`Function Assistant`: A Tool for NL Querying of APIs

Kyle Richardson and **Jonas Kuhn**
University of Stuttgart, Germany
kyle@ims.uni-stuttgart.de

Institut für **Maschinelle Sprachverarbeitung**

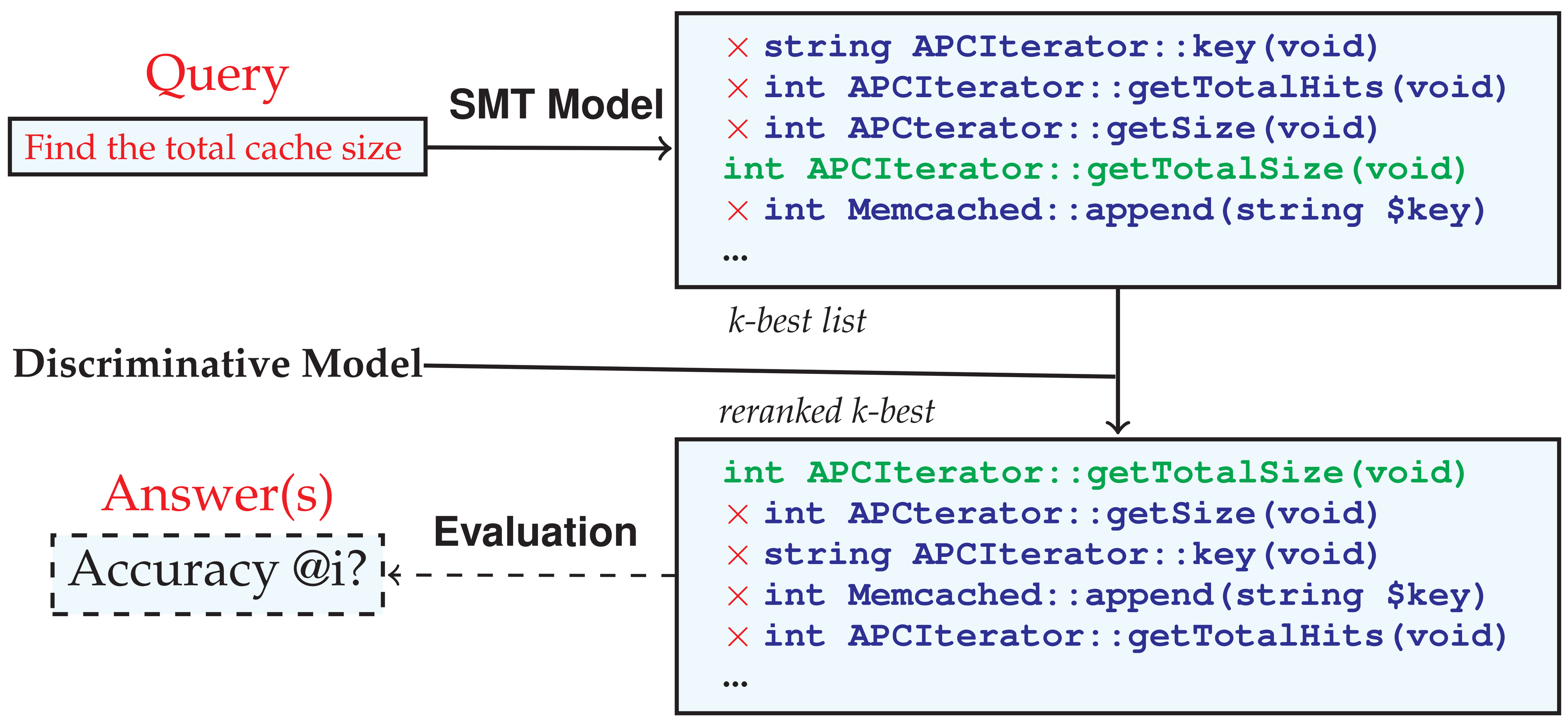
Overview: Towards Automated Question-Answering on Software Libraries (APIs)

- **Source Libraries**: Collections of code with developer documentation (red), sometimes hard to navigate and find functionality:
  - **Parallel Data:** (*Short text description*, `Function signature`)
- **API QA:** Retrieving code components, e.g., function signatures, from high-level natural language descriptions and queries.
  - **Query** *How do I create a dependency arc?* → `target function(s).`
- **Function Assistant**: End-to-end dataset and QA builder for arbitrary APIs, works by training translation model on parallel data.

