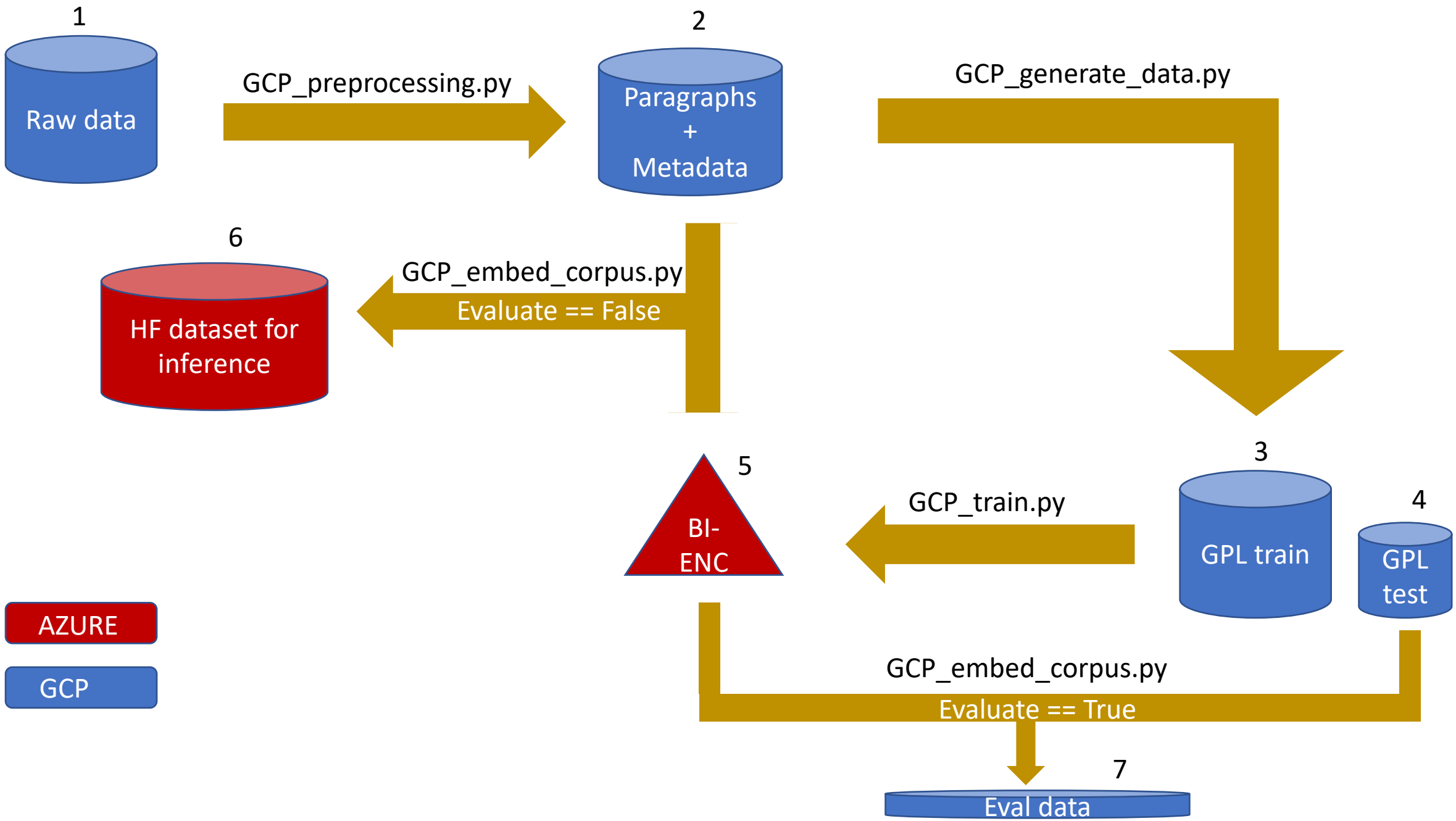
Semantic Search Overdracht


1
Raw data
GCP_preprocessing.py
2
Paragraphs
+
Metadata
GCP_generate_data.py
6
HF dataset for inference
GCP_embed_corpus.py
Evaluate == False
5
BI-ENC
GCP_train.py
3
GPL train
4
GPL test
AZURE
GCP
GCP_embed_corpus.py
Evaluate == True
7
Eval data

