

Documentation

Grammar Website and Search System

Development Setup

Ron Kow

January 2021



This work is licensed under a
[Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-nc-sa/4.0/)

Table of Contents

1	Introduction	1
2	Project Requirements and Virtual Environment	2
2.1	Major Requirements	2
2.2	Anaconda and Miniconda	3
2.2.1	Installation	3
2.2.2	Virtual Environment	5
2.2.3	conda and pip	6
2.2.4	Other conda Commands	6
2.3	Additional Requirements	7
2.4	GitHub Repository	10
2.5	Project Directory Structure	10
3	Django Setup on Development Server	11
3.1	Default Django Website and SQLite Database	11
3.2	Django Admin Site	13
3.3	Templates and Static Files	17
3.4	Django Directory Structure	19
4	Grammar Practice Website	20
4.1	settings.py and urls.py	20
4.2	Source Files	21
5	Quizzes and Database Search	23
5.1	Raw Data	23
5.2	Data Models	24

5.3	Database Tables	24
5.4	Populating the Database	28
5.4.1	Django Admin Site	28
5.4.2	Importing Data	31
5.5	Quizzes and Search	34
5.5.1	Quizzes Page	34
5.5.2	Search Page	36
6	Solr Search	38
6.1	Solr Version and Documentation	38
6.2	Java Runtime Environment	38
6.3	Code	39
6.4	Solr Server	39
6.4.1	Solr Node	39
6.4.2	Creating a Core	41
6.4.3	Deleting a Core	41
6.4.4	Core Directory	42
6.5	Uploading Data to Solr	43
6.5.1	Uploading Methods	43
6.5.2	Uploading Data by Solr's REST API	43
6.6	Deleting Data From Solr	45
6.7	Solr Search	46
6.7.1	Solr Admin UI	46
6.7.2	Website	48
7	Intelligent Solr Search	49
7.1	Learning To Rank Plugin	49
7.2	Feature Store and Model Store	52
7.2.1	Uploading Features	52
7.2.2	Uploading Models	52
7.2.3	Solr Admin UI	53
7.3	Stanford CoreNLP	56
7.4	Intelligent Search	58

7.4.1	Intelligent Search Page	58
7.4.2	Quizzes Page	61
8	Summary of Setup Process	62
9	Creating Model Datasets	64
10	Building and Testing the Models	68
10.1	RankLib	68
10.2	Model Statistics	69
10.3	Converting Models to JSON	69
10.4	Uploading Models and Testing with Queries	70

Chapter 1

Introduction

This document is a tutorial on the setup of the grammar practice website on the Django development web server in Windows 10 environment. The website provides three types of grammar question search:

- *Database search* using Django API
- Solr search with original ranking, by BM25 (which we will call *basic Solr search*)
- Solr search by ranking models (which we will call *intelligent Solr search*)

Intelligent Solr search uses machine-learned ranking models to rerank the original ranking in basic Solr search.

The website is at <https://ronkow.com/grammar> and the source files are at <https://github.com/ronkow/solr-learning-to-rank>. Any information on new features in future will be posted on the website home page.

Setup of Django and Solr in a production environment requires setting up security measures. These are not covered here. You may refer to my documentation on production setup at <https://www.ronkow.com/documentation/>.

I welcome any feedback and correction of errors. You can contact me at ronkow2020@gmail.com.

Chapter 2

Project Requirements and Virtual Environment

2.1 Major Requirements

The following are the software requirements which we will install when we set up the virtual environment for the project. We will install additional libraries after that.

- Python 3.7.* (the latest version of Python that Stanford Stanza 1.1.1 supports)
- Django 2.2.*

Create a directory for the project. Let's name the directory `grammar`. We will download the required libraries and save them in `/grammar/downloads`. First, we need Solr and Stanford CoreNLP:

- Solr 7.7.3 (<https://archive.apache.org/dist/lucene/solr/>)
- Stanford CoreNLP 4.2.0 (<https://stanfordnlp.github.io/CoreNLP/>)

Java Runtime Environment (JRE) is required to run Solr and RankLib. Download and install JRE:

- JRE 1.8 or later (<https://www.java.com/en/download/>)

We will use the Learning To Rank library RankLib to train the ranking models. We will need to download these two libraries:

- RankLib 2.15 (<https://sourceforge.net/projects/lemur/files/lemur/RankLib-2.15/>)
- Apache Commons Mathematics 3.5 (<http://commons.apache.org/proper/commons-math/>)

Lastly, we will use the software **DB Browser for SQLite** to view the database schema and populate the database with data. Download and install it from:

