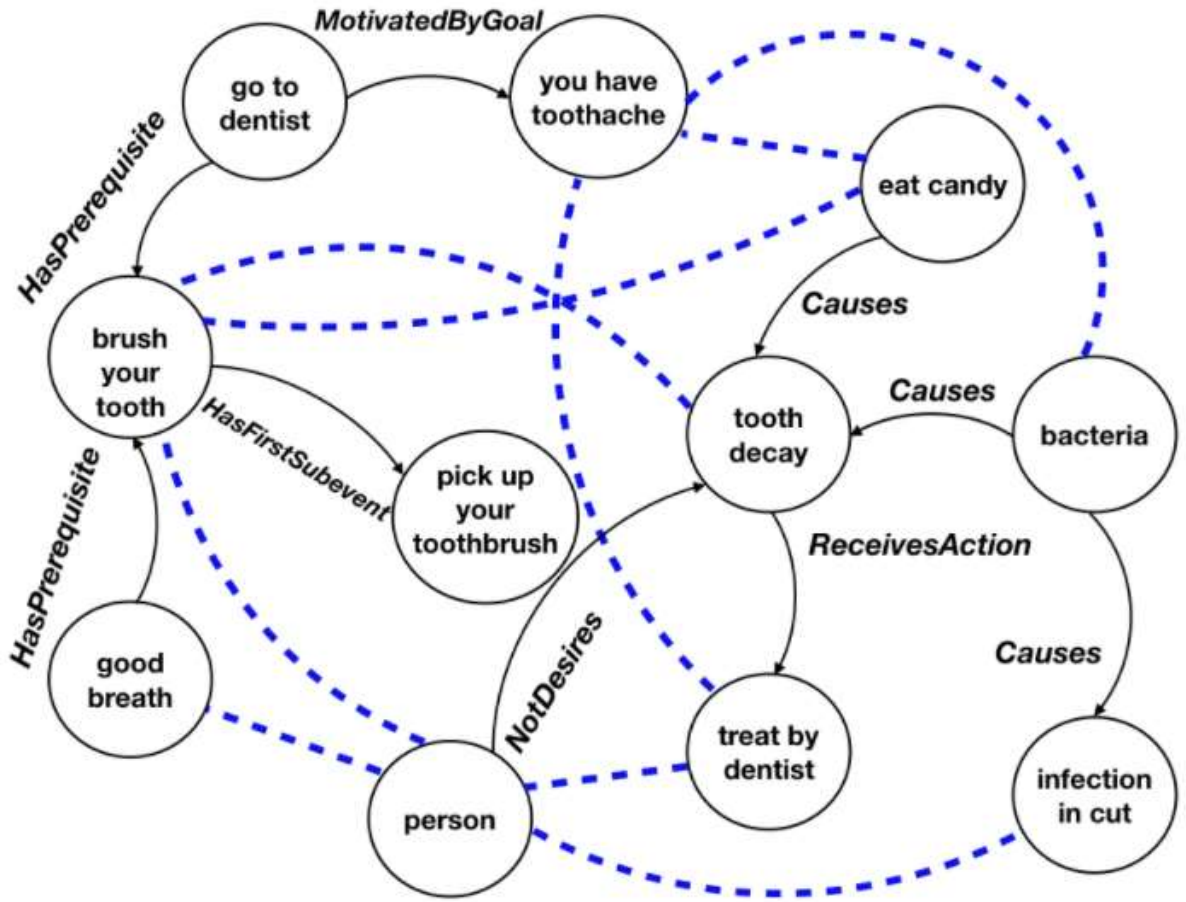
Robust Knowledge Graph Completion

Knowledge Graph Introduction

KG Representation

- Edges in KG are represented as **triples** (h, r, t)
  - head (h) has relation (r) with tail (t)
- **Key Idea:**
  - Model entities and relations in the embedding/vector space $\mathbb{R}^d$.
    - Associate entities and relations with **shallow embeddings**
    - **Note we do not learn a GNN here!**
  - Given a true triple (h, r, t), the goal is that the embedding of (h, r) **should be close** to the embedding of t.
    - How to embed (h, r)?
    - How to define closeness?

