IBM Watson

Ranking Features

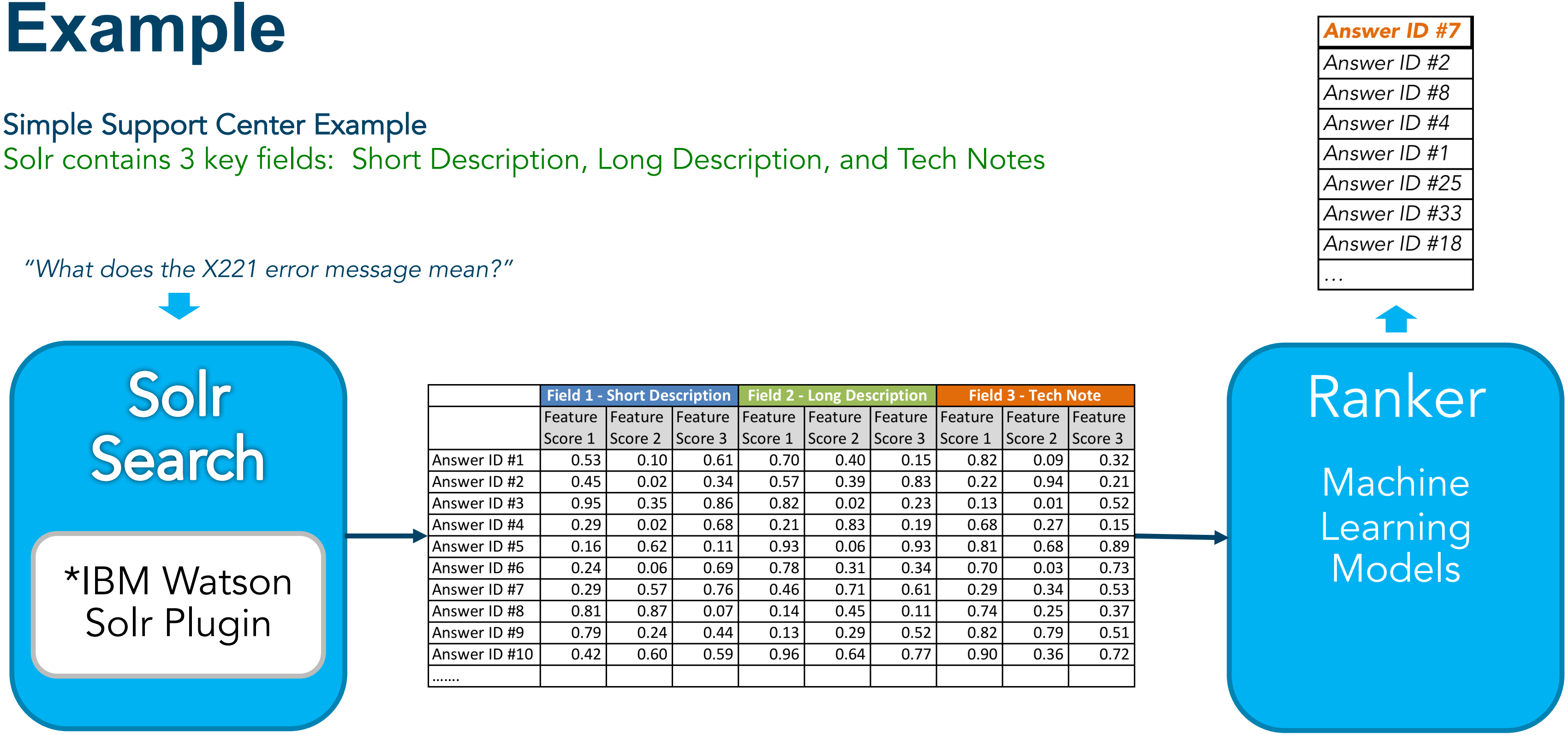
Custom Feature Injection

Example

Simple Support Center Example

Solr contains 3 key fields: Short Description, Long Description, and Tech Notes

"What does the X221 error message mean?"



