



# Fashion-How 발표자료

## Sub-Task 4

---

박상하



# CONTENTS

1. 문제 정의	2. 데이터 확인	3. 데이터 전처리	4. 모델 설계	5. 학습	6. 최종 모델
1. 과제 목표 2. 문제 정의	1. 데이터 확인	1. 데이터 전처리 2. 데이터 증강	1. 모델 구조 설계	1. 학습 과정	1. 앙상블

# 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. 추론을 할 때는 학습할 때 입력되지 않는 액세서리 이미지가 입력이 되기 때문에 Zero-Shot Learning을 진행하도록 한다.
-



## 2. 데이터 확인

### 1. 데이터 확인

---

# 1. 학습 및 평가 데이터 확인

```
1 0 <CO> 어서 오 세 요 코디 랑 입 니다 무엇 을 도와 드릴 까 요 INTRO
2 1 <US> 처음 대학교 들어가 는데 입 을 속 코디 해 주 세 요
3 2 <CO> 신발상 코디 에 어울리 게 화사 한 스커트 를 추천_해 드릴_게 요 EXP_RES_SITUATION;EXP_RES_DESCRIPTION
4 3 <AC> SK-009
5 4 <US> 이 속 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 짧은 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 짧은 치마 입 니다 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> 언밸런스 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 외부 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 도 추천_해 주 세 요
17 16 <AC> CT-019
18 17 <CO> 이너 색상 과 무난 하 게 잘 어울릴 트렌치 코트 입 니다 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 니다 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 SK-009 SK-053 SE-039
26 25 <CO> 네 지금 까지 제안 해 드린 아이템 으로 전체 코디샷 을 제안 해 드립 니다 CONFIRM_SHOW
27 26 <US> 마음 에 드 시_나 요 CONFIRM_SATISFACTION
28 27 <CO> 네 마음에 드 네 디 감사 합 니다 USER_SUCCESS
29 28 <US> 마음에 드 신디 나 다행 입 니다 SUCCESS
30 29 <CO> 이쯤 해 주 셔 감사 합 니다 CLOSING
```

```
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 4 평가 데이터

Sub-Task 4 학습 데이터

1. 학습 및 평가 데이터가 일치하지 않고 상당한 차이가 존재하는 것을 확인
2. 학습 데이터를 전처리를 통해서 평가 데이터와 유사하게 만들어줘야 한다는 것을 파악
3. 평가할 때는 학습할 때 입력되지 않았던 액세서리 옷에 대해서 대응을 해주는 과정이 필요

## 2. 메타 데이터 확인

CD-001	0	CD	F	상의 스트레이트 실루엣
CD-001	0	CD	F	상의 엉덩이 덮는 길이
CD-001	0	CD	F	상의 일자형 밑단
CD-001	0	CD	F	상의 브이 넥라인
CD-001	0	CD	F	상의 드림 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 4 메타 데이터

1. 각 옷에 따른 패션 항목, 종류, 특징 등이 저장 되어있다.
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_리고 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 겔옷 이 있_으면 좋 겠_네 요',  
  '베이지 색상 의 겔옷 이 포함_된 코디 세트 입 니다 마음 에 드 시_나 요',  
  '상의 가 너무 화려 함 니다 흰색 톤 으로 추천 부탁 드려_요',  
  '밝 은 색상 의 블라우스 는 어떠 신가 요',  
  '마음 에 들_니다',  
  '이 블라우스 로 변경 된 전체 코디 세트 를 보여 드릴_까 요',  
  '네',  