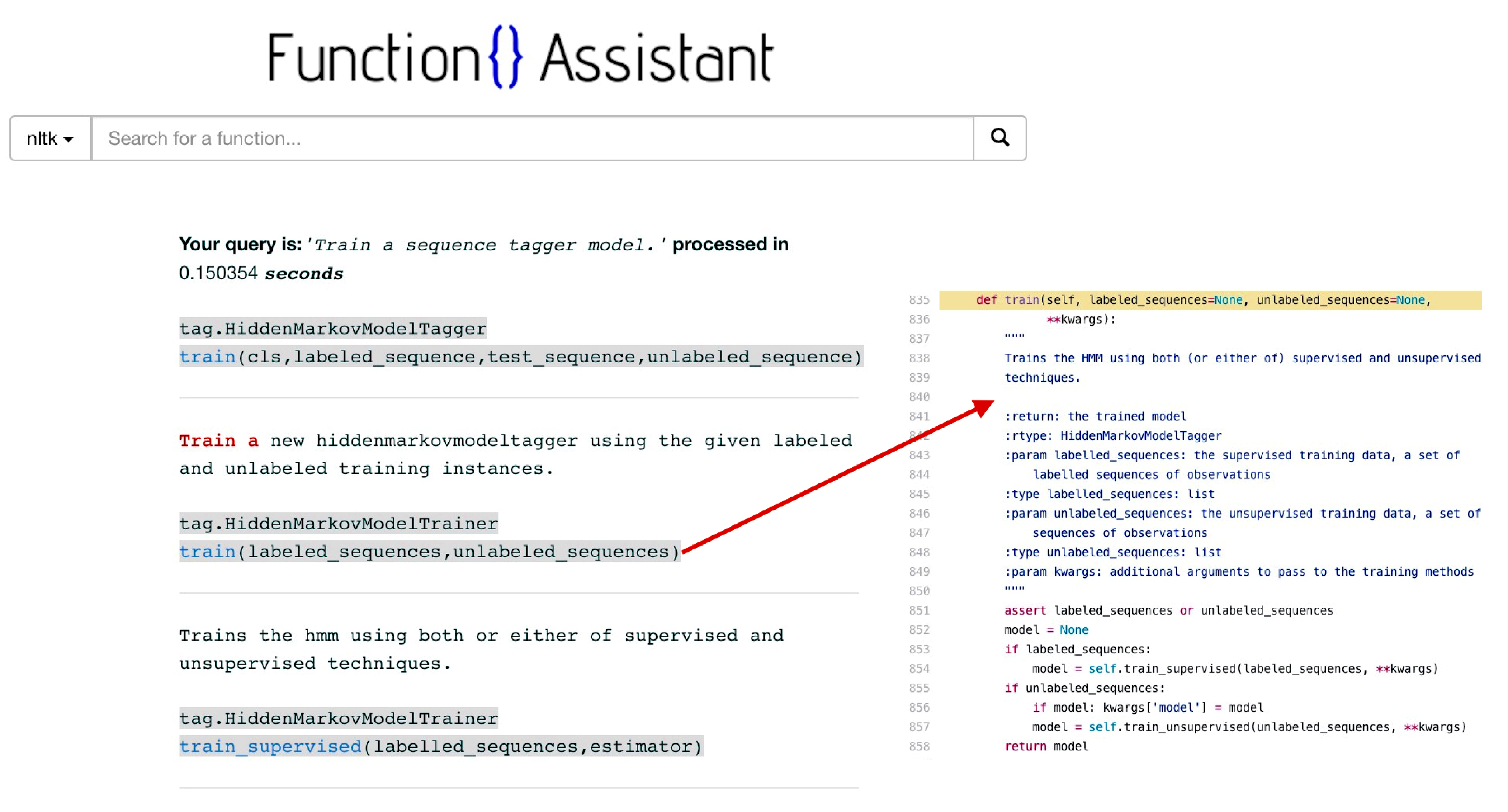
```
## from nltk.parse.dependencygraph.py

class DependencyGraph(object):
  """A container ....for  a dependency structure"""

  def remove_by_address(self, address):
    """
    Removes the node with the given address.
    """
     # => implementation

  def add_arc(self, head_address, mod_address):
    """Adds an arc from the node specified by
    head_address to the node specified by
    the mod address....
    """
```