<https://sqlitebrowser.org/dl/>

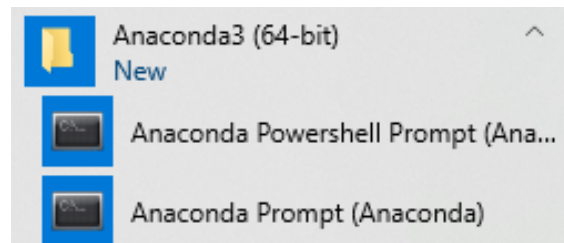
2.2 Anaconda and Miniconda

2.2.1 Installation

To create the project virtual environment, we use the package manager **conda** from the Miniconda repository, which is a small version (i.e., less packages) of the Anaconda repository. Download and install **Miniconda3 Windows 64-bit for Python 3.8** from:

<https://docs.conda.io/en/latest/miniconda.html>

After installation, we will see the two command line applications provided by Anaconda:



We will use Anaconda Powershell Prompt. Open the Anaconda Powershell and we will see the default virtual environment named **base**:

```
(base) PS D:>
```


The following is the list of libraries installed in this default environment:

```
(base) PS D:\> conda list
# packages in environment at C:\Anaconda:
#
# Name                      Version                Build Channel
asgiref                     3.3.1                  pypi_0    pypi
brotlipy                    0.7.0                  py38h2bbff1b_1003
ca-certificates             2020.11.8              h5b45459_0    conda-forge
certifi                     2020.11.8              py38haa244fe_0    conda-forge
cffi                        1.14.3                 py38hcd4344a_2
chardet                     3.0.4                  py38haa95532_1003
conda                       4.9.2                  py38haa244fe_0    conda-forge
conda-package-handling      1.7.2                  py38h76e460a_0
console_shortcut            0.1.1                  4
cryptography                3.2.1                  py38hcd4344a_1
django                      3.1.4                  pypi_0    pypi
git                         2.23.0                 h6bb4b03_0
idna                        2.10                   py_0
kaggle                      1.5.9                  py38h32f6830_0    conda-forge
menuinst                    1.4.16                 py38he774522_1
numpy                       1.19.4                 pypi_0    pypi
openssl                     1.1.1h                  he774522_0    conda-forge
pip                         20.2.4                 py38haa95532_0
powershell_shortcut        0.0.1                  3
protobuf                    3.14.0                 pypi_0    pypi
pycosat                     0.6.3                  py38h2bbff1b_0
pyparser                    2.20                   py_2
pyopenssl                   19.1.0                 pyhd3eb1b0_1
pysocks                     1.7.1                  py38haa95532_0
python                      3.8.5                  h5fd99cc_1
python-dateutil             2.8.1                  py_0    conda-forge
python-slugify              4.0.1                  pyh9f0ad1d_0    conda-forge
python_abi                  3.8                    1_cp38    conda-forge
pytz                        2020.4                 pypi_0    pypi
pywin32                     227                    py38he774522_1
requests                    2.24.0                 py_0
ruamel_yaml                 0.15.87                py38he774522_1
setuptools                  50.3.1                 py38haa95532_1
six                         1.15.0                 py38haa95532_0
sqlite                      3.33.0                 h2a8f88b_0
sqlparse                    0.4.1                  pypi_0    pypi
text-unidecode              1.3                    py_0    conda-forge
torch                      1.7.1                  pypi_0    pypi
tqdm                        4.51.0                 pyhd3eb1b0_0
typing-extensions           3.7.4.3                pypi_0    pypi
unidecode                   1.1.1                  py_0    conda-forge
urllib3                     1.24.3                 py_1    conda-forge
vc                          14.1                   h0510ff6_4
vs2015_runtime              14.16.27012            hf0eaf9b_3
wheel                       0.35.1                 pyhd3eb1b0_0
win_inet_pton               1.1.0                  py38haa95532_0
wincertstore                0.2                    py38_0
yaml                        0.2.5                  he774522_0
zlib                        1.2.11                 h62dcd97_4
```

2.2.2 Virtual Environment

Create a YAML file with the following definitions and name it `environment_grammar.yml`. In the YAML file, we specify the name of the virtual environment (`grammar`) and the list of libraries to be installed during the creation of the virtual environment:

```
name: grammar
dependencies:
- python==3.7.*
- Django==2.2.*
```

Create the virtual environment:

```
(base) PS D:\> conda env create -f environment_grammar.yml
Collecting package metadata (repodata.json): done
Solving environment: done
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
```

