Marcos Grobas, José Romero

P0 - Sistemas de Recomendación

1 Introducción

2 ¿Cómo hicimos?

Usamos dos modelos.

2.1 Modelo de reputación

2.2 Modelo de Filtrado colaborativo [1]

$$B = AA^{'}$$

hola

3 Evaluacion

```python

# marcos pongo esto por si nos hace falta usar codigo python, si no
    lo quitamos

class Attention(nn.Module):
    def __init__(self, dim, heads=8, dim_head=64, dropout=0., order
    ='first'):
        super().__init__()
        inner_dim = dim_head * heads
        project_out = not (heads == 1 and dim_head == dim)

        self.heads = heads
        self.scale = dim_head ** -0.5
        self.order = order  # 'first' or 'second'

        self.attend = nn.Softmax(dim=-1)
        self.dropout = nn.Dropout(dropout)

        self.qkv = nn.Linear(dim, inner_dim, bias=False)

        self.to_out = nn.Sequential(
```

[1] https://en.wikipedia.org/wiki/Collaborative_filtering

```
            nn.Linear(inner_dim, dim),
            nn.Dropout(dropout)
        ) if project_out else nn.Identity()

    def forward(self, x):
        w = rearrange(self.qkv(x), 'b n (h d) -> b h n d', h=self.
    heads)

        # Compute (U^T Z)^T (U^T Z)
        dots = torch.matmul(w, w.transpose(-1, -2)) * self.scale

        if self.order == 'first':
            # First-order Neumann approximation
            # out = (U^T Z) * softmax((U^T Z)^T (U^T Z))
            attn = self.attend(dots)
            attn = self.dropout(attn)
            out = torch.matmul(attn, w)

        elif self.order == 'second':
            # Second-order Neumann approximation
            # out = out_1st - out_2nd

            # First order term: (U^T Z) * softmax((U^T Z)^T (U^T Z)
    )
            attn_1st = self.attend(dots)
            attn_1st = self.dropout(attn_1st)
            out_1st = torch.matmul(attn_1st, w)

            # Second order term: (U^T Z) * softmax(((U^T Z)^T (U^T
    Z))^2)
            # Compute ((U^T Z)^T (U^T Z))^2
            dots_2nd = torch.matmul(dots, dots)
            attn_2nd = self.attend(dots_2nd)
            attn_2nd = self.dropout(attn_2nd)
            out_2nd = torch.matmul(attn_2nd, w)

            # Combine: subtract second order correction
            out = out_1st - out_2nd

        else:
            raise ValueError(f"order must be 'first' or 'second',
    got {self.order}")

        out = rearrange(out, 'b h n d -> b n (h d)')
        return self.to_out(out)
```

Listing 1: la atencion