 신_다 니 다행 입 니 다  SUCCESS
30 29 <CO>  이용_해 주 셔 서 감사_합 니 다  CLOSING
```

Sub-Task 3 학습 데이터



```
 1 ; 0
 2 US  학원 으로 아르바이트 가_는데 편하 면서 따뜻_한 코디 좀 추천_해 주 세 요
 3 CO  어두운 색상 인 스트레이트 핏 바지 를 추천_해 드릴_까 요
 4 US  좋 아 요
 5 CO  여유_있 는 핏 의 따뜻_한 겉옷 을 추천_해 드릴_까 요
 6 US  좋 아 요
 7 CO  편하_게 들 기 좋 은 무늬 가 있 는 가방 을 추천_해 드릴_까 요
 8 US  좋 아 요
 9 R1  JP-249 KN-214 PT-214 SE-040 BG-005
10 R2  JP-249 KN-214 PT-214 SE-040 BG-001
11 R3  JP-306 KN-214 PT-106 SE-040 BG-005
```

Sub-Task 3 평가 데이터

1. 학습 및 평가 데이터가 일치하지 않고 상당한 차이가 존재하는 것을 확인
2. **학습 데이터를 전처리를 통해서 평가 데이터와 유사하게 만들어줘야 한다는 것**을 파악

2. 메타 데이터 확인

CD-001	0	CD	F	상의 스트레이트 실루엣
CD-001	0	CD	F	상의 엉덩이 덮 는 길이
CD-001	0	CD	F	상의 일자형 밑단
CD-001	0	CD	F	상의 브이 네크라인
CD-001	0	CD	F	상의 드랍 숄더
CD-001	0	CD	F	상의 일자형 소매
CD-001	0	CD	F	상의 긴 소매 길이
CD-001	0	CD	F	상의 일자형 소매단
CD-001	0	CD	F	상의 허리선 부분 양쪽 패치 포켓
CD-001	0	CD	M	폴리에스터 혼방
CD-001	0	CD	M	네오프렌
CD-001	0	CD	M	도톰 한 소재
CD-001	0	CD	C	핑크 파스텔 톤 단색
CD-001	0	CD	E	데이트
CD-001	0	CD	E	러블리
CD-001	0	CD	E	귀여운
CD-001	0	CD	E	단조로운
CD-002	0	CD	F	상의 박시 실루엣
CD-002	0	CD	F	상의 긴 길이
CD-002	0	CD	F	상의 옆 직선 밑단 트임

Sub-Task 3 메타 데이터

1. 각 옷에 따른 패션 항목, 종류, 특징 등이 저장되어있다.
2. **하나의 옷에 여러 특징이 존재하는 것을 확인**

3. 데이터 전처리

1. 데이터 전처리
2. 데이터 증강

1. 대화 데이터 불러오기

```
['안녕_하 세 요 코디 봇 입 니다 무엇 을 도와 드릴_까 요',
 '안녕_하 세 요 꽃 축제 가_려고 하_는데 옷 추천_해 주 세 요',
 '선선 한 날씨 에 대비 하_여 점퍼 가 포함_된 코디 를 보여 드릴_까 요',
 '네 겉옷 이 있_으면 좋 겠_네 요',
 '베이지 색상 의 겉옷 이 포함_된 코디 세트 입 니다 마음 에 드 시_나 요',
 '상의 가 너무 화려 합 니다 흰색 톤 으로 추천 부탁 드려_요',
 '밝 은 색상 의 블라우스 는 어떠 신가 요',
 '마음 에 듭_니다',
 '이 블라우스 로 변경 된 전체 코디 세트 를 보여 드릴_까 요',
 '네',
 '흰색 의 블라우스 와 점퍼 가 포함_된 코디 셋트 입니다 어떠 신가 요',
 '화려_하 면서_도 겉옷 이 있_어서 좋 습니다',
 '다만 신발 이 볼편_해 보여 서 편한 운동화 로 추천_해 주 세 요',
 '운동화 는 어떠 신가 요',
 '네 운동화 로 추천_해 주 세 요',
 '앞 부분 에 포인트 가 있 는 캐주얼 한 느낌 의 운동화 는 어떠_세 요',
 '신발 에 포인트 가 있_어서 좋 아 요',
 '선택_한 아이템 을 이용_한 최종 코디 셋트 입 니다 마음 에 드 시_나 요',
 '네 지금 제일 마음 에 드_네 요 이렇게 입_고 가_면 좋 겠_어 요',
 '마음 에 드 셨_다 니 기쁩 니다',
 '이용_해 주 셔 서 감사_합 니다 코디 봇 을 종료 합 니다']
```

사용자와 코디네이터 간의 대화

