Fine-tuning BERT for Ontology Alignment

Yuan He

Department of Computer Science

University of Oxford

yuan.he@cs.ox.ac.uk

BERT: The Masked Language Model

- Transformer encoder:
  - Text sequences → **contextual embeddings.**
- Pre-training:
  - **Masked language modelling.**
  - ~~Next sentence prediction.~~ RoBERTa
- Fine-tuning:
  - Using the sentence head **[CLS]**.
  - Customised downstream layers.
    - E.g., Linear+Activation

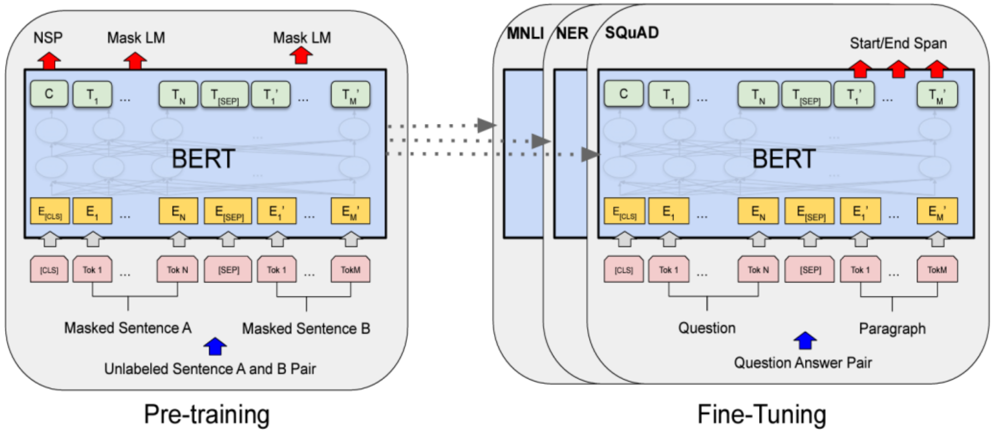
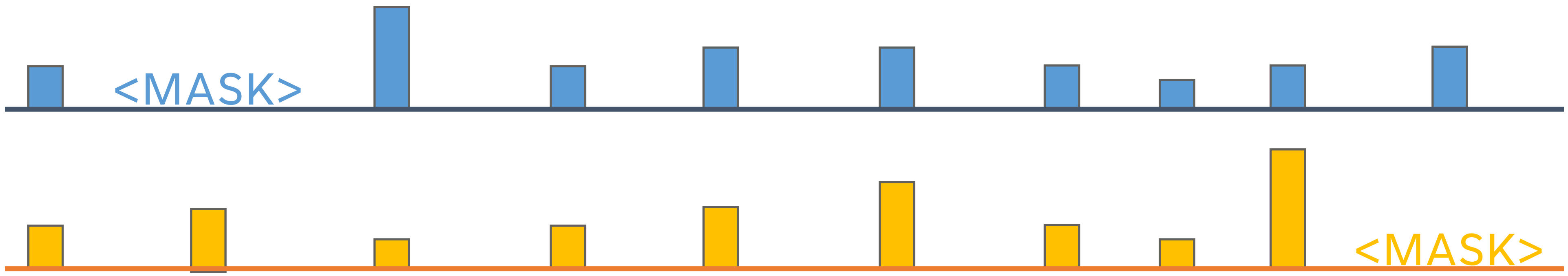


Figure: Illustration of BERT architecture by Delvin et al. (NAACL'19).

Why BERT yields contextual embeddings?

The bank robber was seen fishing on the river bank.



Fine-tuning Approaches

- Sentence head [CLS]:
  - BERT input format: [CLS] text_1 [SEP] text_2 [SEP] ...
  - Embedding of [CLS] fed into downstream layers.
  - Often require ≤ 5 epochs for convergence.

- Embedding-based:
  - Contrastive learning.
  - $\uparrow \mathbf{sim}(e, e^{+})$ and $\downarrow \mathbf{sim}(e, e^{-})$; E.g., similarities of synonyms/antonyms.