Solr Search

*IBM Watson Solr Plugin

	Field 1 - Short Description			Field 2 - Long Description			Field 3 - Tech Note		
	Feature Score 1	Feature Score 2	Feature Score 3	Feature Score 1	Feature Score 2	Feature Score 3	Feature Score 1	Feature Score 2	Feature Score 3
Answer ID #1	0.53	0.10	0.61	0.70	0.40	0.15	0.82	0.09	0.32
Answer ID #2	0.45	0.02	0.34	0.57	0.39	0.83	0.22	0.94	0.21
Answer ID #3	0.95	0.35	0.86	0.82	0.02	0.23	0.13	0.01	0.52
Answer ID #4	0.29	0.02	0.68	0.21	0.83	0.19	0.68	0.27	0.15
Answer ID #5	0.16	0.62	0.11	0.93	0.06	0.93	0.81	0.68	0.89
Answer ID #6	0.24	0.06	0.69	0.78	0.31	0.34	0.70	0.03	0.73
Answer ID #7	0.29	0.57	0.76	0.46	0.71	0.61	0.29	0.34	0.53
Answer ID #8	0.81	0.87	0.07	0.14	0.45	0.11	0.74	0.25	0.37
Answer ID #9	0.79	0.24	0.44	0.13	0.29	0.52	0.82	0.79	0.51
Answer ID #10	0.42	0.60	0.59	0.96	0.64	0.77	0.90	0.36	0.72
…….									