Activate the `grammar` environment and display the list of libraries installed:

```
(base) PS D:\> conda activate grammar
(grammar) PS D:\> conda list
# packages in environment at C:\Anaconda\envs\grammar:
#
# Name                                Version                                Build      Channel
ca-certificates                      2020.12.8                              haa95532_0
certifi                              2020.12.5                              py37haa95532_0
django                               2.2.5                                  py37_1
openssl                              1.1.1i                                h2bbff1b_0
pip                                  20.3.3                                py37haa95532_0
python                               3.7.9                                  h60c2a47_0
pytz                                  2020.5                                pyhd3eb1b0_0
setuptools                           51.1.2                                py37haa95532_3
sqlite                                3.33.0                                h2a8f88b_0
sqlparse                             0.4.1                                  py_0
vc                                    14.2                                  h21ff451_1
vs2015_runtime                       14.27.29016                           h5e58377_2
wheel                                0.36.2                                pyhd3eb1b0_0
wincertstore                         0.2                                    py37_0
zlib                                  1.2.11                                h62dcd97_4
```

In my system, the directory for the virtual environment is in the following location:

```
C:\Anaconda\envs\grammar
```

Note that my installation of Anaconda is in C drive while my project files are in D drive.

2.2.3 conda and pip

We use `conda` to install additional packages or libraries from the Anaconda repository:

```
(environment_name) PS D:\> conda install package_name
```

For Python 3.7 and 64-bit Windows, the complete list of available packages in Anaconda is shown in:

https://docs.anaconda.com/anaconda/packages/py3.7_win-64/

We may need a package not found in Anaconda. Anaconda documentation (<https://docs.conda.io/projects/conda/en/master/user-guide/tasks/manage-pkgs.html>) recommends trying to find such packages in **conda-forge** (<https://conda-forge.org/feedstock-outputs/>) first and use `conda` if found:

```
(environment_name) PS D:\> conda install -c conda-forge package_name
```

For Python packages not found in Anaconda, we can use `pip` instead. `conda` (package manager for packages of any language) is designed to be as compatible with `pip` (package manager for Python packages only) as possible. Always try using `conda` to install any Python package first. If that does not work (i.e., the package is not found in Anaconda), use `pip`:

```
(environment_name) PS D:\> pip install package_name
```

2.2.4 Other conda Commands

To create a new virtual environment with no libraries:

```
(base) PS D:\> conda create --name environment_name
```

To delete a virtual environment:

```
(base) PS D:\> conda remove --name environment_name --all
```

To uninstall a package in a virtual environment:

```
(environment_name) PS D:\> conda uninstall package_name
```

To exit from a virtual environment:

```
(environment_name) PS D:\> conda deactivate
```

2.3 Additional Requirements

We will now install the additional requirements. Install the following libraries using `conda`:

```
(grammar) PS D:\> conda install -c stanfordnlp stanza==1.1.*
(grammar) PS D:\> conda install nltk==3.5.*
(grammar) PS D:\> conda install pandas==1.2.*
(grammar) PS D:\> conda install jupyterlab
```

Install the following libraries using `pip`:

```
(grammar) PS D:\> pip install django-crispy-forms==1.10.*
(grammar) PS D:\> pip install pycorenlp==0.3.*
```

Jupyter Lab is required to run our code. **NLTK** is used to generate POS tags. **Stanford Stanza** is required as a client for **CoreNLP** server, which we use to generate grammar production rules. **pycorenlp** provides the Python API for **CoreNLP**. **django-crispy-forms** is used for front-end web page design. **pandas** is required to run the code to create the model datasets.

We may also use **Haystack** API to upload data to Solr from the database. However, as I will explain in Section 6.5, this is optional as we can also use the Solr API to upload data. If you wish to experiment with **Haystack**, install the following:

```
(grammar) PS D:\> pip install pysolr==3.9.*
(grammar) PS D:\> pip install django-haystack==2.8.*
```

The following is the complete list of libraries installed in the virtual environment (excluding **pysolr** and **Haystack**):

```
(grammar) PS D:\> conda list
# packages in environment at C:\Anaconda\envs\grammar:
#
_pytorch_select      1.1.0                  cpu
argon2-cffi          20.1.0                py37he774522_1
async_generator      1.10                  py37h28b3542_0
attrs                20.3.0                pyhd3eb1b0_0
backcall             0.2.0                  py_0
blas                 1.0                    mkl
bleach               3.2.3                  pyhd3eb1b0_0
brotlipy             0.7.0                  py37h2bbff1b_1003
ca-certificates      2021.1.19              haa95532_0
certifi              2020.12.5              py37haa95532_0
cffi                 1.14.4                 py37hcd4344a_0
chardet              4.0.0                  py37haa95532_1003
click                7.1.2                  pyhd3eb1b0_0
colorama             0.4.4                  pyhd3eb1b0_0
cryptography         3.3.1                  py37hcd4344a_0
decorator            4.4.2                  py_0
```

