BERT + GRAPH CONVOLUTIONAL NETWORK

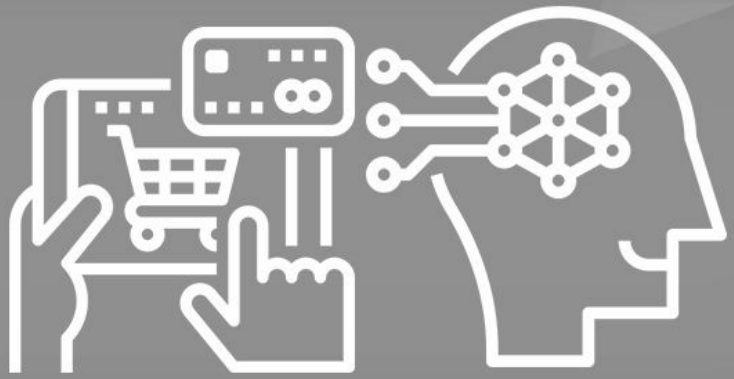
Retail Revamp

Jackson Kaunismaa, Julia Wang, Ryan Chen

1

Why

The impact of product recommendation systems on business profit

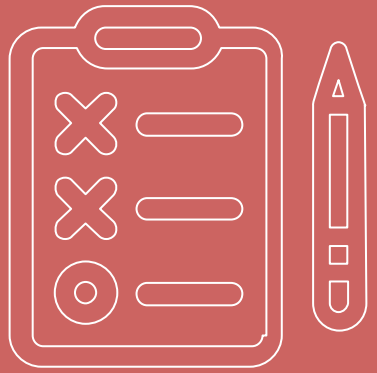


The Issue in Small Business Economy

- Small businesses are reliant on regular customers
  - Forbes: 40% of a business's revenue comes from repeat customers [1]
- Our Goal: grow number of regulars by appealing to niche product categories
  - Using AI to recommend products
- Nvidia: the average intelligent recommender system delivers a 22.66% lift in conversions rates (converting clicks to purchases) [2]

Metrics

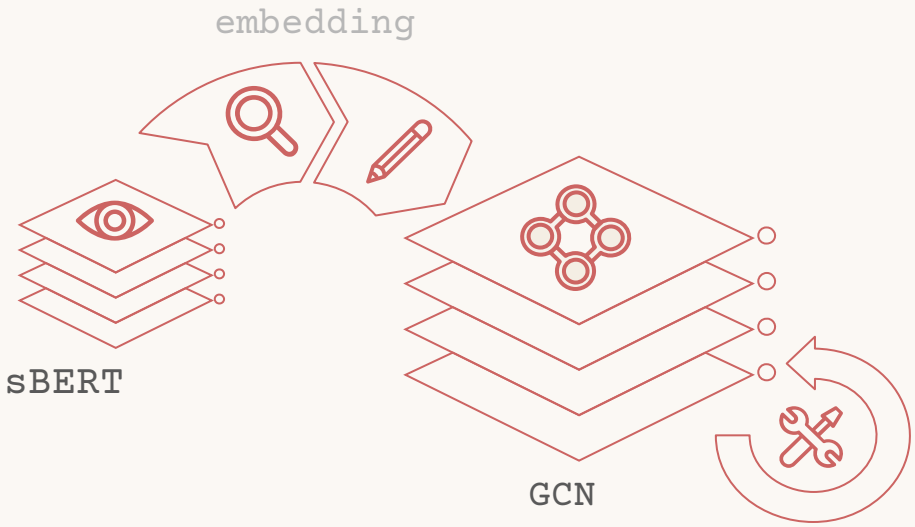
- Recall evaluates the model's ability to pick items relevant to customers
- Rather than use collaborative filtering, content-based, or knowledge-based recommenders:
  - Combining sentence Bidirectional Encoder Representations from Transformers (sBERT)
  - with a Graph Convolutional Network (GCN)



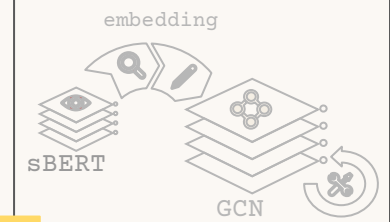
2

Our Model

sBERT with a Graph Convolutional Network (GCN)