```
[{0: 'JP-345', 1: 'BL-151', 2: 'PT-418', 3: 'SE-097'},
 {0: 'JP-345', 1: 'BL-234', 2: 'PT-418', 3: 'SE-097'},
 {0: 'JP-345', 1: 'BL-234', 2: 'PT-418', 3: 'SE-097'},
 {0: 'JP-345', 1: 'BL-234', 2: 'PT-418', 3: 'SE-097'},
 {0: 'JP-345', 1: 'BL-234', 2: 'PT-418', 3: 'SE-100'},
 {0: 'JP-345', 1: 'BL-234', 2: 'PT-418', 3: 'SE-100'}]
```

코디 데이터

```
['USER_SUCCESS_PART',
 'USER_SUCCESS',
 'USER_SUCCESS',
 'USER_FAIL',
 'USER_SUCCESS_PART',
 'USER_SUCCESS']
```

사용자 평가 데이터

1. 대화 내에서 **코디네이터의 추천 및 사용자 평가는 제외하여 순수 대화**로 전처리
2. 각 추천에 따른 사용자의 평가를 따로 정리

2. 메타 데이터 불러오기

```
Outer
[('CD-001', 0), ('CD-002', 1), ('CD-003', 2), ('CD-004', 3), ('CD-005', 4)]
--------------------------------------------------------------------------------
Top
[('BL-001', 0), ('BL-002', 1), ('BL-003', 2), ('BL-004', 3), ('BL-005', 4)]
--------------------------------------------------------------------------------
Bottom
[('OP-001', 0), ('OP-002', 1), ('OP-003', 2), ('OP-004', 3), ('OP-005', 4)]
--------------------------------------------------------------------------------
Shoes
[('SE-001', 0), ('SE-002', 1), ('SE-003', 2), ('SE-004', 3), ('SE-005', 4)]
--------------------------------------------------------------------------------
```

옷의 분류에 따른 옷 → 인덱스 딕셔너리

```
Outer
[(0, 'CD-001'), (1, 'CD-002'), (2, 'CD-003'), (3, 'CD-004'), (4, 'CD-005')]
--------------------------------------------------------------------------------
Top
[(0, 'BL-001'), (1, 'BL-002'), (2, 'BL-003'), (3, 'BL-004'), (4, 'BL-005')]
--------------------------------------------------------------------------------
Bottom
[(0, 'OP-001'), (1, 'OP-002'), (2, 'OP-003'), (3, 'OP-004'), (4, 'OP-005')]
--------------------------------------------------------------------------------
Shoes
[(0, 'SE-001'), (1, 'SE-002'), (2, 'SE-003'), (3, 'SE-004'), (4, 'SE-005')]
--------------------------------------------------------------------------------
```

옷의 분류에 따른 인덱스 → 옷 딕셔너리

```
img2description = {m["name"] : m["fashion_description"] for m in mdataset}
img_list = list(img2description.keys())
img_category = collections.defaultdict(list)

for img in img_list :
    img_categorty = position_of_fashion_item(img)
    img_category[img_categorty].append(img)

img_vectors = collections.defaultdict(dict) # img to vector
for i in img_category :
    sub_img_list = img_category[i]

    for k in sub_img_list :
        desc = img2description[k]
        embed = [self.swer.get_sent_emb(d) for d in desc]
        vector = np.mean(embed, axis=0)
        img_vectors[i][k] = vector