defusedxml	0.6.0	py_0	
django	2.2.5	py37_1	
django-crispy-forms	1.10.0	pypi_0	pypi
entrypoints	0.3	py37_0	
idna	2.10	pyhd3eb1b0_0	
importlib-metadata	2.0.0	py_1	
importlib_metadata	2.0.0	1	
intel-openmp	2020.2	254	
ipykernel	5.3.4	py37h5ca1d4c_0	
ipython	7.19.0	py37hd4e2768_0	
ipython_genutils	0.2.0	pyhd3eb1b0_1	
jedi	0.18.0	py37haa95532_1	
jinja2	2.11.2	pyhd3eb1b0_0	
joblib	1.0.0	pyhd3eb1b0_0	
json5	0.9.5	py_0	
jsonschema	3.2.0	py_2	
jupyter_client	6.1.7	py_0	
jupyter_core	4.7.0	py37haa95532_0	
jupyterlab	2.2.6	py_0	
jupyterlab_pygments	0.1.2	py_0	
jupyterlab_server	1.2.0	py_0	
libprotobuf	3.13.0.1	h200bbdf_0	
libsodium	1.0.18	h62dcd97_0	
m2w64-gcc-libgfortran	5.3.0	6	
m2w64-gcc-libs	5.3.0	7	
m2w64-gcc-libs-core	5.3.0	7	
m2w64-gmp	6.1.0	2	
m2w64-libwinpthread-git	5.0.0.4634.697f757	2	
markupsafe	1.1.1	py37hfa6e2cd_1	
mistune	0.8.4	py37hfa6e2cd_1001	
mkl	2020.2	256	
mkl-service	2.3.0	py37h196d8e1_0	
mkl_fft	1.2.0	py37h45dec08_0	
mkl_random	1.1.1	py37h47e9c7a_0	
msys2-conda-epoch	20160418	1	
nbclient	0.5.1	py_0	
nbconvert	6.0.7	py37_0	
nbformat	5.1.2	pyhd3eb1b0_1	
nest-asyncio	1.4.3	pyhd3eb1b0_0	
ninja	1.10.2	py37h6d14046_0	
nlTK	3.5	py_0	
notebook	6.2.0	py37haa95532_0	
numpy	1.19.2	py37hadc3359_0	
numpy-base	1.19.2	py37ha3acd2a_0	
openssl	1.1.1i	h2bbff1b_0	
packaging	20.8	pyhd3eb1b0_0	
pandas	1.2.1	py37hf11a4ad_0	
pandoc	2.11	h9490d1a_0	
pandocfilters	1.4.3	py37haa95532_1	
parso	0.8.1	pyhd3eb1b0_0	
pickleshare	0.7.5	pyhd3eb1b0_1003	

pip	20.3.3	py37haa95532_0	
prometheus_client	0.9.0	pyhd3eb1b0_0	
prompt-toolkit	3.0.8	py_0	
protobuf	3.13.0.1	py37ha925a31_1	
pycorenlp	0.3.0	pypi_0	pypi
pycparser	2.20	py_2	
pygments	2.7.4	pyhd3eb1b0_0	
pyopenssl	20.0.1	pyhd3eb1b0_1	
pyparsing	2.4.7	pyhd3eb1b0_0	
pyrsistent	0.17.3	py37he774522_0	
pysocks	1.7.1	py37_1	
python	3.7.9	h60c2a47_0	
python-dateutil	2.8.1	py_0	
pytorch	1.3.1	cpu_py37h9f948e0_0	
pytz	2020.5	pyhd3eb1b0_0	
pywin32	227	py37he774522_1	
pywinpty	0.5.7	py37_0	
pyzmq	20.0.0	py37hd77b12b_1	
regex	2020.11.13	py37h2bbff1b_0	
requests	2.25.1	pyhd3eb1b0_0	
send2trash	1.5.0	pyhd3eb1b0_1	
setuptools	52.0.0	py37haa95532_0	
six	1.15.0	py37haa95532_0	
sqlite	3.33.0	h2a8f88b_0	
sqlparse	0.4.1	py_0	
stanza	1.1.1	py37_0	stanfordnlp
terminado	0.9.2	py37haa95532_0	
testpath	0.4.4	py_0	
tornado	6.1	py37h2bbff1b_0	
tqdm	4.55.1	pyhd3eb1b0_0	
traitlets	5.0.5	py_0	
urllib3	1.26.3	pyhd3eb1b0_0	
vc	14.2	h21ff451_1	
vs2015_runtime	14.27.29016	h5e58377_2	
wcwidth	0.2.5	py_0	
webencodings	0.5.1	py37_1	
wheel	0.36.2	pyhd3eb1b0_0	
win_inet_pton	1.1.0	py37haa95532_0	
wincertstore	0.2	py37_0	
winpty	0.4.3	4	
zeromq	4.3.3	ha925a31_3	
zipp	3.4.0	pyhd3eb1b0_0	
zlib	1.2.11	h62dcd97_4	