  '흰색 의 블라우스 와 점퍼 가 포함_된 코디 세트 입니다 어떠 신가 요',  
  '화려_하 면서_도 겔옷 이 있_어서 좋 습니다',  
  '다만 신발 이 불편_해 보여 서 편안 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 세트 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 셧_다 니 기쁩 니다',  
  '이용_해 주 셔 서 감사_함 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 대화로 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_categority = position_of_fashion_item(img)
    img_category[img_categority].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 값을 구한다.
3. 2번 과정을 통해서 구한 벡터들을 기반으로 카테고리 내에서 옷들의 유사도를 계산한다.

### 3. 코디 데이터 전처리



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) :

    org_cordi = cordi[0]
    targets = [k for k in range(self.num_cordi) if "NONE" not in org_cordi[k]]

    aug_cordi = copy.deepcopy(org_cordi)
    target_ids = random.sample(targets, 2)
    for t_id in target_ids :
        t_img = org_cordi[t_id]

        img_id = img2id[t_id][t_img]
        img_sim = img_similarity[t_id][img_id]

        rank_args = np.argsort(img_sim[::-1])[1:]
        select_id = np.random.randint(100)
        select_arg = rank_args[select_id]
        select_img = id2img[t_id][select_arg]

        aug_cordi[t_id] = select_img

    data = {"diag" : diag,
           "cordi" : [cordi[0], cordi[np.random.randint(1,3)]] + [aug_cordi],
           "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 두 개를 선정한다.
3. 해당 옷들의 카테고리 내에서 그 옷과 가장 유사한 옷 100개 옷들 중에서 임의로 1개의 옷을 선택한다.
4. 해당 옷을 선정된 옷으로 변경하여 조합을 만들어서 데이터를 추가한다.
5. Sub-Task3에서는 최대한 유사한 조합으로 변경하였다면 Sub-Task4에서는 최대한 다양한 조합을 만들도록 증강을 진행

## 5. 데이터 인코딩

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

    def __call__(self, dataset) :
        diag_list, cordi_list, rank_list = [], [], []
        for data in tqdm(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))]

            cordi = data["cordi"]
            cordi_names = [
                [[c(j) for j in range(self.num_cordi)]]
                for c in cordi
            ]

            diag_list.append(vectors)
            cordi_list.append(cordi_names)
            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 옷의 이름을 추출해서 리스트 형태로 바꾼다.

## 6. 모델의 입력 Tensor