img_similarity = collections.defaultdict(list) # img to similarity

for i in img_vectors :
    vectors = list(img_vectors[i].values())
    array = np.array(vectors)
    for j in range(len(vectors)) :
        org_vector = vectors[j]
        org_vector = np.expand_dims(org_vector, axis=0)
        similarity = cosine_similarity(org_vector, array)
        img_similarity[i].append(similarity[0])
```

옷 간의 유사도 계산하는 코드

1. 이미지를 Outer, Top, Bottom, Shoes 4개의 카테고리로 분류하고 인덱스로 변환할 수 있는 딕셔너리를 생성 (역 관계의 딕셔너리도 생성)
2. **옷에 대한 각 설명을 SubWordEmbReaderUtil을 통해서 벡터로 변환하고 그 벡터들의 평균 값**을 구한다.
3. 2번 과정을 통해서 구한 **벡터들을 기반으로 카테고리 내에서 옷들의 유사도**를 계산한다.

3. 코디 데이터 전처리

```
[{0: 'JP-345', 1: 'BL-151', 2: 'PT-418', 3: 'SE-097'},
 {0: 'JP-345', 1: 'BL-234', 2: 'PT-418', 3: 'SE-097'},
 {0: 'JP-345', 1: 'BL-234', 2: 'PT-418', 3: 'SE-097'},
 {0: 'JP-345', 1: 'BL-234', 2: 'PT-418', 3: 'SE-097'},
 {0: 'JP-345', 1: 'BL-234', 2: 'PT-418', 3: 'SE-100'},
 {0: 'JP-345', 1: 'BL-234', 2: 'PT-418', 3: 'SE-100'}]
```

전처리 Case 1

```
[{0: 'JP-345', 1: 'BL-234', 2: 'PT-418', 3: 'SE-100'},
 {0: 'JP-345', 1: 'BL-234', 2: 'PT-418', 3: 'SE-097'},
 {0: 'JP-345', 1: 'BL-151', 2: 'PT-418', 3: 'SE-097'}]
```

```
[{2: 'OP-030', 3: 'SE-237', 0: 'NONE-OUTER', 1: 'NONE-TOP'},
 {2: 'OP-107', 3: 'SE-232', 0: 'CD-165', 1: 'NONE-TOP'},
 {2: 'OP-139', 3: 'SE-175', 0: 'CD-165', 1: 'NONE-TOP'},
 {2: 'OP-050', 3: 'SE-175', 0: 'CD-165', 1: 'NONE-TOP'}]
```

전처리 Case 2

```
[{2: 'OP-050', 3: 'SE-175', 0: 'CD-165', 1: 'NONE-TOP'},
 {2: 'OP-139', 3: 'SE-175', 0: 'CD-165', 1: 'NONE-TOP'},
 {2: 'OP-107', 3: 'SE-232', 0: 'CD-165', 1: 'NONE-TOP'}]
```

```
[{0: 'CD-043', 1: 'BL-151', 2: 'PT-432', 3: 'SE-158'},
 {0: 'CD-043', 1: 'SH-157', 2: 'PT-199', 3: 'SE-158'}]
```

전처리 Case 3

```
[{0: 'CD-043', 1: 'SH-157', 2: 'PT-199', 3: 'SE-158'},
 {0: 'CD-043', 1: 'BL-151', 2: 'PT-432', 3: 'SE-158'},
 {0: 'CD-043', 1: 'SH-157', 2: 'PT-199', 3: 'SE-182'}]
```

1. 추천된 코디의 수가 3개 이상이면 뒤에서부터 3개를 선정한다.
2. 추천된 코디에서 4개의 카테고리 중에 없는 것이 있다면 Dummy Label로 채운다.
3. 추천된 코디의 수가 3개 미만이면 있는 코디들 중에서 임의로 선정해서 데이터를 증강한다.
    - **메타데이터를 통해서 구한 옷들 간의 유사도 정보를 활용**
4. 정답 데이터는 '0'으로 설정한다.

4. 데이터 증강