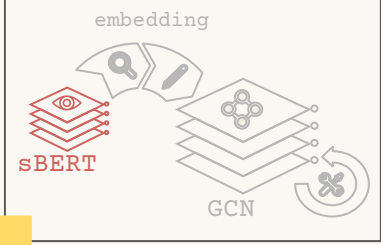


Data Processing



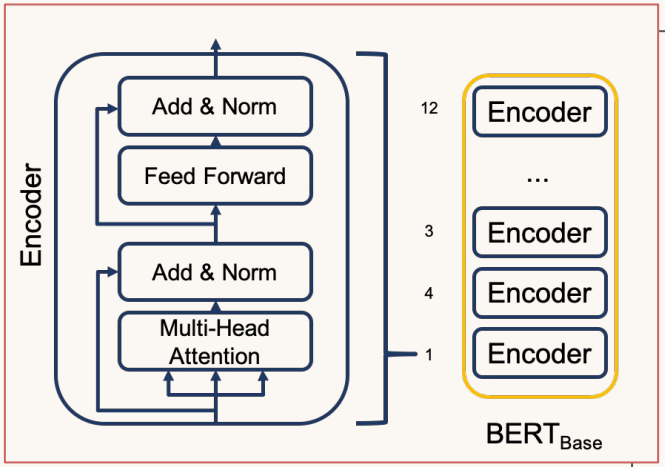
- UCI Online Retail dataset
- Trained on features:
  - Purchase description
  - Purchase quantity and unit price
- Numerical encoding of input features
- Filtering of invalid data entries
- Temporally split data, 80:20 training-validation

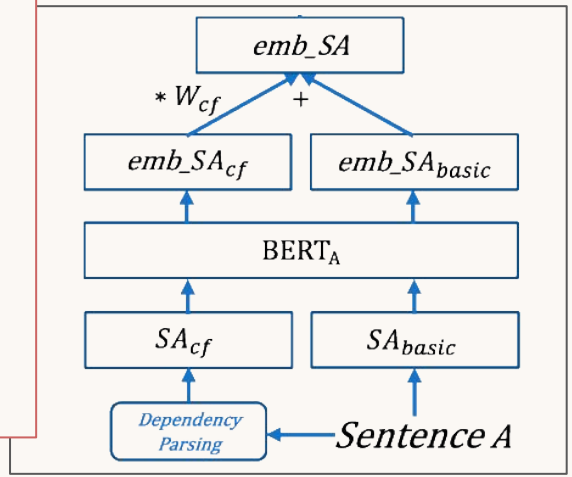
Sentence BERT (sBERT)



Input: item description in text format

Output: numeric embeddings to use as input feature to GCN

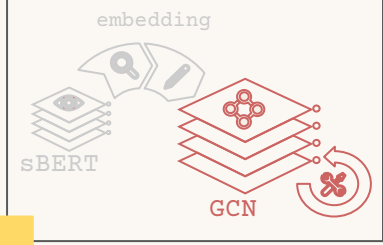




- Transformer with encoder layers and self-attention heads
- Bidirectional: reads entire sequence of words at once
- Encodes semantic knowledge
- Deals with new entries (weakness of CBRS)

```
sbert_model = SentenceTransformer('bert-base-nli-mean-tokens')
sentence_embeddings = sbert_model.encode(df['Description'])
```

Graph Convolutional Network



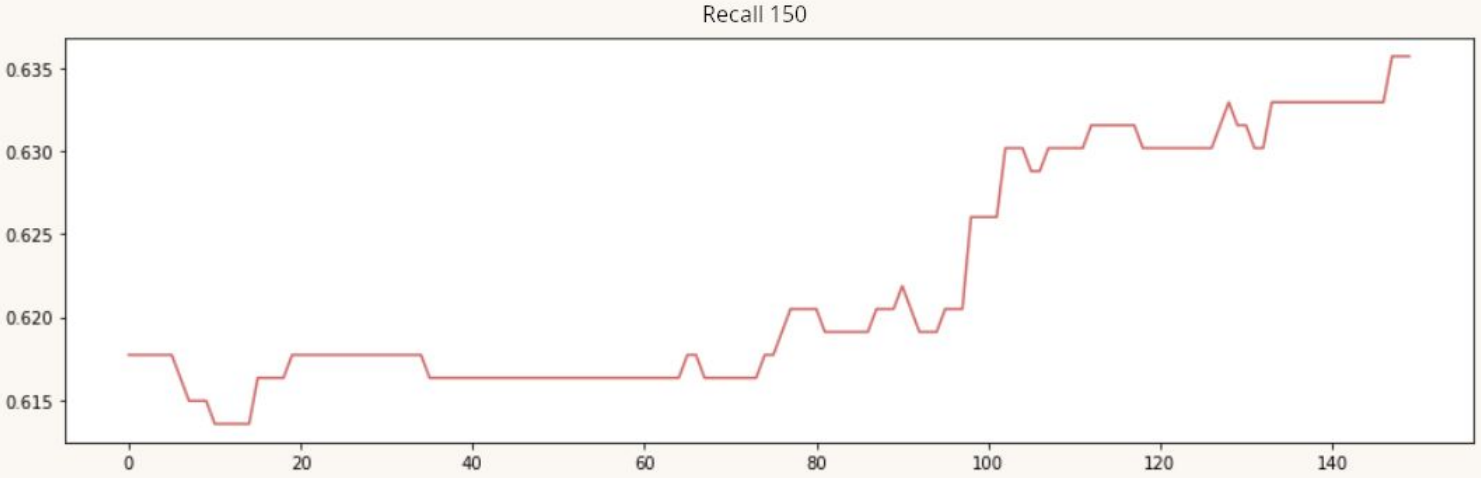

```
[ ]  class GraphEmbedder(nn.Module):
       def __init__(self, embed_size, num_layers, num_nodes, dropout_p=0.5):
         super().__init__()
         self.embed = nn.Embedding(num_nodes, embed_size)
         layers = [ptg.nn.GCNConv(embed_size, embed_size) for _ in range(num_layers)]
         self.layers = nn.ModuleList(layers)
         self.dropout_p = dropout_p

       def forward(self, data):
         x,edge_indices = data.x, data.edge_index
         x = self.embed(x)
         for i, layer in enumerate(self.layers):
           dropped = F.dropout(x, p=self.dropout_p, training=self.training)
           x = layer(dropped, edge_indices)
           if i != len(self.layers) - 1:
             x = F.relu(x)
         return x
```