Dataset Extractor and Translation Pipeline

- **Function Assistant Pipeline**: Raw (Python) Source Code Library → Parallel Dataset → SMT model → Query Server.
- SMT model has two components (Richardson and Kuhn 2017):
  - **Word SMT Model:** Generates candidate function signatures from input, uses simple decoding strategy.
  - **Discriminative Model**: Ranks translation candidates using additional word, phrase and API-level features.
- **Implementation:** Py/Cython, dependency-injection design, customizable, can be used to implement new models or as baseline.



Query Server Illustration



- **API QA:** Takes a natural language query, runs through trained SMT model, returns reranked list as possible answers to query, provides pointers to implementation.

Experiment: PY27 Dataset

Project	# Pairs	# Symbols	# Words	Vocab.
scapy	757	1,029	7,839	1,576
zipline	753	1,122	8,184	1,517
biopython	2,496	2,224	20,532	2,586
renpy	912	889	10,183	1,540
pyglet	1,400	1,354	12,218	2,181
kivy	820	861	7,621	1,456
pip	1,292	1,359	13,011	2,201
twisted	5,137	3,129	49,457	4,830
vispy	1,094	1,026	9,744	1,740
orange	1,392	1,125	11,596	1,761
tensorflow	5,724	4,321	45,006	4,672
pandas	1,969	1,517	17,816	2,371
sqlalchemy	1,737	1,374	15,606	2,039
pyspark	1,851	1,276	18,775	2,200
nupic	1,663	1,533	16,750	2,135
astropy	2,325	2,054	24,567	3,007
sympy	5,523	3,201	52,236	4,777
ipython	1,034	1,115	9,114	1,771
orator	817	499	6,511	670
obspy	1,577	1,861	14,847	2,169
rdkit	1,006	1,380	9,758	1,739
django	2,790	2,026	31,531	3,484
ansible	2,124	1,884	20,677	2,593
statsmodels	2,357	2,352	21,716	2,733
theano	1,223	1,364	12,018	2,152
nltk	2,383	2,324	25,823	3,151
sklearn	1,532	1,519	13,897	2,115

- Ran our pipeline on 27 popular Python open-source projects from `github.com`, selection of *Awesome Python* list.
- **Goal**: Test robustness of extractor, provide new resource for system development and baseline results.

Synthetic QA Experiments and Baseline Results

- **QA Evaluation:** Hard to find real queries, using normal train/test split, treat held out data as synthetic user queries (Deng and Chrupała 2014, Richardson and Kuhn 2017)
  - `Data Extrator`: API → *train set*, *dev. set*, *test set* (= synthetic queries)
- Measure **Accuracy @1**, **Accuracy @10**, **MRR**, results conform to previous findings.