What's Knowledge Graph ?



Knowledge Dataset Introduction

1. SNOMED CT DATASET (System Nomenclature Of Medicine)
2. CN DATASET (Common sense)
3. FB15K-237 DATASET (FREE BASE)
4. FB15K-237-Sparse DATASET (FREE BASE)

Abstract

1.使用兩種 ranking方式

共同決定 (ensemble的感覺) model ranking的感覺

(1) Teacher network

(2) Student network

2.Sparsity 的所帶來的影響

兩種 ranking方式

1.Entity ranking

2.Entity re-ranking

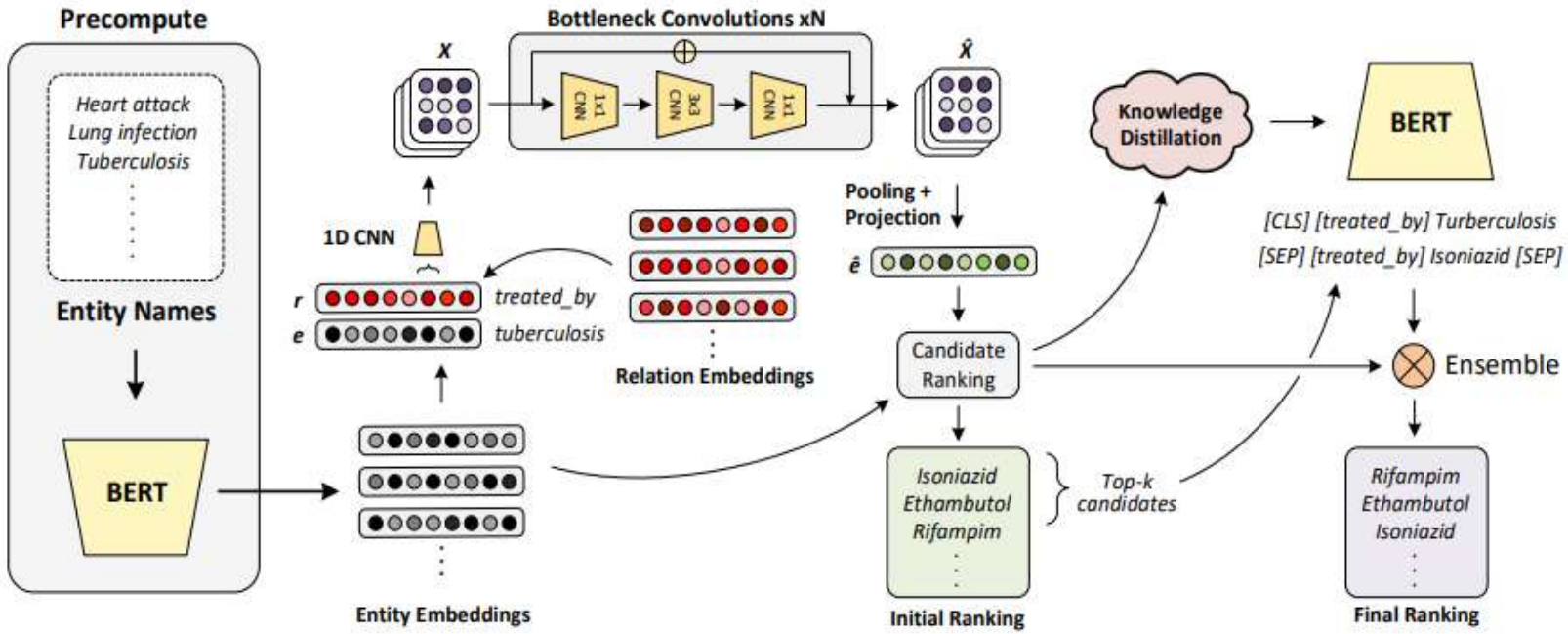


Figure 2: We utilize BERT to precompute entity embeddings. We then stack the precomputed entity embedding with a learned relation embedding and project them to a two-dimensional spatial feature map, upon which we apply a sequence of two-dimensional convolutions. The final feature map is then average pooled and projected to a query vector, which is used to rank candidate entities. We extract promising candidates and train a re-ranking model utilizing knowledge distilled from the original ranking model. The final candidate ranking is generated by ensembling the ranking and re-ranking models.