Ranker

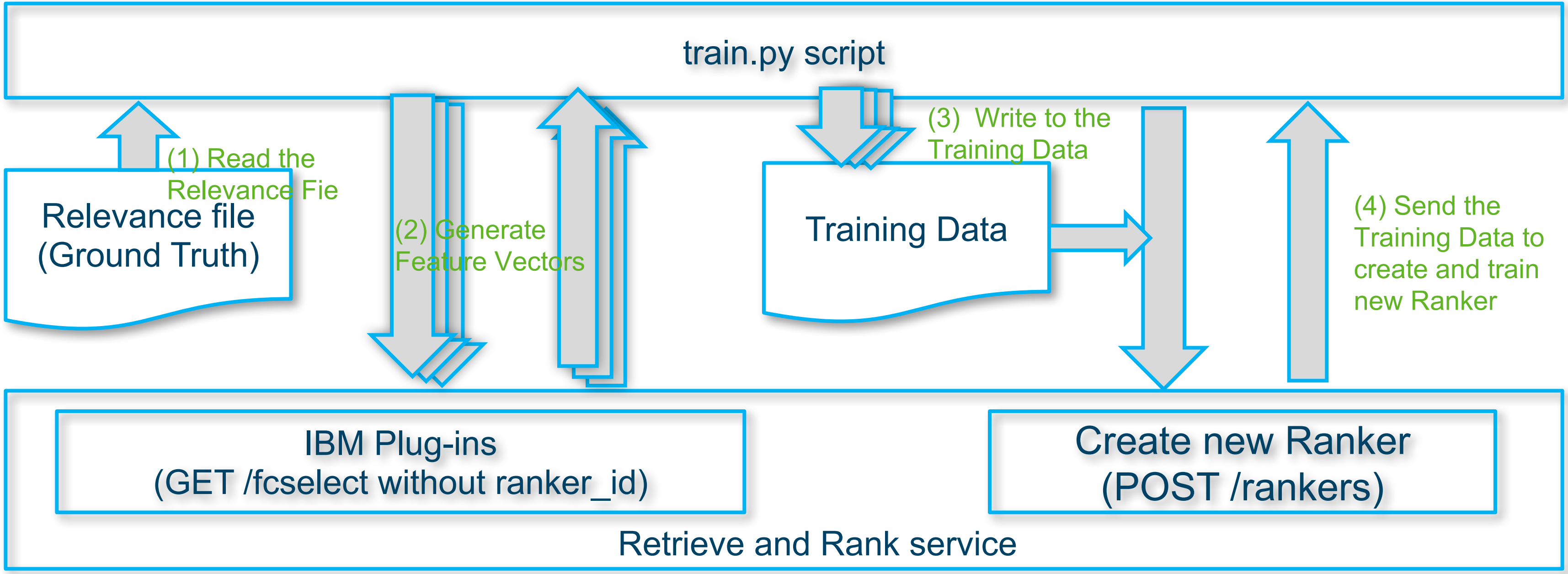
Machine Learning Models

Answer ID #7
Answer ID #2
Answer ID #8
Answer ID #4
Answer ID #1
Answer ID #25
Answer ID #33
Answer ID #18
…

*IBM Watson Solr Plugin contains a set of 'scorers' that scores each data attribute (Feature Score)

Training Data - Creation Script

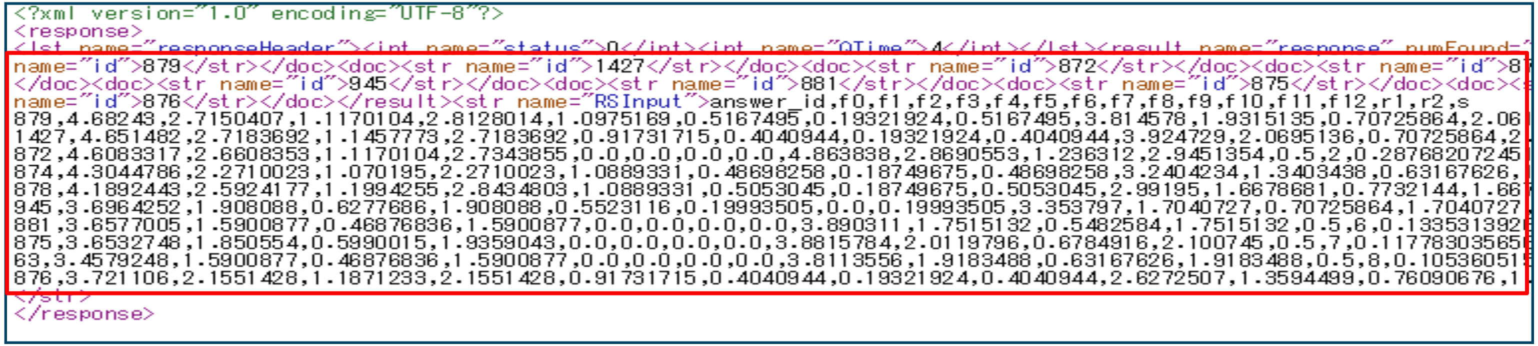
- train.py script helps you to create Training Data and upload it for creating and training new Ranker.



train.py script
(1) Read the Relevance Fie
Relevance file (Ground Truth)
(2) Generate Feature Vectors
(3) Write to the Training Data
Training Data
(4) Send the Training Data to create and train new Ranker
IBM Plug-ins (GET /fcselect without ranker_id)
Create new Ranker (POST /rankers)
Retrieve and Rank service