```
class FashionHowDataset(Dataset):
    def __init__(self, dataset, img_feat_dir):
        self.diag = dataset["diag"]
        self.cordi = dataset["cordi"]
        self.rank = dataset["rank"] if "rank" in dataset else None
        self.img_feat_dir = img_feat_dir

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

    def __getitem__(self, idx):
        diag = self.diag[idx]
        cordi = self.cordi[idx]

        cordi_array = []
        for c in cordi :
            vectors = []
            for cloth in c :
                if "NONE" in cloth :
                    vector = np.zeros((4, 2048)).tolist()
                else :
                    file_name = os.path.join(self.img_feat_dir, cloth + "_feat.npy")
                    vector = np.load(file_name).tolist()
                vectors.append(vector)
            cordi_array.append(vectors)

        if self.rank == None :
            return diag, cordi_array
        else :
            rank = self.rank[idx]
            return diag, cordi_array, rank
```

Torch Dataset 모듈

1. 앞서 추출한 옷의 이름들을 활용해서 사전에 저장되어 있는 npy파일을 불러온다.
2. npy 파일에 저장되어있는 Feature의 크기는 (4, 2048)이다.
3. 조합 중에서 Dummy Label에 해당되는 옷은 zero 벡터로 (4, 2048)을 만들어서 대체한다.
4. 최종적으로 옷 조합의 벡터는 (Batch Size, Num Rank, Num Coordi, 4, 2048) 이 된다.

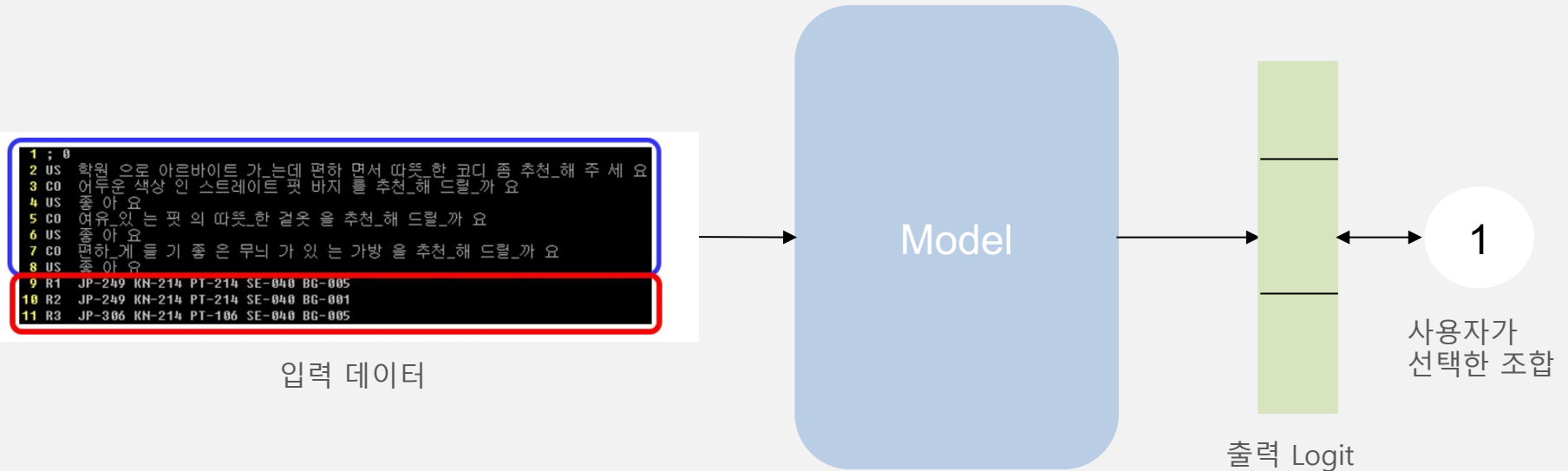


## 4. 모델 설계

### 1. 모델 구조 설계

---

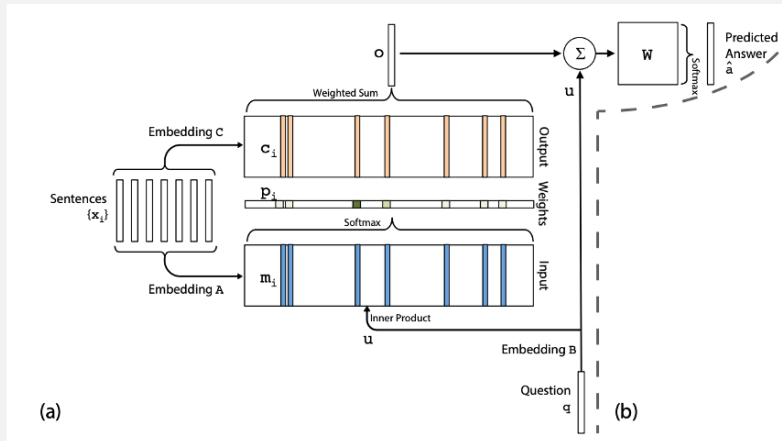
# 1. 모델의 입출력 설계



1. 모델에는 대화데이터, 추천 옷 조합데이터가 입력되어야 한다.
  - 대화데이터, 추천 옷 조합데이터를 인코딩하는 부분이 각각 필요하다.
2. **학습을 할 때는 조합을 구성하는 옷의 수가 4개이지만 평가를 할 때는 5개이다.**
3. 모델은 3개의 Logit 값을 출력한다.
4. 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