Entity ranking

介紹:

標準的formula 是 一個 entity (資料)就是 tuple (e1, r, e2)

我們給定(e1, r, ?)

求出 e2

此外，會把 (e_2, r^{-1}, e_1) 丟進去當作entity，也就是 上面的inverse

Entity ranking Training 方式(Teacher)

Loss: binary cross-entropy (補充好處)

1.compute scores for all entities

2.apply a sigmoid operator(softmax) to induce a probability for each entity

Optimizer: Adam

(1)decoupled weight de-cay regularization

(2)label smoothing

$$\mathbf{s}_{ik:(i+1)k} = \mathrm{softmax}(f_T(\mathbf{x}_i)_{0:k}/T)$$

Adam Code

```
optimizer = torch.optim.AdamW(
    [
        {'params': params, 'weight_decay': args.weight_decay},
        {'params': heldout_params, 'weight_decay': 0},
    ],
    lr=args.lr)
```

Entity re-ranking Training 方式(Student)

ties. We design our student re-ranking network as a triplet classification model that utilizes the full candidate fact, (e_i, r_j, e_k), instead of an incomplete fact, $(e_i, r_j, ?)$. This allows the network to model

("head name", r_i, "tail name")

Knowledge Distillation

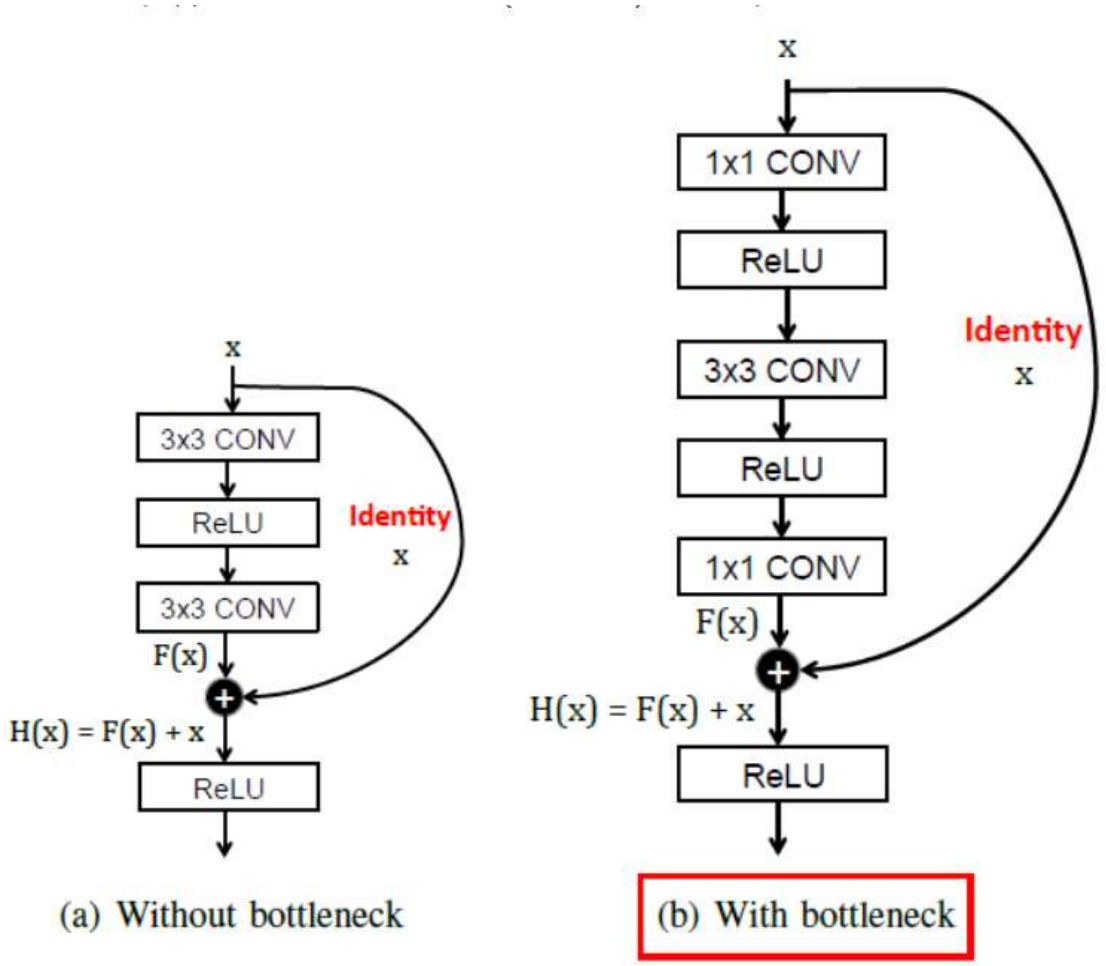
Our training objective for our student model, $f_S(x_i)$, is a weighted average of the binary cross entropy loss, $\mathcal{L}_{bce}$, using the teacher's normalized logits, $\mathbf{s}$, and the noisy training labels, $\mathbf{y}$.