Runtime Search - Separate Processes of R&R

- The Search of Retrieve and Rank service is done by IBM plugged-in Query Parser and the result is different with standard Solr search by "/select"
- To mimic the internal process of the Retrieve and Rank service, follows
  1. Call "/fcselect" without ranker_id, but with the following parameters to generate feature vector.
  2. Save the value of <str name="RSInput"> of the response into CSV file (retrieved_answers.csv).
  3. Call ranker to re-rank the retrieved result in the CSV



```
<?xml version="1.0" encoding="UTF-8"?>
<response>
<lst name="responseHeader"><int name="status">0</int><int name="QTime">4</int></lst><result name="response" numFound="
name="id">879</str></doc><doc><str name="id">1427</str></doc><doc><str name="id">872</str></doc><doc><str name="id">8
</doc><doc><str name="id">945</str></doc><doc><str name="id">881</str></doc><doc><str name="id">875</str></doc><doc><s
name="id">876</str></doc></result><str name="RSInput">answer_id,f0,f1,f2,f3,f4,f5,f6,f7,f8,f9,f10,f11,f12,r1,r2,s
879,4.68243,2.7150407,1.1170104,2.8128014,1.0975169,0.5167495,0.19321924,0.5167495,3.814578,1.9315135,0.70725864,2.06
1427,4.651482,2.7183692,1.1457773,2.7183692,0.91731715,0.4040944,0.19321924,0.4040944,3.924729,2.0695136,0.70725864,2
872,4.6083317,2.6608353,1.1170104,2.7343855,0.0,0.0,0.0,0.0,4.863838,2.8690553,1.236312,2.9451354,0.5,2,0.287682072450
874,4.3044786,2.2710023,1.070195,2.2710023,1.0889331,0.48698258,0.18749675,0.48698258,3.2404234,1.3403438,0.63167626,
878,4.1892443,2.5924177,1.1994255,2.8434803,1.0889331,0.5053045,0.18749675,0.5053045,2.99195,1.6678681,0.7732144,1.66
945,3.6964252,1.908088,0.6277686,1.908088,0.5523116,0.19993505,0.0,0.19993505,3.353797,1.7040727,0.70725864,1.7040727
881,3.6577005,1.5900877,0.46876836,1.5900877,0.0,0.0,0.0,0.0,3.890311,1.7515132,0.5482584,1.7515132,0.5,6,0.13353139
875,3.6532748,1.850554,0.5990015,1.9359043,0.0,0.0,0.0,0.0,3.8815784,2.0119796,0.6784916,2.100745,0.5,7,0.1177830365
63,3.4579248,1.5900877,0.46876836,1.5900877,0.0,0.0,0.0,0.0,3.8113556,1.9183488,0.63167626,1.9183488,0.5,8,0.10536051
876,3.721106,2.1551428,1.1871233,2.1551428,0.91731715,0.4040944,0.19321924,0.4040944,2.6272507,1.3594499,0.76090676,1
</str>
</response>
```

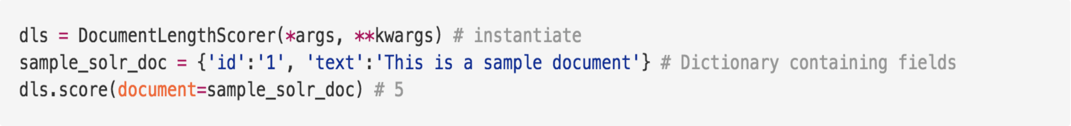
https:// {username}:{password}@gateway.watsonplatform.net/retrieve-and-rank/api/ v1/solr_clusters/{solr_cluster_id}/solr/kyotoCollection/fcselect?q={your question}&returnRSInput=true&generateHeader=true&fl=id

curl -X POST -u "{username}":"{password}" -F "answer_data=@retrieved_answers.csv" "https://gateway.watsonplatform.net/retrieve-and-rank/api/v1/rankers/{ranker_id}/rank"

Custom Features

- Retrieve and Rank features provide out of the box scores for lexical overlap.
- Custom features can define scores along other facets, gives the ranker one more feature to look at
  - i.e Number of views a document has
  - Ranker would learn to judge when this new feature is predictive of relevance.
- Requires separating retrieve and rank to inject custom features or define a proxy application.
  - Train the ranker through the proxy
  - Call the proxy at runtime for retrieve and rank joint operations

Types of Custom Feature / Scores



```
dls = DocumentLengthScorer(*args, **kwargs) # instantiate
sample_solr_doc = {'id':'1', 'text':'This is a sample document'} # Dictionary containing fields
dls.score(document=sample_solr_doc) # 5
```