Fine-tuning Approaches

- Prompt-based:
  - Formulate downstream tasks like **pre-training**.
  - Use **prompts** as additional contexts.
  - Map the **answers** to the final outputs.
  - The basis of recent **Large Language Models** (LLMs).

- More techniques like adapters, distilling, ensemble...

Our Previous Works

- Two works that adopt **[CLS] fine-tuning** for ontology alignment.
- BERTMap [He et al. AAAI'22] for concept **equivalence matching**.
- BERTSubs [Chen et al. WWW Journal'23] for concept **subsumption matching**.

BERTMap

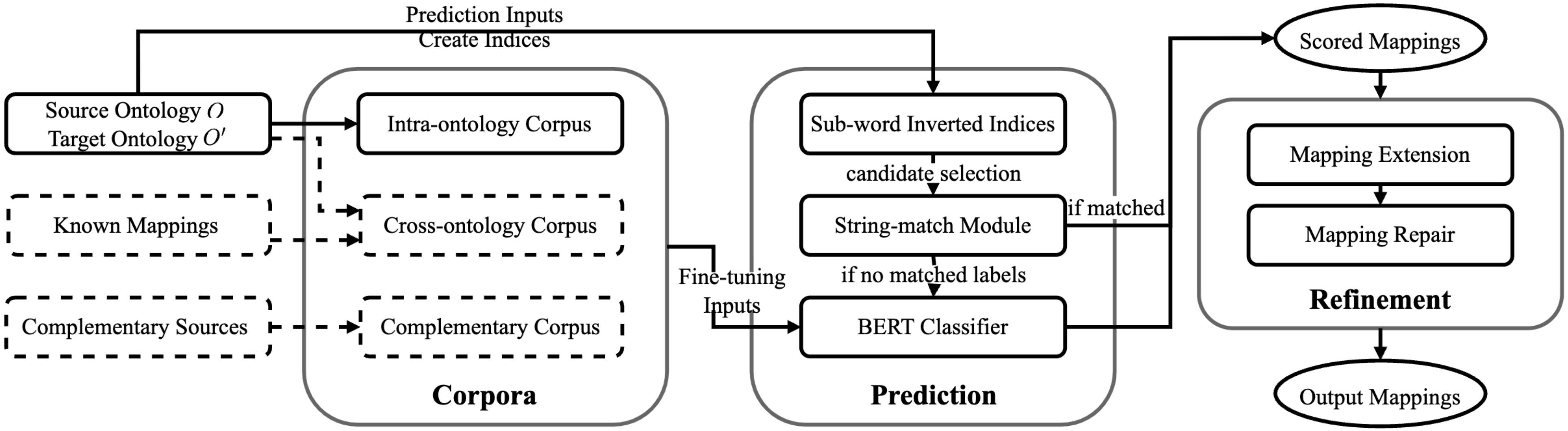


Figure: Illustration of the BERTMap system by He et al. (AAAI'22).

BERTMap

- Corpus construction:
  - **Synonyms** are labels from the **same** concepts.
  - **Non-synonyms** are labels from **different** concepts.
  - **Hard non-synonyms** are labels from **sibling** concepts.

- Fine-tuning BERT for classification:
  - Input format: [CLS] label_1 [SEP] label_2 [SEP]
  - **[CLS]** fed into downstream linear followed by a softmax activation.
  - Output a **binary score** indicating if two phrases are **synonymous or not**.

BERTMap

- Mapping prediction:
  - **Candidate selection** using a **sub-word inverted index**.
  - Compute a **mapping score** based on **aggregated synonym scores**.

- Mapping extension:
  - **Locality principle**: parents/children of aligned concepts could match.
  - **Iteratively search** new mappings with scores $\geq$ threshold.

- Mapping repair:
  - Remove a minimal set of mappings that cause **inconsistency**.

BERTMap

Highlights

- Support **unsupervised**, **semi-supervised**, and **data-augmented** modes.
- Simple and effective **candidate selection** with a **high recall**.
- Mostly improving **lexical** matching, but also consider **structural** (extension) and **logical** (repair) contexts.