$$\begin{aligned}\mathcal{L}_{KD}(y_i, x_i) &= \lambda \mathcal{L}_{bce}(s_i, f_S(x_i)) \\ &\quad + (\lambda - 1)\mathcal{L}_{bce}(y_i, f_S(x_i)) \\ &= \mathcal{L}_{bce}((\lambda - 1)y_i + \lambda s_i, f_S(x_i))\end{aligned}$$

Bottleneck

透過增加1x1層數 (就像瓶頸)

改變維度



Student-Teacher Ensemble(一起ranking)

networks. The final ranking are computed with

$$\hat{\mathbf{s}}_{ik:(i+1)k} = \alpha(\text{softmax}(f_S(\mathbf{x}_{ik:(i+1)k}))) + (1-\alpha)(\text{softmax}(f_T(\mathbf{x}_i)_{0:k}))$$

student (distillation).

teacher

where $0 \leq \alpha \leq 1$ controls the impact of the student re-ranker. The cost of computing $\hat{\mathbf{s}}_{ik:(i+1)k}$ is negligible, so we sweep over $[0, 1]$ in increments of .01 and select the α that achieves the best validation MRR.

Ensemble: a group producing a single effect.

Experiment Result

	SNOMED CT Core					CN-100K				
	MR	MRR	H@1	H@3	H@10	MR	MRR	H@1	H@3	H@10
DistMult [♣]	5146	.293	.226	.318	.426	—	.090	.045	.098	.174
ComplEx [♣]	3903	.302	.224	.332	.456	—	.114	.074	.125	.190
ConvE [♣]	3739	.271	.191	.303	.429	—	.209	.140	.229	.340
ConvTransE [♣]	3585	.290	.213	.321	.442	—	.187	.079	.239	.390
BERT-ConvE	414	.383	.277	.430	.591	260	.453	.332	.521	.691
BERT-ConvTransE	514	.373	.273	.417	.568	276	.458	.340	.520	.675
BERT-Large-ConvTransE [♣]	—	—	—	—	—	—	.523	.410	.585	.735
BERT-DeepConv	265	.479	.374	.532	.685	**161**	.540	.418	.610	**.772**
BERT-ResNet	265	.492*	.389	.544	**.691**	169	.550*	.426	.628	.769
+ Re-ranking	265	.562†	.482	.608	**.691**	170	.377	.216	.437	.769
+ Knowledge Distillation (KD)	265	.566†	.487	.614	**.691**	169	.528	.402	.603	.769
+ Ranking Ensemble (RE)	**264**	.576†	**.503**	.619	**.691**	169	.555	.438	.623	.769
+ KD and RE	**264**	**.577**†	.501	**.623**	**.691**	169	**.569**†	**.452**	**.647**	.769