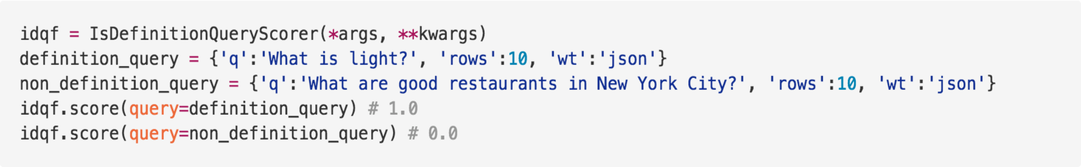
- **DocumentScorer**

- Class whose input is a python dictionary whose keys correspond to the field entries for a single Solr document.

- Examples:
  - DocumentLengthScorer, the number of words in the text field of a Solr Document.
  - DocumentViewsScorer, the number of views a document has

Types of Custom Feature / Scores

- **QueryScorer**



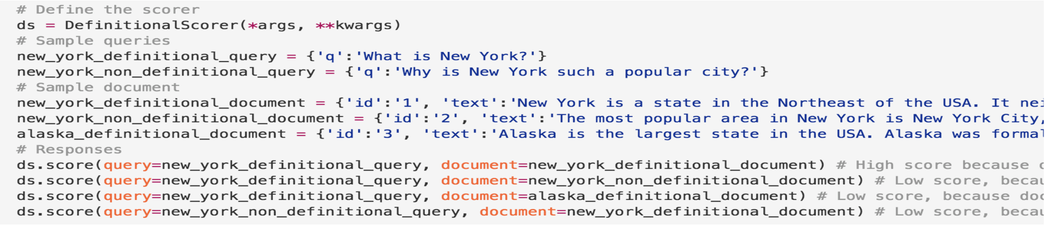
```
idqf = IsDefinitionQueryScorer(*args, **kwargs)
definition_query = {'q':'What is light?', 'rows':10, 'wt':'json'}
non_definition_query = {'q':'What are good restaurants in New York City?', 'rows':10, 'wt':'json'}
idqf.score(query=definition_query) # 1.0
idqf.score(query=non_definition_query) # 0.0
```

- Class whose input is a python dictionary whose keys correspond to the query params for a Solr query.

- Examples:
  - IsDefinitionQueryScorer, that scores queries based on whether it thinks the underlying query text is asking for a 'definition’
  - OnTopicScorer, scores if the question is on topic and to what extent it is on topic

Types of Custom Feature / Scores

- **QueryDocumentScorer**



```
# Define the scorer
ds = DefinitionalScorer(*args, **kwargs)
# Sample queries
new_york_definitional_query = {'q':'What is New York?'}
new_york_non_definitional_query = {'q':'Why is New York such a popular city?'}
# Sample document
new_york_definitional_document = {'id':'1', 'text':'New York is a state in the Northeast of the USA. It nei
new_york_non_definitional_document = {'id':'2', 'text':'The most popular area in New York is New York City,
alaska_definitional_document = {'id':'3', 'text':'Alaska is the largest state in the USA. Alaska was formal
# Responses
ds.score(query=new_york_definitional_query, document=new_york_definitional_document) # High score because c
ds.score(query=new_york_definitional_query, document=new_york_non_definitional_document) # Low score, becau
ds.score(query=new_york_definitional_query, document=alaska_definitional_document) # Low score, because doc
ds.score(query=new_york_non_definitional_query, document=new_york_definitional_document) # Low score, becau
```

- Class whose is 1) a python dictionary whose keys correspond to the query params for a Solr query, and 2) a python dictionary whose keys correspond to the field entries in a Solr document.