BERTSubs

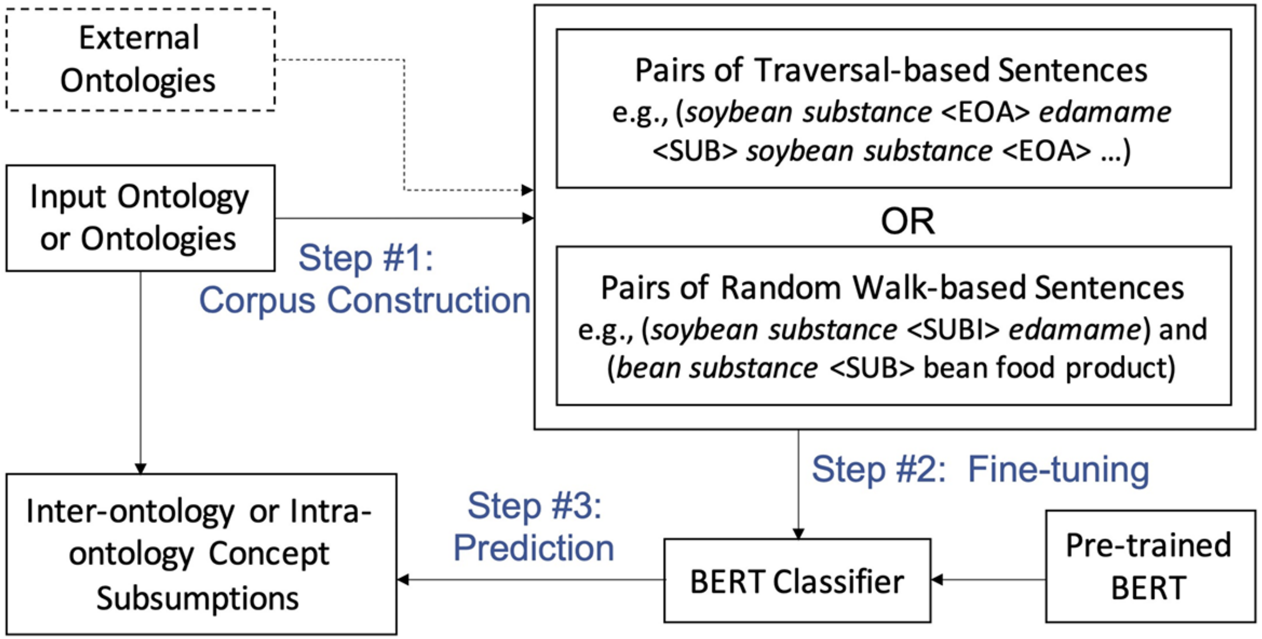


Figure: Illustration of the BERTSubs system by Chen et al. (WWWJ'23).

BERTSubs

- Corpus construction:
  - **Positive samples** are (C, C^{+})s from **asserted subsumptions**.
  - **Negative samples** are (C, C^{-})s from **non-entailed subsumptions**.

- Input template:
  - **Isolated concept**: only use concept labels.
  - **Path context**: concatenate **ancestors** as context.
  - **Breadth-first context**: concatenate **neighbours (BFS)** as context.

BERTSubs

Highlights

- Support both **intra-ontology (completion)** and **inter-ontology (alignment)** subsumption prediction.
- Explore different kinds of **structural contexts**.
- Consider not only named concepts but also **existential restrictions**.

Note

- Subsumption matching is usually evaluated with ranking-based metrics (e.g., Hits@K, MRR); search is not implemented for BERTSubs.

More Related Works

- **Bio-ML Track of the OAEI** [He et al. ISWC'22]: Five OM pairs for equivalence and subsumption matching.

- **OntoLAMA** [He et al. ACL'23 Findings] : Prompt-based subsumption inference for LM probing.

- **LLMap** [He et al. ISWC'23 Posters&Demos]: Preliminary investigation of applying LLMs on OM, also contributing to the **Bio-LLM sub-track** of Bio-ML.

THANKS!

Check everything mentioned in our Python package **DeepOnto**:

https://krr-oxford.github.io/DeepOnto/