```
diag = d["diag"]
cordi = d["cordi"]
reward = d["reward"]

for j in range(self.num_aug) :
    source_id = np.random.randint(3)
    source = cordi[source_id]

    targets = [k for k in range(4) if "NONE" not in source[k]]
    target_id = random.sample(targets, 1)[0]
    target_img = source[target_id]

    img_id = img2id[target_id][target_img]
    img_sim = img_similarity[target_id][img_id]

    rank_args = np.argsort(img_sim)[::-1][1:]
    select_id = np.random.randint(50)
    select_arg = rank_args[select_id]
    select_img = id2img[target_id][select_arg]

    aug = copy.deepcopy(source)
    aug[target_id] = select_img

    if source_id == 0 :
        data = {"diag" : diag,
            "cordi" : [cordi[0]] + [cordi[np.random.randint(1,3)]] + [aug],
            "reward" : reward
        }
    else :
        data = {"diag" : diag,
            "cordi" : cordi[:source_id] + [aug] + cordi[source_id+1:],
            "reward" : reward
        }
    aug_dataset.append(data)
```

데이터 증강 코드

1. 3개의 추천 중 하나를 선정한다.
2. 추천된 4가지 옷들 중에서 Dummy Label이 아닌 것들 중 하나를 선정한다.
3. **해당 옷의 카테고리 내에서 그 옷과 가장 유사한 옷 50개 옷들 중에서 임의로 1개의 옷을 선택한다.**
4. 그 옷만 변경한 채로 조합을 만들어서 데이터를 추가한다.

5. 데이터 인코딩

```python
class Encoder :
    def __init__(self, swer, img2id, num_cordi, mem_size) :
        self.swer = swer
        self.img2id = img2id
        self.num_cordi = num_cordi
        self.mem_size = mem_size

    def __call__(self, dataset) :
        diag_list, cordi_list, rank_list = [], [], []
        for data in dataset :
            diag = data["diag"]
            vectors = [self.swer.get_sent_emb(sen).tolist() for sen in diag]
            if len(vectors) >= self.mem_size :
                vectors = vectors[:self.mem_size]
            else :
                vectors = [np.zeros(self.swer.get_emb_size()).tolist() for i in range(self.mem_size - len(vectors))] + vectors

            cordi = data["cordi"]
            cordi_rows = []
            for c in cordi :
                imgs = [self.img2id[j][c[j]] for j in range(self.num_cordi)]
                cordi_rows.append(imgs)

            diag_list.append(vectors)
            cordi_list.append(cordi_rows)
            if "reward" in data :
                rank_list.append(data["reward"])