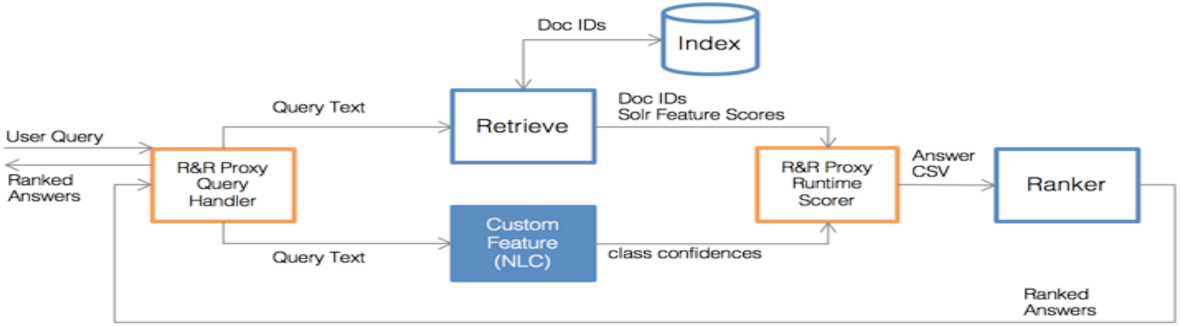
- Examples:
  - Extent to which the "text" of a Solr document answers definitional questions. More specifically, the scorer will 1) identify if a query is asking for a definition and 2) if so, identify whether the document contains a likely definition or not.

Query Boosting w/runtime feature injection

For some R&R implementations, the native lexical features within documents are sufficient, but adding more domain knowledge through custom features often increases answer relevance and specificity. We can achieve this by extracting additional features not directly supported by Solr to inject additional cognitive training to our R&R system. This is an advanced feature that should likely be added after the other basic R&R implementation is in place.

There are many custom features we could create for R&R implementations but they fall into 1 of 3 categories: document, query, and query+document scorers. This blog post provides more details on injecting your own custom features:

https://medium.com/machine-learning-with-ibm-watson/developing-with-ibm-watson-retrieve-and-rank-part-3-custom-features-826fe88a5c63#.9hybpgj5p



Lexical Answer Type

NLC Feature Example

- **What are the symptoms of Appendicitis?**

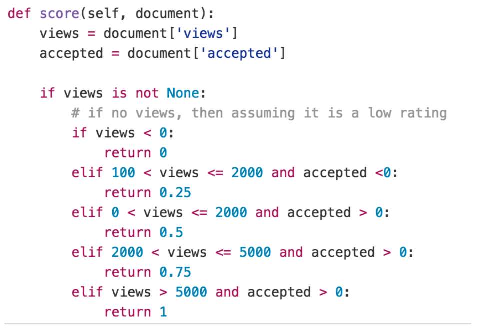
- Step 1: Classify Question
- Step 2: Retrieve
- Step 3: Score
  - Assign the NLC confidence for the class that matches the doc_type of that document as the score for our custom feature

```
“top_class” : “condition_symptom”,

“classes” : [ {“class_name” : “condition_symptom”,“confidence” :
0.9526760239113281}, {

“class_name” : “condition_complications”,“confidence” :
0.01749617382185962}, {

“class_name” : “condition_cause”,“confidence” :
0.011224721995999037}, {

“class_name” : “condition_diagnosis”,“confidence” :
0.006870593080181918}, {

“class_name” : “condition_prevention”,“confidence” :
0.004564829920859066}, {

“class_name” : “condition_definition”,“confidence” :
0.0036075458415429787}, {

“class_name” : “condition_treatment”,“conf
0.003560114282294262} ]

}
```