	FB15k-237					FB15k-237-Sparse				
	MR	MRR	H@1	H@3	H@10	MR	MRR	H@1	H@3	H@10
DistMult [♠]	—	.343	—	—	.531	3061	.136	.092	.146	.223
ComplEx [♠]	—	.348	—	—	**.536**	3333	.132	.091	.143	.216
ConvE [♠]	—	.339	—	—	.521	2263	.156	.106	.165	.258
ConvTransE [◇]	—	.33	.24	.37	.51	2285	.153	.103	.161	.255
BERT-ConvE	193	.305	.224	.330	.465	408	.190	.128	.200	.315
BERT-ConvTransE	211	.296	.218	.321	.449	**390**	.188	.127	.199	.310
BERT-DeepConv	190	.327	.246	.354	.488	422	.188	.127	.197	.314
BERT-ResNet	186	.346*	.262	.379	.514	413	.191*	.128	.201	**.317**
+ Re-ranking	187	.304	.212	.329	.514	413	.190	.128	.200	**.317**
+ Knowledge Distillation (KD)	187	.310	.220	.334	.514	413	.197†	.135	.209	**.317**
+ Ranking Ensemble (RE)	**186**	**.354**†	**.270**	**.387**	.514	413	**.199**†	**.137**	.210	**.317**
+ KD and RE	**186**	.353†	.269	.386	.514	413	.198†	.136	**.211**	**.317**

MR MRR H@k (k 是 正整數)

MR: 是 MRR再除以平均之前的倒數，越低答對率越高

MRR:越高越好，越高代表答對率越高

H@k: 計算 是否 predict中 真的有測試到前i個

$$\mathrm{MR} = \frac{1}{|\mathcal{T}|} \sum_{x_i \in \mathcal{T}} \mathrm{rank}(x_i)$$

The Mean Reciprocal Rank (MRR) is computed as

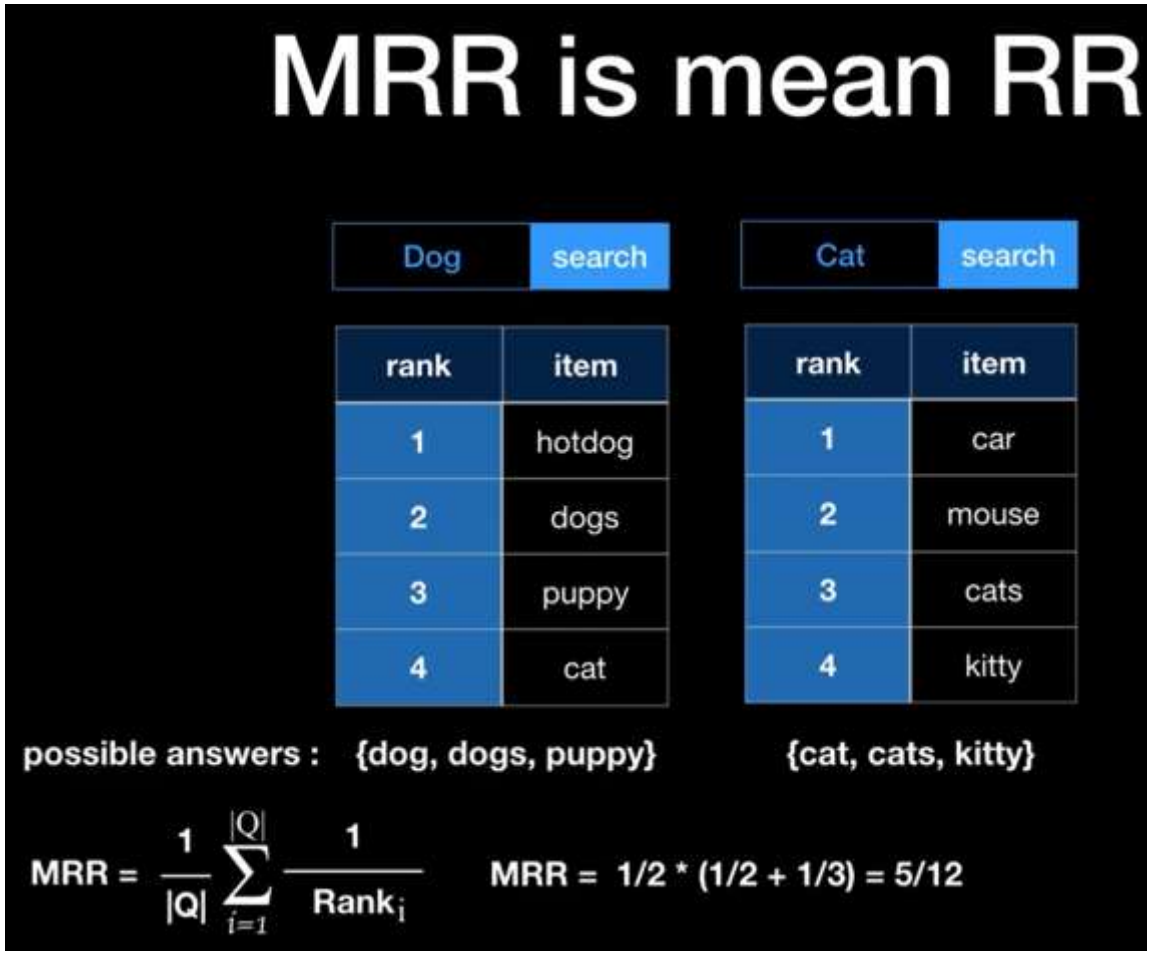
$$\mathrm{MRR} = \frac{1}{|\mathcal{T}|} \sum_{x_i \in \mathcal{T}} \frac{1}{\mathrm{rank}(x_i)}$$