        if "reward" in dataset[0] :
            dataset = {"diag" : diag_list, "cordi" : cordi_list, "rank" : rank_list}
        else :
            dataset = {"diag" : diag_list, "cordi" : cordi_list}
        return dataset
```

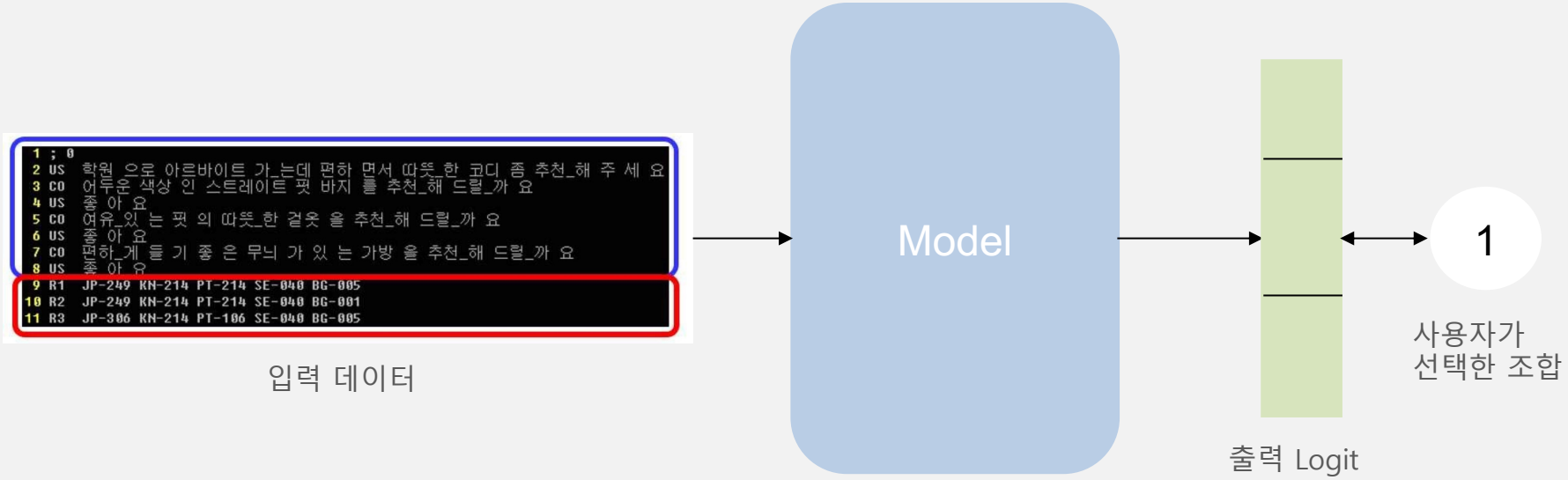
데이터 인코딩 코드

1. 대화를 SubWordEmbReaderUtil을 통해서 벡터로 변환한다.
    - **대화의 문장 수가 mem_size 보다 길면 뒤에서부터 mem_size 만큼 자른다.**
    - **대화의 문장 수가 mem_size 보다 적으면 앞에서부터 zero 벡터로 채운다.**