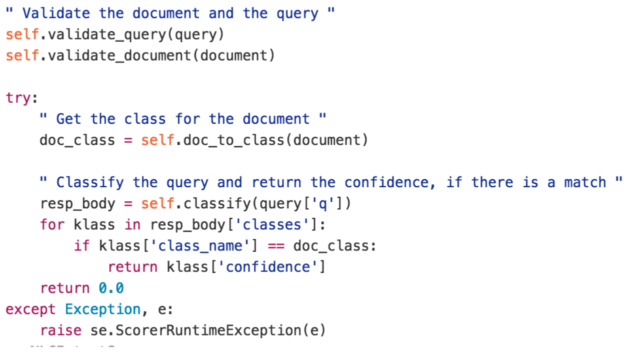
```
Doc 1:

{
 “id”: “a9e69b96–099e–4a02–b1ae–96a0956c484b”,
 “source”: “Appendicitis”,
 “doc_type”: “symptom”,
 “topic”: “What are the symptoms of appendicitis?”,
 “text_description”: “The symptoms of appendicitis are typically
easy for a health care provider to diagnose. The most common symptom
of appendicitis is abdominal pain. Abdominal pain ...“
 }

Doc 2:

{
 “id”: “b5eae497–17df–4510–8c07–12d8e18bd6bc”,
 “source”: “Appendicitis”,
 “doc_type”: “definition”,
 “topic”: “What is appendicitis?”,
 “text_description”: “Appendicitis is inflammation of the appendix.
Appendicitis is the leading cause of emergency abdominal operations.
Spirt MJ. Complicated intra–abdominal infections: a focus on
appendicitis and diverticulitis. Postgraduate Medicine.
2010;122(1):39–51. “
 }

Doc 1 LAT Feature Score = 0.9526760239113281
Doc 2 LAT Feature Score = 0.0036075458415429787
```

Custom Scorer Project

- Project defines a framework for building custom scorers.
  - Creates base samples to sub-class / implement.
  - Wheel file used to package the scorer
  - Score function is the function to generate a feature score



```
def score(self, document):
    views = document['views']
    accepted = document['accepted']

    if views is not None:
        # if no views, then assuming it is a low rating
        if views < 0:
            return 0
        elif 100 < views <= 2000 and accepted <0:
            return 0.25
        elif 0 < views <= 2000 and accepted > 0:
            return 0.5
        elif 2000 < views <= 5000 and accepted > 0:
            return 0.75
        elif views > 5000 and accepted > 0:
            return 1
```



```
" Validate the document and the query "
self.validate_query(query)
self.validate_document(document)

try:
    " Get the class for the document "
    doc_class = self.doc_to_class(document)

    " Classify the query and return the confidence, if there is a match "
    resp_body = self.classify(query['q'])
    for klass in resp_body['classes']:
        if klass['class_name'] == doc_class:
            return klass['confidence']
    return 0.0
except Exception, e:
    raise se.ScorerRuntimeException(e)
```

Custom Proxy Application

- To use custom features, need to separate calls of retrieve and rank.
  - Proxy can be set up to emulate combined R&R (fcselect) service call.

```
{“scorers”:[
{“init_args”:{
“name”:”DocQueryIntentScorer”,
“short_name”:”qd1",
“description”:”Score based on Document/Query alignment”,
“service_url”: “https://gateway.watsonplatform.net/natural-language-
classifier/api",
“service_username”: “username”,
“service_password”: “password”,
“classifier_id”:”9a8879x44-nlc-969"},
“type”:”query_document”,
“module”:”nlc_intent_scorer”,
“class”:”QuestionDocumentIntentAlignmentScorer”
}]}
```

- Project contains a flask server will handle the integration/injection of custom features that have been registered.
  - Configuration file used to define scorers being.
  - "scorers" field is a list of individual scorer configurations, JSON object that must define:
    - 'init_args' json object, whose fields are the arguments to the constructor for the scorer
    - 'type' field, which should be either 'query', 'document' or 'query_document', depending on the type of scorer. The type is used to identify the package within the 'rr_scorers' project that contains the appropriate scorer
    - 'module' field, which is the name of the python module which contains the scorer
    - 'class' field, which is the name of the scorer class

Thank You