Method	scapy	zipline	biopython	renpy	pyglet	kivy	pip	twisted	vispy
BoW	00.0 51.3 17.4	01.7 38.3 12.9	05.8 54.8 20.4	06.6 41.1 16.6	05.7 52.3 19.2	07.3 53.6 22.0	06.2 40.9 17.1	06.6 38.8 16.9	07.3 48.7 18.6
Term Match	**21.2** 43.3 28.7	28.5 50.8 36.2	23.5 48.1 31.7	25.7 59.5 38.7	20.4 50.9 31.2	30.0 62.6 41.3	19.1 50.2 30.7	17.6 44.1 26.2	29.2 64.0 41.1
Translation	20.3 61.9 34.7	27.6 62.5 40.7	29.6 75.6 45.8	30.8 61.7 42.0	26.1 69.5 41.3	33.3 67.4 45.3	18.6 56.4 32.3	27.7 61.4 39.4	28.6 70.1 42.3
Reranker	**21.2 67.2 37.2**	**30.3 70.5 45.3**	**32.3 79.1 48.6**	**38.9 73.5 48.9**	**29.0 77.1 45.5**	**35.7 75.6 49.1**	**25.9 65.8 39.9**	**28.8 65.8 42.2**	**33.5 80.4 50.3**

Method	orange	tensorflow	pandas	sqlalchemy	pyspark	nupic	astropy	sympy	ipython
BoW	13.4 60.5 29.1	09.4 47.4 21.2	03.7 40.6 15.6	07.3 45.0 18.4	07.5 50.9 20.8	06.4 55.0 22.8	07.7 52.0 21.1	06.4 44.4 18.5	01.9 41.2 13.9
Term Match	37.9 69.7 49.3	25.2 48.7 33.5	19.3 43.7 27.9	17.3 48.4 26.6	20.5 46.9 29.1	23.6 51.0 33.1	26.1 49.1 34.3	20.2 44.9 28.8	23.8 56.7 33.8
Translation	40.3 78.3 54.0	35.3 71.5 48.0	29.1 62.7 41.0	28.8 70.3 43.0	37.1 78.7 52.1	**30.9** 69.8 44.6	30.7 66.6 43.4	**32.8** 70.2 45.5	24.5 59.3 36.5
Reranker	**45.1 84.1 59.9**	**38.4 77.7 51.8**	**31.1 66.1 43.1**	**35.0 76.1 49.7**	**41.5 81.5 55.3**	29.3 **76.7 45.6**	**33.9 74.4 47.4**	32.1 **75.0 46.6**	**29.6 66.4 42.3**

Method	orator	obspy	rdkit	django	ansible	statsmodels	theano	nltk	sklearn
BoW	10.6 66.3 28.6	06.7 49.5 20.2	05.3 40.6 17.1	04.5 40.9 16.2	17.9 55.3 30.5	05.6 46.1 18.6	03.2 43.7 16.2	05.0 44.2 16.3	05.2 45.8 17.7
Term Match	31.9 64.7 43.7	19.9 46.6 30.0	13.3 46.6 23.9	19.3 48.0 29.1	24.8 54.0 35.8	16.7 39.9 25.1	16.3 37.1 24.0	19.8 45.6 28.4	24.4 50.6 32.5
Translation	**32.7** 79.5 47.5	33.8 75.8 48.3	**25.3** 60.6 37.2	22.9 57.8 34.6	35.5 71.6 47.5	25.4 64.8 37.8	26.2 58.4 37.8	28.2 68.0 41.5	27.9 67.6 41.3
Reranker	**32.7 82.7 49.7**	**37.7 80.0 52.3**	**25.3 63.3 39.6**	**25.8 64.5 39.4**	**40.5 77.0 53.1**	**28.8 69.1 41.7**	**27.3 66.1 39.9**	**31.6 72.5 45.7**	**29.2 75.5 44.5**

References and Info

- GPL-licensed, data and code can be found via github: `https://github.com/yakazimir/Code-Datasets`
- **Code retrieval prototype**: `zubr.ims.uni-stuttgart.de`
- Supported by the German Research Foundation (DFG), project D2 of SFB 732.

Huijing Deng and Grzegorz Chrupała. 2014. **Semantic Approaches to Software Components Retrieval with English Queries.** Proceedings of LREC.
Kyle Richardson and Jonas Kuhn, 2017 **Learning Semantic Correspondences in Technical Documentation.** Proceedings of ACL