2. 딕셔너리형태의 코디 데이터를 각각의 옷을 인덱스로 바꾸고 numpy array 타입으로 변환한다.

4. 모델 설계

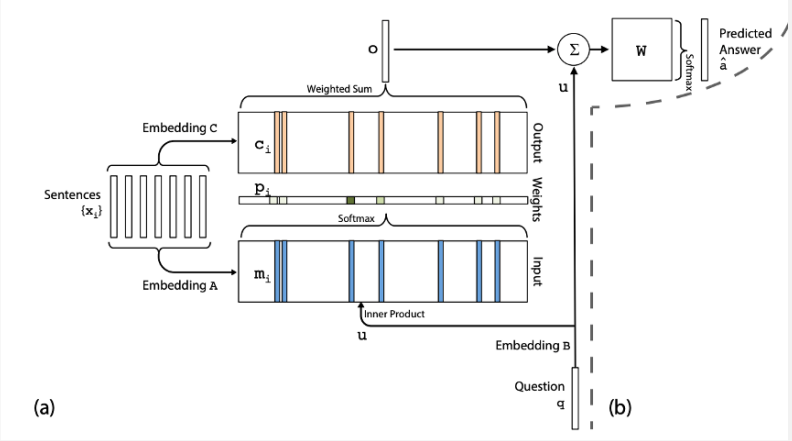
1. 모델 구조 설계

1. 모델의 입출력 설계



1. 모델에는 대화데이터, 추천 옷 조합데이터가 입력되어야 한다.
    - **대화데이터, 추천 옷 조합데이터를 인코딩하는 부분이 각각 필요하다.**
2. 모델은 3개의 Logit 값을 출력한다.
3. **3개의 Logit은 각 조합이 사용자가 좋아하는 정도를 의미하게 된다.**
    - 정답 조합을 해당 순서에 넣어서 추천 옷 조합데이터를 입력하면 된다.
    - 추론을 할 때는 3개의 Logit 값을 정렬해서 해당 추천 옷 조합데이터의 순위를 반환하도록 한다.

2. 대화 데이터 인코더



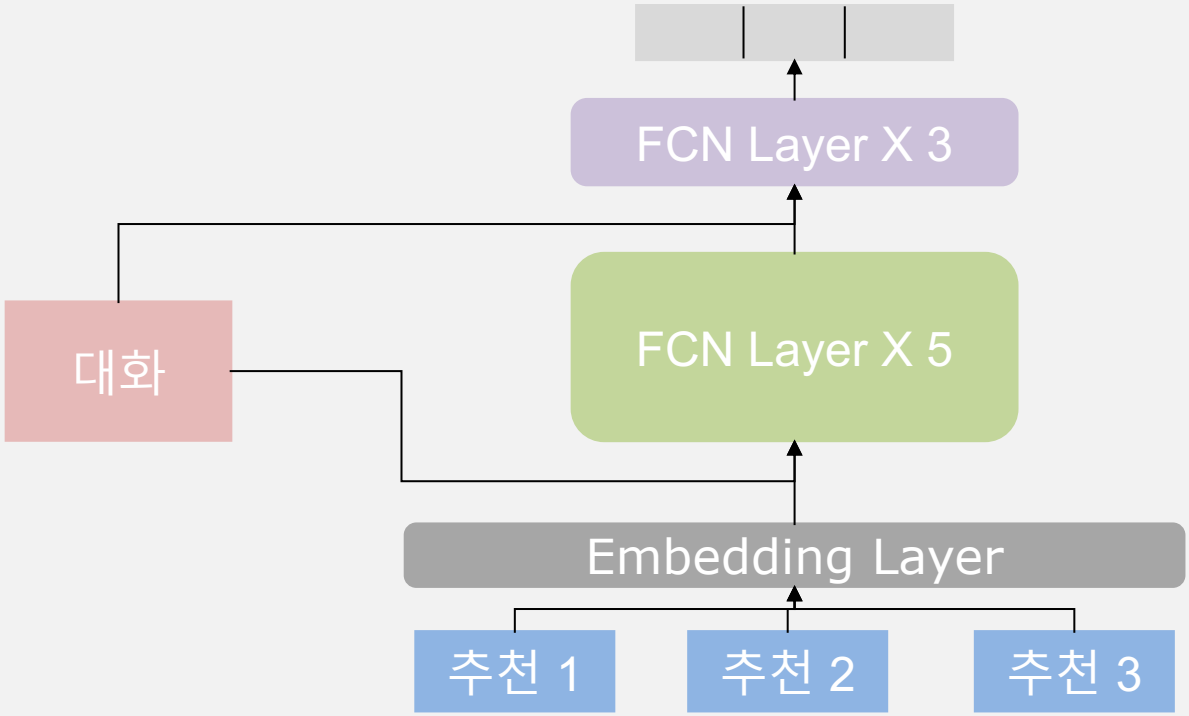
Memory Network 구조