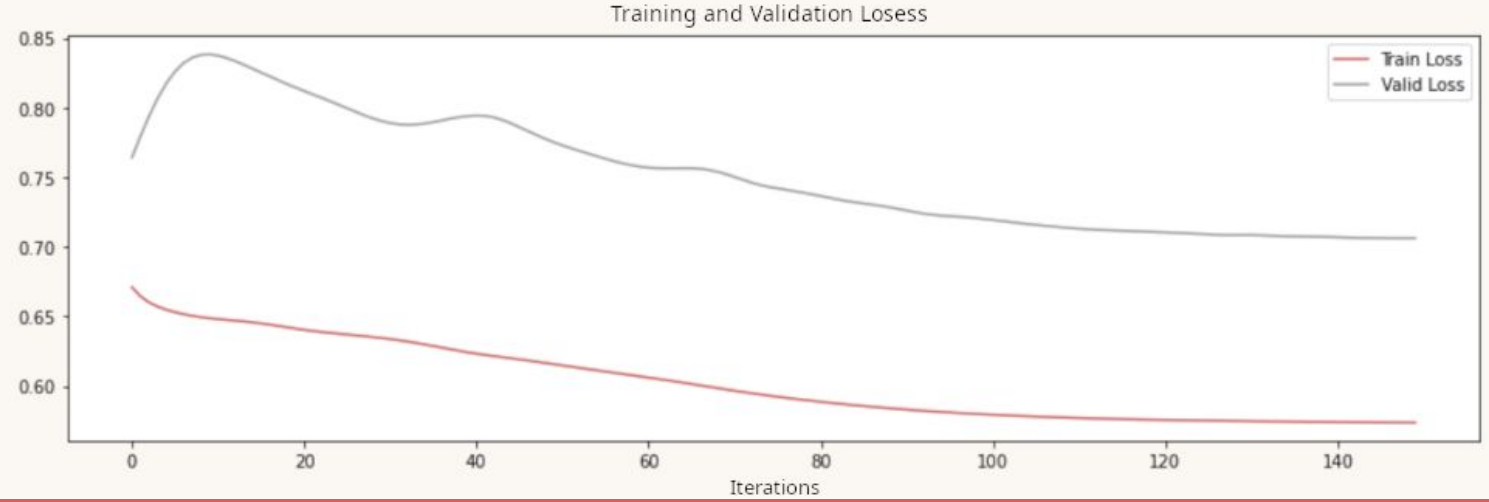
- Computes graph-aware embeddings of users and items
- Uses generated embeddings to predict item-user “similarity” as a proxy for items to recommend

3

Results

Using sBERT and GCN to recommend products


Recall 150
0.635
0.630
0.625
0.620
0.615
0
20
40
60
80
100
120
140


Training and Validation Losess
Train Loss
Valid Loss
0.85
0.80
0.75
0.70
0.65
0.60
0
20
40
60
80
100
120
140
Iterations

Next Steps

- Better hardware for training
- Testing on more diverse datasets
- Validation on actual real world data + implementation in an actual business
- Investigate how to deal with the “new user” and “new item” problems more effectively (add exploration/exploitation tradeoff)

Thanks for listening

References

[1] https://www.forbes.com/sites/forbesbusinesscouncil/2020/08/21/three-reasons-small-business-recovery-relies-on-customer-retention-not-acquisition/?sh=66066d135073

[2] https://www.nvidia.com/en-us/glossary/data-science/recommendation-system/#:~:text=This%20feature%20is%20often%20implemented,conversions%20rates%20for%20web%20products.