nlp-question-answering

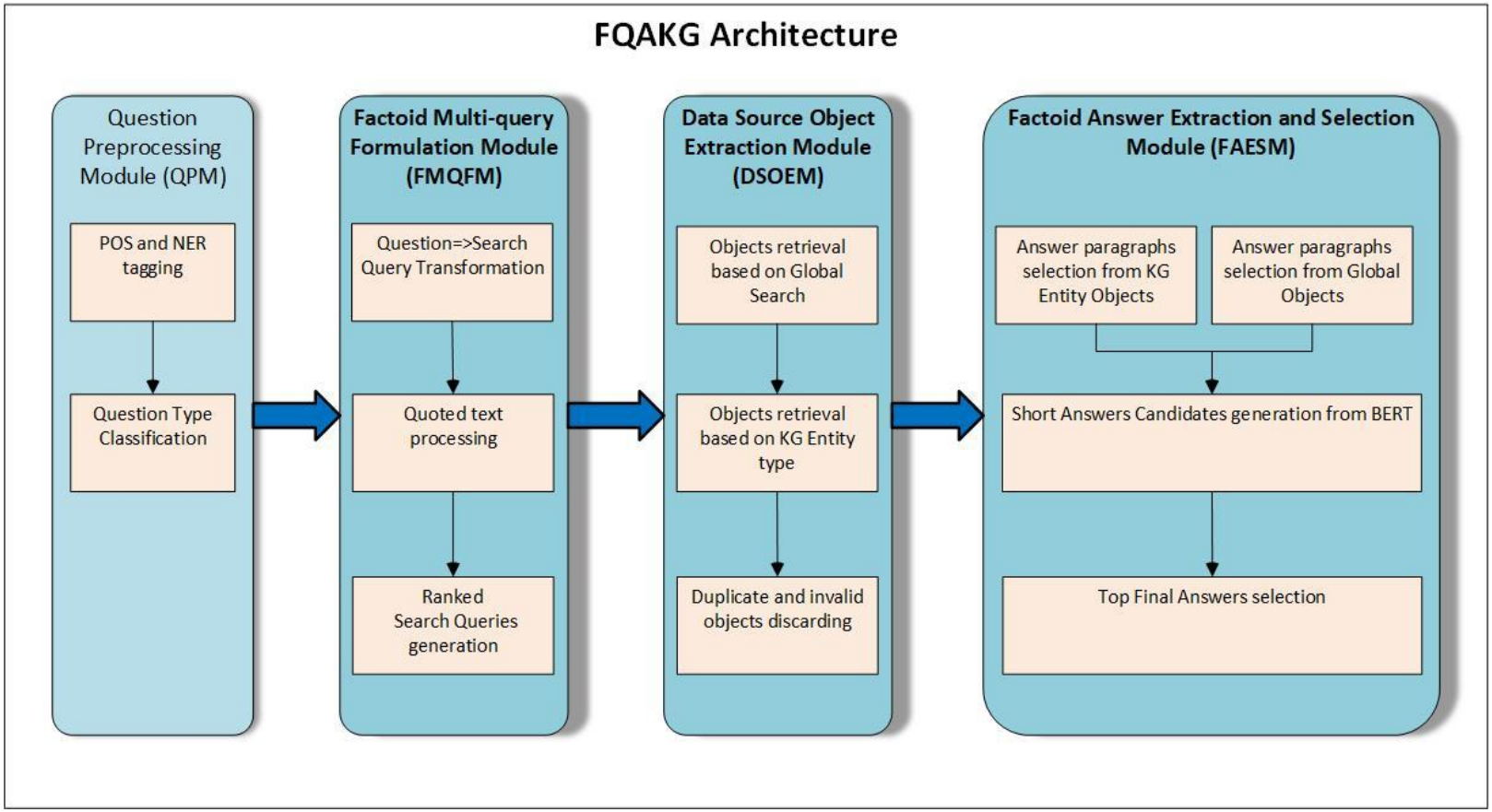
Overview

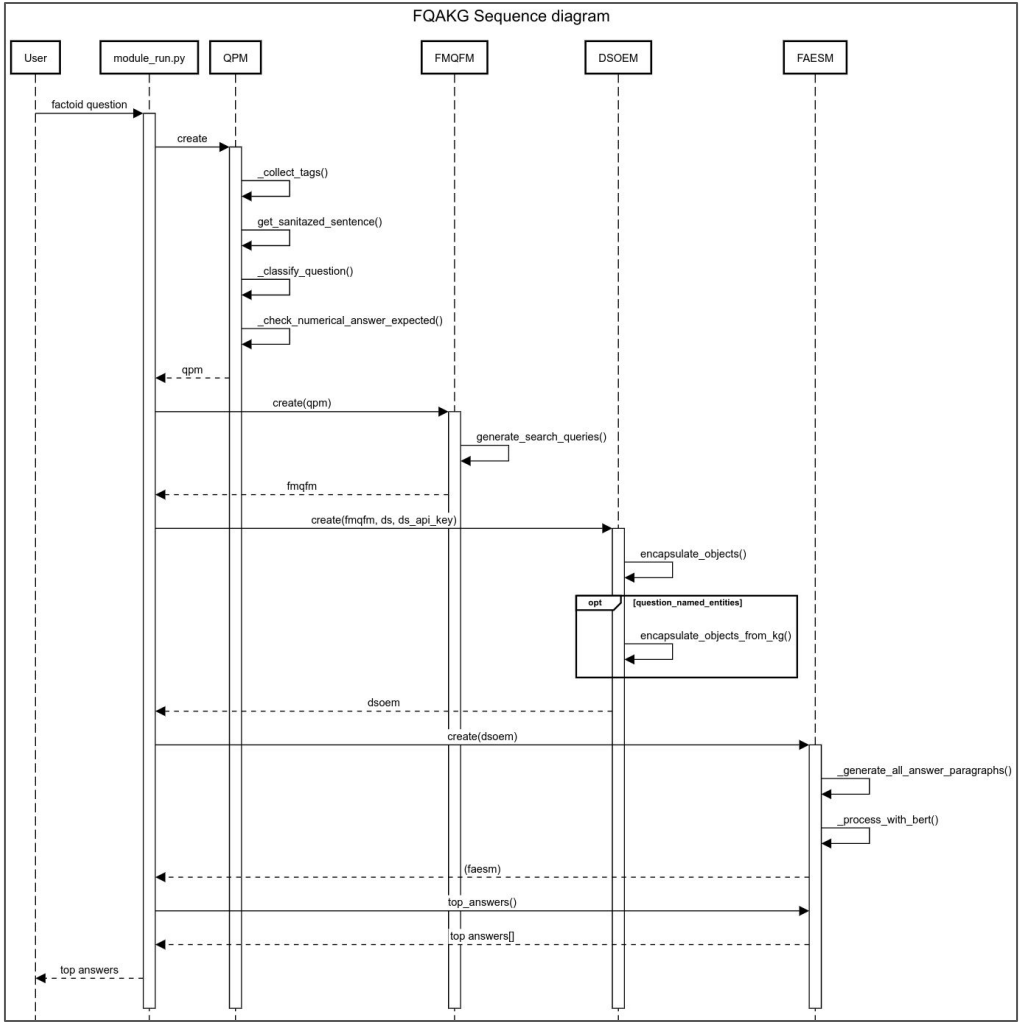
Eduard Kegulskiy
ekegulsk@mail.sfsu.edu

What is nlp-question-answering?

- Automated Factoid Question Answering using Knowledge Graphs (FQAKG)
- Leverages large Knowledge Graphs (KG) and Deep Learning techniques to identify concise answers to factoid-type questions from any domain.
    - Uses Google KG or Diffbot KG – online Web Services that provide API for accessing Web data as structured entities - to search for related content.
    - Uses Deep Learning system BERT – one of Google's latest advancements in NLP - to select the most probable short answers from the search results.
- Designed to provide learning experience for building Question Answering systems, by allowing to run individual modules of the Question Answering pipeline and swap them with alternative implementations.

System Architecture


FQAKG Architecture
Question Preprocessing Module (QPM)
POS and NER tagging
Question Type Classification
Factoid Multi-query Formulation Module (FMQFM)
Question=>Search Query Transformation
Quoted text processing
Ranked Search Queries generation
Data Source Object Extraction Module (DSOEM)
Objects retrieval based on Global Search
Objects retrieval based on KG Entity type
Duplicate and invalid objects discarding
Factoid Answer Extraction and Selection Module (FAESM)
Answer paragraphs selection from KG Entity Objects
Answer paragraphs selection from Global Objects
Short Answers Candidates generation from BERT
Top Final Answers selection


FQAKG Sequence diagram
User
module_run.py
QPM
FMQFM
DSOEM
FAESM
factoid question
create
_collect_tags()