```python
def forward(self, stories):
    """
    build graph for end-to-end memory network
    """
    # query embedding
    u_0 = torch.matmul(self._queries, self._B)
    u = [u_0]
    for _ in range(self._hops):
        # key embedding
        m_temp = torch.matmul(
            torch.reshape(stories, (-1, self._embedding_size)),
            self._A
        )
        m = torch.reshape(m_temp,
            (-1, self._mem_size, self._text_feat_size)
        )

        u_temp = torch.transpose(
            torch.unsqueeze(u[-1], -1), 2, 1
        )

        # get attention
        dotted = torch.sum(m * u_temp, 2)
        probs = F.softmax(dotted, 1)
        probs_temp = torch.transpose(
            torch.unsqueeze(probs, -1), 2, 1
        )

        c = torch.matmul(torch.reshape(stories,
            (-1, self._embedding_size)),
            self._C
        )
        c = torch.reshape(c,
            (-1, self._mem_size, self._text_feat_size)
        )
        c_temp = torch.transpose(c, 2, 1)

        o_k = torch.sum(c_temp * probs_temp, 2)
        u_k = torch.matmul(u[-1], self._H) + o_k
        u_k = self.nonlin(u_k)

        # [u_0, u_1, u_2 ... u_k]
        u.append(u_k)

    req = torch.matmul(u[-1], self._W)
    return req
```

Memory Network 코드

1. **Memory Network 구조를 통해서 데화 데이터에 대한 Feature를 추출**
    - 베이스라인 코드 그대로 사용
    - (Batch Size, Mem Size, Feature Size)의 입력이 (Batch Size, Feature Size)로 변환이 된다.
2. 인자로 주어지는 hops의 값만큼 왼쪽에서의 동작을 반복적으로 진행하게 된다.
    - **처음에는 q 벡터가, 그 다음부터는 이전의 결과값을 바탕으로 attention 값을 구한다.**

3. 모델의 구조



1. Memory Network 구조를 통해서 데화 데이터에 대한 Feature를 추출한다.
2. Embedding Layer를 통해서 옷 추천 조합 데이터에서 Feature를 추출한다.
3. 해당 Feature를 Concatenate해서 첫 번째 FCN Layer로 보낸다.
4. 첫 번째 FCN Layer의 출력 값을 다시 대화 데이터 Feature로 Concatenate를 한다.
5. 4번의 결과를 두 번째 FCN Layer로 보낸다.

5. 학습 및 평가

1. 학습 과정
2. 모델 성능 평가

1. 학습 과정

```
python train.py --in_file_trn_dialog ./data/task1.ddata.wst.txt \
    --subWordEmb_path ./data/sstm_v0p5_deploy/sstm_v4p49_np_n36134_d128.dat \
    --epochs 10 \
    --seed 42 \
    --learning_rate 2e-5 \
    --dropout_prob 0.1 \
    --batch_size 16 \
    --hops 3 \
    --mem_size 24 \
    --key_size 512 \
    --text_feat_size 512 \
    --img_feat_size 512 \
    --logging_steps 300 \
    --eval_node [6000,6000,6000,6000,512][2000,2000,2000]

python evaluate.py --in_file_tst_dialog ./data/cl_eval_task1.wst.dev \
    --subWordEmb_path ./data/sstm_v0p5_deploy/sstm_v4p49_np_n36134_d128.dat \
    --model_path gAIa_CL_model \
    --model_file gAIa-final-42.pt \
    --dropout_prob 0.1 \
    --eval_batch_size 8 \
    --hops 3 \
    --mem_size 24 \
    --key_size 512 \
    --text_feat_size 512 \
    --img_feat_size 512 \
    --eval_node [6000,6000,6000,6000,512][2000,2000,2000]
```

task1.ddata.wst 학습

```
# STEP 2
python train.py --in_file_trn_dialog ./data/task2.ddata.wst.txt \
    --subWordEmb_path ./data/sstm_v0p5_deploy/sstm_v4p49_np_n36134_d128.dat \
    --epochs 10 \
    --seed 42 \
    --model_path gAIa_CL_model \
    --model_file gAIa-final-42.pt \
    --learning_rate 1e-5 \
    --dropout_prob 0.1 \
    --batch_size 16 \
    --hops 3 \
    --mem_size 24 \
    --key_size 512 \
    --text_feat_size 512 \
    --img_feat_size 512 \
    --logging_steps 300 \
    --eval_node [6000,6000,6000,6000,512][2000,2000,2000]

python evaluate.py --in_file_tst_dialog ./data/cl_eval_task1.wst.dev \
    --subWordEmb_path ./data/sstm_v0p5_deploy/sstm_v4p49_np_n36134_d128.dat \
    --model_path gAIa_CL_model \
    --model_file gAIa-final-42.pt \
    --dropout_prob 0.1 \
    --eval_batch_size 8 \
    --hops 3 \
    --mem_size 24 \
    --key_size 512 \
    --text_feat_size 512 \
    --img_feat_size 512 \
    --eval_node [6000,6000,6000,6000,512][2000,2000,2000]