```
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

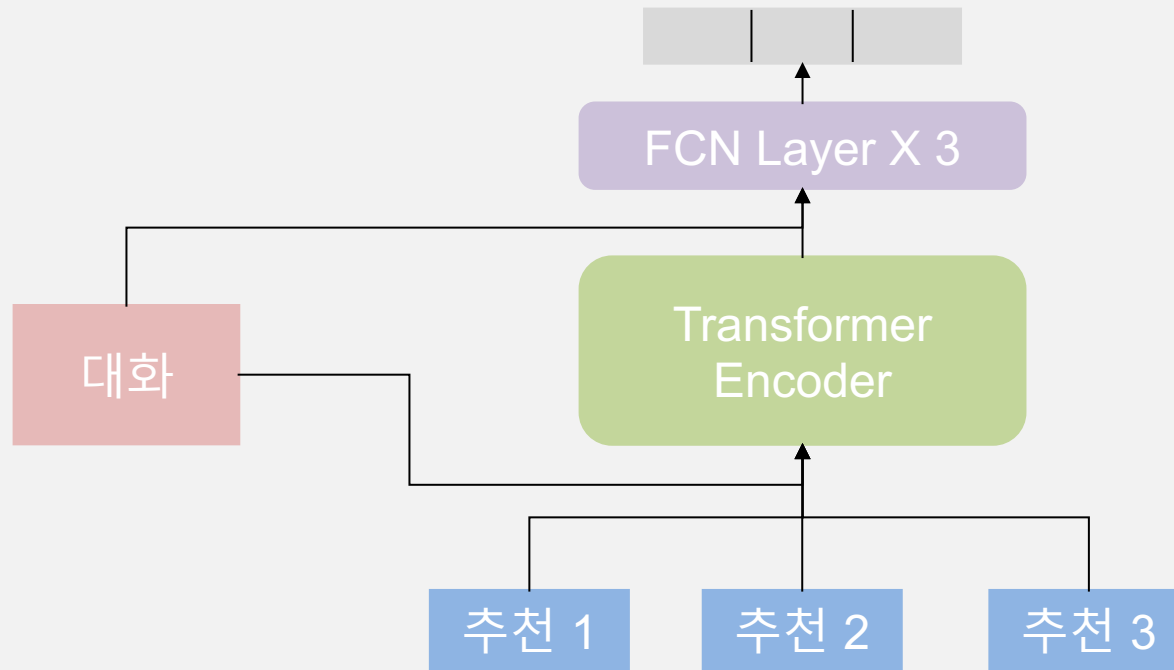
### 1. Memory Network 구조를 통해서 대화 데이터에 대한 Feature를 추출

- 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 대화 데이터에 대한 Feature를 추출한다.
2. Dataset 모듈을 통해서 사전에 저장 되어있는 Feature를 불러오기 때문에 Embedding Layer은 필요 없다.
3. (Batch Size, Num Rank, Num Coordi, 4, 2048)의 Feature를 (Batch Size, Num Rank, 4, 2048)로 Coordi 축을 기준으로 mean 벡터를 구한다.
4. mean 벡터를 최종적인 Feature로 간주해서 1번에서 구한 Feature와 Concatenate해서 Transformer Encoder구조의 모듈에 입력한다.