2.4 GitHub Repository

To use the most updated code, download the project repository from GitHub:

```
(grammar) PS D:\grammar> git clone https://github.com/ronkow/solr-learning-to-rank
```

2.5 Project Directory Structure

When we complete the entire setup, the project directory will have the following structure:

```
grammar/  
  corenlp4  
  data  
  django  
  downloads  
  model  
  ranklib2  
  solr7  
  solr-learning-to-rank
```

Within the directory **data**, there will be three sub-directories **raw**, **feature**, and **model** to store the different types of datasets. In subsequent chapters, we will call the directory **solr-learning-to-rank** the *repository directory*.

Chapter 3

Django Setup on Development Server

We will now set up the default Django web site (i.e., the official page from Django). For the official tutorial, visit:

<https://www.djangoproject.com/start/>

3.1 Default Django Website and SQLite Database

Create the directory `django`. In this directory, we create a Django project. Every Django project has one configuration directory containing the configuration files for the project. Let's name this directory `config`:

```
(grammar) PS D:\grammar\django> django-admin startproject config .
```

Note the dot at the end of the command above. This creates the following Django directory structure with five Python scripts:

```
django/  
  config/  
    __init__.py  
    settings.py  
    urls.py  
    wsgi.py  
  manage.py
```

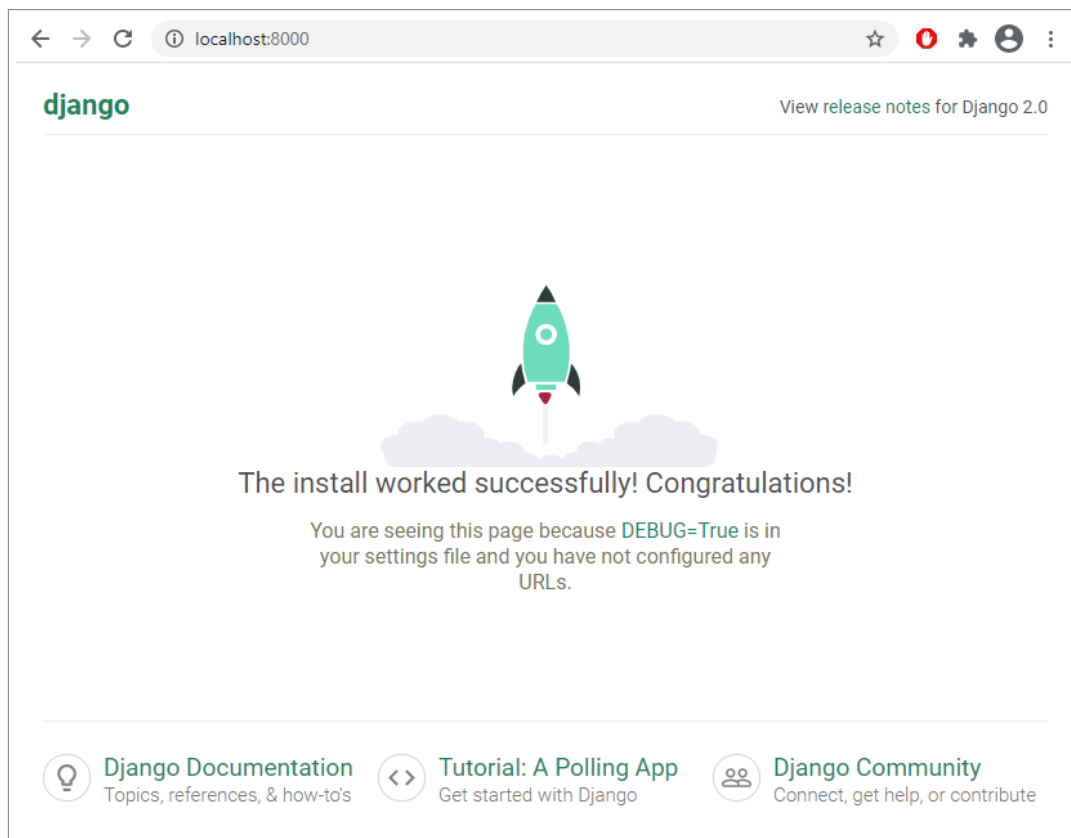

Django includes a basic web server for development. Run this development server:

```
(grammar) PS D:\grammar\django> python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).

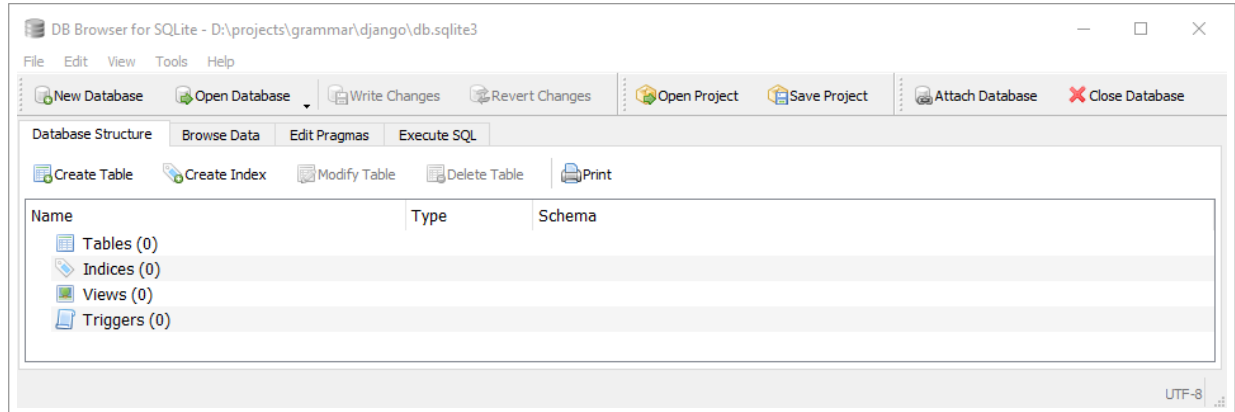
You have 17 unapplied migration(s).
Your project may not work properly until you apply the migrations for app(s):
admin, auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
January 28, 2021 - 05:27:38
Django version 2.2.5, using settings 'config.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