python evaluate.py --in_file_tst_dialog ./data/cl_eval_task2.wst.dev \
    --subWordEmb_path ./data/sstm_v0p5_deploy/sstm_v4p49_np_n36134_d128.dat \
    --model_path gAIa_CL_model \
    --model_file gAIa-final-42.pt \
    --dropout_prob 0.1 \
    --eval_batch_size 8 \
    --hops 3 \
    --mem_size 24 \
    --key_size 512 \
    --text_feat_size 512 \
    --img_feat_size 512 \
    --eval_node [6000,6000,6000,6000,512][2000,2000,2000]
```

task2.ddata.wst 학습

1. task1 → task2 → task3 → …. 순서대로 **학습하고, 저장하고 평가하고, 불러오고 등의 과정을 반복**한다.
2. **task2 데이터를 학습할 때부터 ewc loss를 구해서 망각효과를 최소화 한다.**
3. task6 데이터를 학습할 때 까지 eval_task1, eval_task2, eval_task3에 대한 성능을 지켜보면서 모델의 성능을 평가한다.

2. 모델의 성능 평가

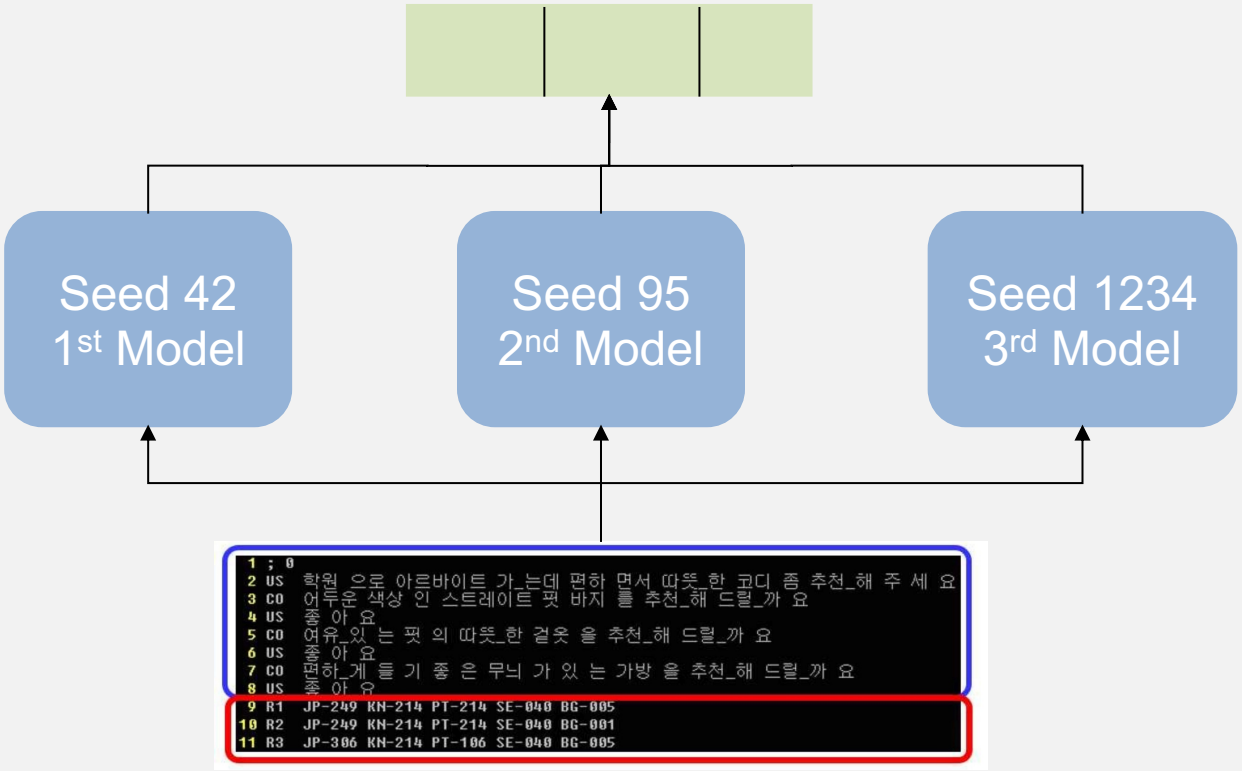
	eval_task1	eval_task2	eval_task3
task1.ddata	0.6568		
task2.ddata	0.6091	0.6205	
task3.ddata	0.5909	0.5909	0.5091
task4.ddata	0.6455	0.6182	0.4159
task5.ddata	0.6341	0.4159	0.4750
task6.ddata	0.6773	0.5636	0.5159

모델 성능 결과표

6. 최종 모델 선정

1. 앙상블

1. 앙상블



- **동일한 모델 구조, 동일한 Hyperparameter 에서 Seed 만을 다르게 하고 3개의 모델을 생성한다.**
- 각 모델의 출력 결과로 나온 Logit 값을 모두 더하고 그 Logit을 가지고서 추천 옷 조합 데이터의 순위를 매긴다.