get_sanitazed_sentence()
_classify_question()
_check_numerical_answer_expected()
qpm
create(qpm)
generate_search_queries()
fmqfm
create(fmqfm, ds, ds_api_key)
encapsulate_objects()
opt
[question_named_entities]
encapsulate_objects_from_kg()
dsoem
create(dsoem)
_generate_all_answer_paragraphs()
_process_with_bert()
(faesm)
top_answers()
top answers[]
top answers

Code Structure

Main pipeline modules

- **module_run.py** - main executable script to connect and run the system
- **QPM.py** - **Q**uestion **P**re-processing **M**odule
- **FMQFM.py** - **F**actoid **M**ulti-**Q**uery **F**ormatuation **M**odule
- **DSOEM.py** - **D**ata **S**ource **O**bject **E**xtraction **M**odule
- **FAESM.py** - **F**actoid **A**nswer **E**xtraction and **S**election **M**odule

Data source modules

- data_source folder
  - **DataSourceObject.py** - base class representing a data source object (i.e. object containing various search results properties such as text, url, language, etc..)
  - google_kg_client folder
    - **GKGAPI.py** - class that implements search using Google KG APIs
    - **GKG_Content.py** - classes to convert Google KG results into a list of DataSourceObject objects
  - sfsu_diffbot folder - collection of classes to convert Diffbot KG results into a list of DataSourceObject objects

Utility modules

- qa-utils.py - utility classes
  - **KGQAPOSTagger** - Parts-of-Speech (POS) and Named Entity Recognition (NER) taggers, based on Stanford Taggers in nltk
  - **KGQANumberDetector** - Detects whether a given text contains a number in either digit form or textual form
- setup_env.py - helper script to download dependencies such as Stanford taggers, etc..