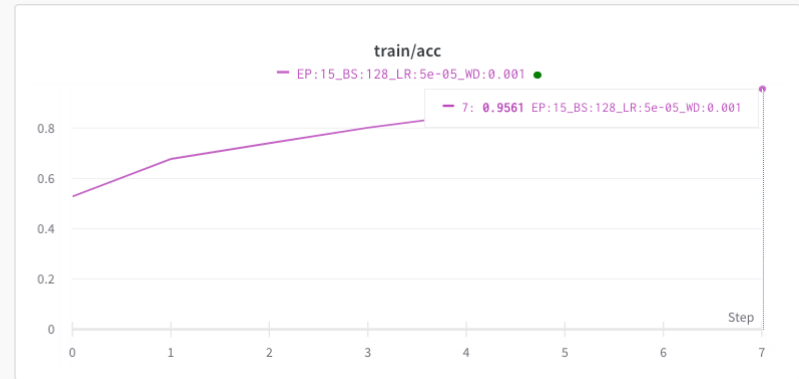


## 5. 학습 및 평가

1. 학습 과정
  2. 모델 성능 평가
-

# 1. 학습 과정

train 4



모델 학습 로그

1. 평가 데이터에서는 훈련 데이터에 없는 액세서리 이미지가 입력이 되기 때문에 평가 데이터를 따로 만들 수가 없다.
2. 모델 학습 과정만으로 모델이 잘 학습이 되는지 안되는지 판단을 해야 한다.
  1. 학습 과정에서 사용자가 가장 좋아하는 옷을 잘 맞추는 것을 알 수 있었다.
  2. 학습 과정이든, 평가 과정이든 옷 조합에 대한 Feature는 항상 (Batch Size, Num Rank, 4, 2048)이 입력되기 때문에 이를 바탕으로 Zero-Shot Learning을 잘 할 수 있을 것이라고 예상했다.

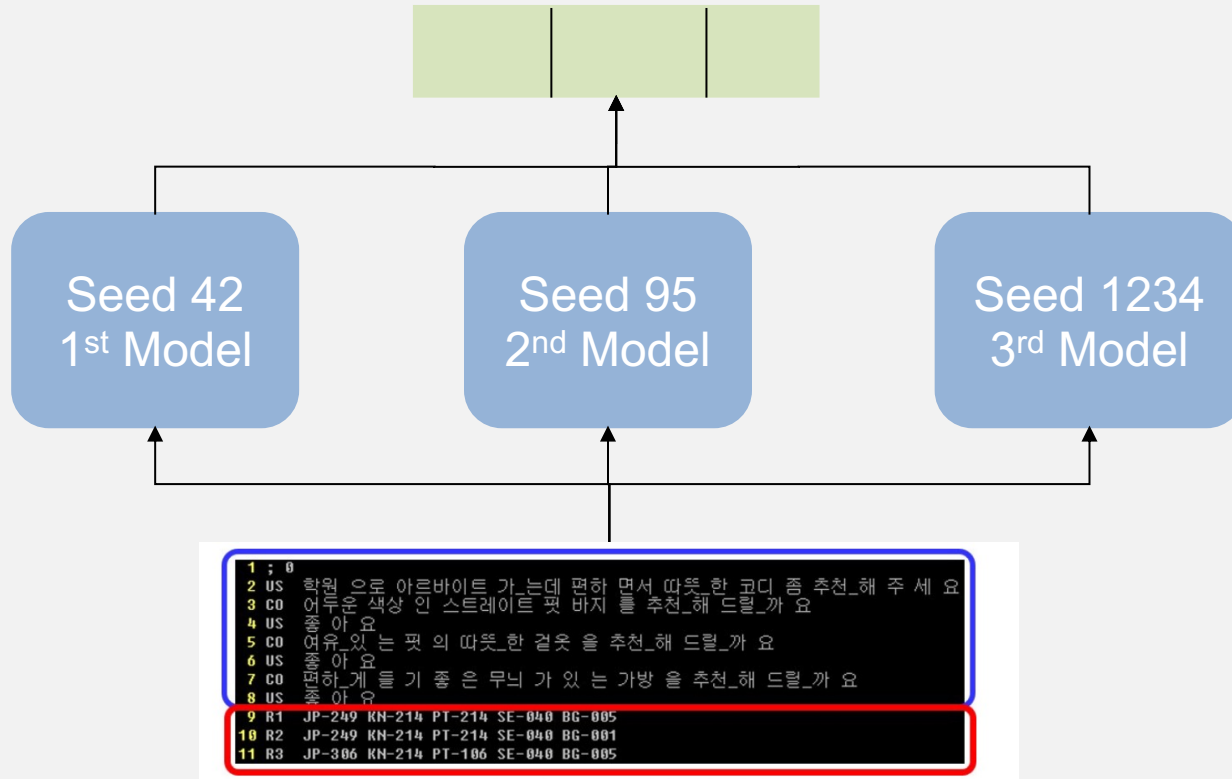


## 6. 최종 모델 선정

### 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.