Stored where

1. **Raw data**
   GCP gsc-migrated-blobstore/raw/publicregister
2. **Paragraphs**
   semantic_search_blobstore/preprocessed_corpus
3. **GPL train data**
   semantic_search_blobstore/gpl_training_data/corpus_all_length_250/train
4. **GPL test data**
   semantic_search_blobstore/gpl_training_data/corpus_all_length_250/test
5. **Bi-encoder**
   SemanticSearch_Train/models
6. **HF inference dataset**
   SemanticSearch_Train/workspaceblobstore/corpus_ds_with_embedding
7. **Eval data**
   semantic_search_blobstore/corpus_ds_with_embedding/evaluation/corpus_all_length_250_eval/multi-qa-mpnet-base-dot-v1_ TRAINED_2022-10-21-14-33-13_version_45

*All GCP stuff in Rebatch-sandbox project

*corpus_all_length_250 is the latest corpus (used for GSC demo)

*multi-qa-mpnet-base-dot-v1_TRAINED_2022-10-21-14-33-13_version_45 it the latest model (used for GSC demo)

Preprocessing

- Start from raw json files → extract info for every paragraph
  - _id: *string*, important as unique_id for GPL training later
  - title: *string,* title of document containing paragraph
  - doc_id: *string,* id of document containing paragraph (e.g. ST …. INIT)
  - passage_id: *integer*, paragraph number within document
  - text: *string,* paragraph text
  - chunked: *bool,* if True, paragraph was further split into pieces after initial document split on '/n/n' (not entirely relevant anymore I think)

- NOTES
  - To change max tokens (~words) per paragraph change max_length under split_paragraph()
  - Slow on GCP for large corpora: speed up by doing different parts in parallel and stitching back together with merge_jsonfiles_gcp.py
  - Lots of text cleaning + check for bad_paragraph because text in jsons extracted from pdf and often very low quality

GPL generate data

Generate training data from preprocessed paragraphs

- Generate GPL data and training was 1 single script originally.
- Original script many, many parameters, most of default in training pipeline should be okay. Important parameters to specify yourself are in GPL/gpl/gpl_args.yml.
- Original script only adapted by introducing custom functions
  - prepend_title: prepend title to paragraph text
  - copy_corpus_and_filter: use fraction of all available documents and keep ‘good’ ones
  - postprocess_queries: Select fraction of all ‘best’ synthetically generated questions
  - upload_train_test_split: upload train and test data to separate folders

GPL generate data

- Structure of output folder important as train script will look for location of specific files
  - corpus.jsonl -> selection of entire preprocessed corpus used for generating queries
  - gpl-training-data.tsv -> triplets of query_id, positive_id, hard-negative_id and cross-score (effective training lines)
  - hard-negatives.jsonl -> for every query_id: positive_id and list of all hard-negatives_id
  - qrels
    - train.tsv (or test.tsv for test data) -> for every query_id: positive_id
  - qgen-queries.jsonl -> query_id and query text

GPL train

Start training from synthetic generated data

- Generate GPL data and training was 1 single script originally.

- Original script many, many parameters, most of default in training pipeline should be okay. Important parameters to specify yourself are in GPL/gpl/gpl_args.yml.
- Make sure gpl_score_function corresponds to correct model!
  - e.g. ‘dot’ if using ‘multi-qa-mpnet-base-dot’

- Original script only adapted by introducing custom functions
  - evaluation: compute precision, recall and mAP on test data
  - save_callback: save model on azure with evaluation details
- Tricky to know how many training steps
  - steps = training_lines/batch_size
  - training_lines = queries_per_paragraph*negatives_per_paragraph*number_of_paragraphs

Embed corpus

Use trained bi-encoder to transform entire corpus into dataset with embeddings

- Evaluation == False: Transform entire corpus to dataset and upload to Azure
- Evaluation == True: Only embed data select for train+test,
  in the follwing step, use this smaller dataset for evaluation

NOTES:

- If evaluation==True: checks automatically if dataset already exists, if so, only do evaluation step
- Possibility to prepend title to paragraph only for embedding
- Currently no FAISS index added as flat one is used