The Hits at k (H@k) is calculated as

$$\mathrm{H@k} = \frac{1}{|\mathcal{T}|} \sum_{x_i \in \mathcal{T}} I[\mathrm{rank}(x_i) \leq k]$$

MRR舉例

```
d['MR'] = np.mean(ranks).item()
d['MRR'] = np.mean(1. / ranks).item()
```



H@k舉例

找有幾個 取平均

```
>>> a = np.arange(10)
>>> a
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
>>> np.where(a < 5, a, 10*a)
array([ 0,  1,  2,  3,  4, 50, 60, 70, 80, 90])
```

```
d['H@1'] = np.mean(np.where(ranks < 2, 1, 0)).item()
d['H@3'] = np.mean(np.where(ranks < 4, 1, 0)).item()
d['H@5'] = np.mean(np.where(ranks < 6, 1, 0)).item()
d['H@10'] = np.mean(np.where(ranks < 11, 1, 0)).item()
```

實驗完的重要程度

A plausible explanation for this is that knowledge distillation improves performance by reducing the divergence between the re-ranker and the teacher, but ensembling can already achieve a similar effect by simply increasing the weight of the teacher in the final prediction. We observe that

Impact of Sparsity

observe the outsized impact of sparsity on models that do not utilize textual information

	FB15k-237					FB15k-237-Sparse				
	MR	MRR	H@1	H@3	H@10	MR	MRR	H@1	H@3	H@10
DistMult [♠]	–	.343	–	–	.531	3061	.136	.092	.146	.223
ComplEx [♠]	–	.348	–	–	**.536**	3333	.132	.091	.143	.216
ConvE [♠]	–	.339	–	–	.521	2263	.156	.106	.165	.258
ConvTransE [♢]	–	.33	.24	.37	.51	2285	.153	.103	.161	.255
BERT-ConvE	193	.305	.224	.330	.465	408	.190	.128	.200	.315
BERT-ConvTransE	211	.296	.218	.321	.449	**390**	.188	.127	.199	.310
BERT-DeepConv	190	.327	.246	.354	.488	422	.188	.127	.197	.314
BERT-ResNet	186	.346*	.262	.379	.514	413	.191*	.128	.201	**.317**
+ Re-ranking	187	.304	.212	.329	.514	413	.190	.128	.200	**.317**
+ Knowledge Distillation (KD)	187	.310	.220	.334	.514	413	.197†	.135	.209	**.317**
+ Ranking Ensemble (RE)	**186**	**.354**†	**.270**	**.387**	.514	413	**.199**†	**.137**	.210	**.317**
+ KD and RE	**186**	.353†	.269	.386	.514	413	.198†	.136	**.211**	**.317**

提出讀這篇論文的疑慮

Question: 架構不好