When the server is running, we will see the following official page from Django at `http://127.0.0.1:8000/`, or equivalently, `http://localhost:8000/`:



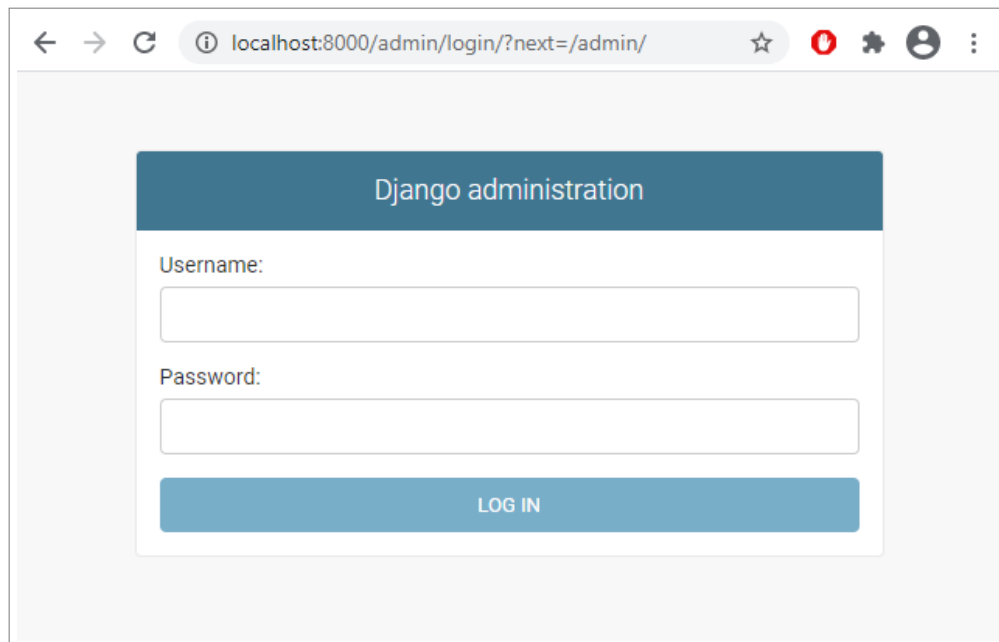
To quit the server, press **Ctrl-C**. We need to quit the server before we can run any command from `manage.py`.

Running the command `runserver` the first time will also create the SQLite database named `db.sqlite3` in `/grammar/django`. Using **DB Browser** (download and install it from <https://sqlitebrowser.org/dl/>), we can open the database, which currently has no tables:



3.2 Django Admin Site

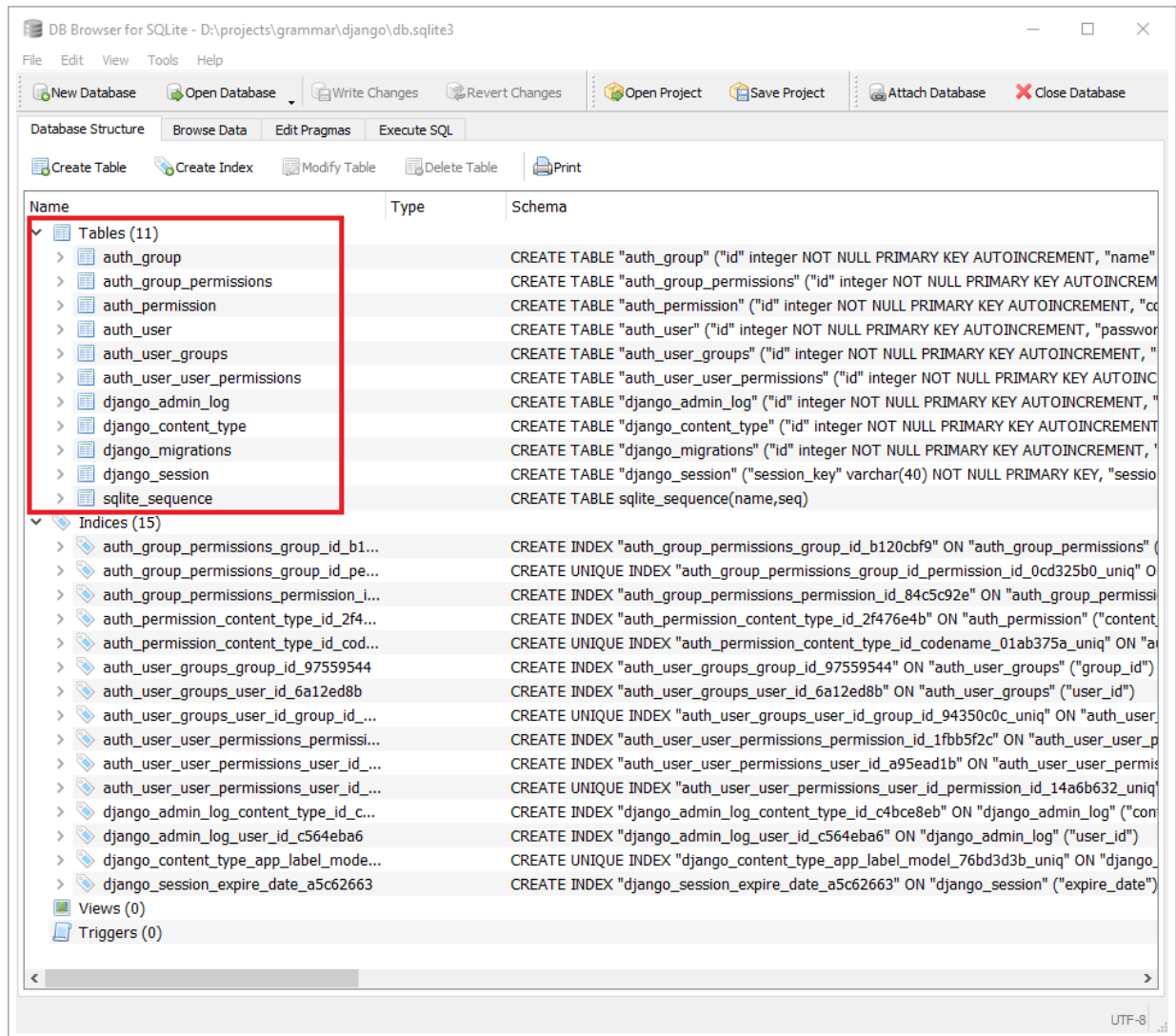
The next step is to set up the Django admin site. Go to `http://localhost:8000/admin/` and we will see the admin site login page:



However, we will not be able to log in to the admin site until we create the required database tables for user data and create an admin user account. Run the command `migrate`:

```
(grammar) PS D:\grammar\django> python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying sessions.0001_initial... OK
```

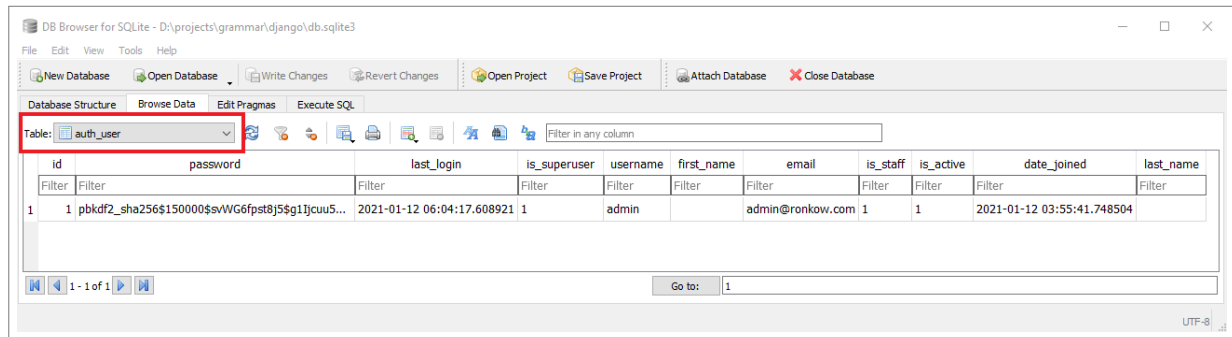
This will create the following tables:



Next, create an admin user account:

```
(grammar) PS D:\grammar\django> python manage.py createsuperuser
Username (leave blank to use 'myusername'): admin
Email address: admin@mydomain.com
Password:
Password (again):
Superuser created successfully.
```

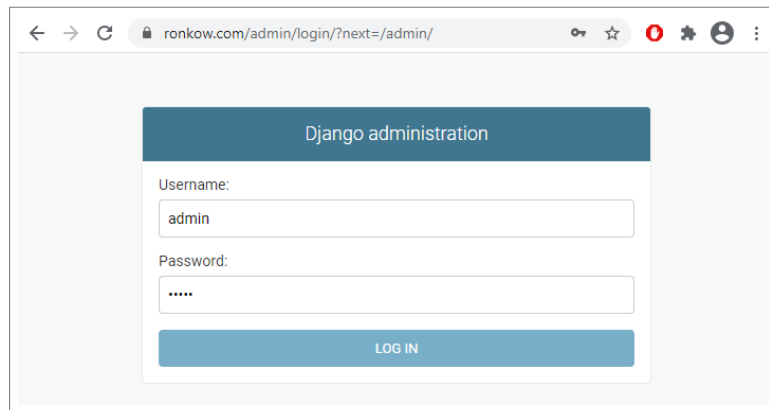
User data are stored in the table `auth_user`:



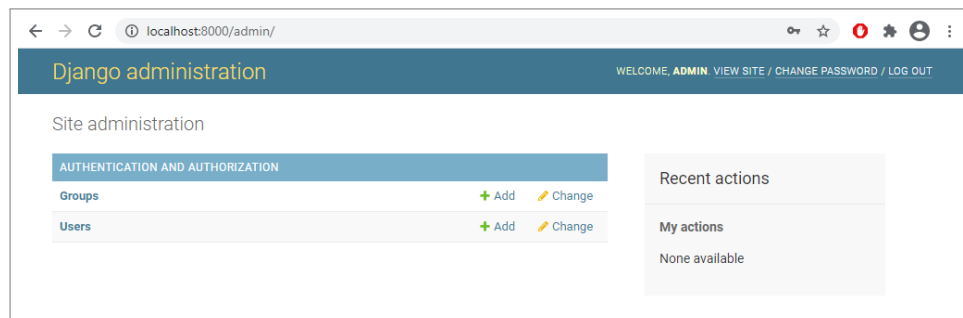
The screenshot shows the DB Browser for SQLite interface. The 'Table: auth_user' is selected in the 'Database Structure' tab. The table has the following columns: id, password, last_login, is_superuser, username, first_name, email, is_staff, is_active, date_joined, and last_name. The data shows one record for the 'admin' user.

id	password	last_login	is_superuser	username	first_name	email	is_staff	is_active	date_joined	last_name
1	pbkdf2_sha256\$150000\$svWG6fpst8j5\$g1jcuu5...	2021-01-12 06:04:17.608921	1	admin		admin@ronkow.com	1	1	2021-01-12 03:55:41.748504	

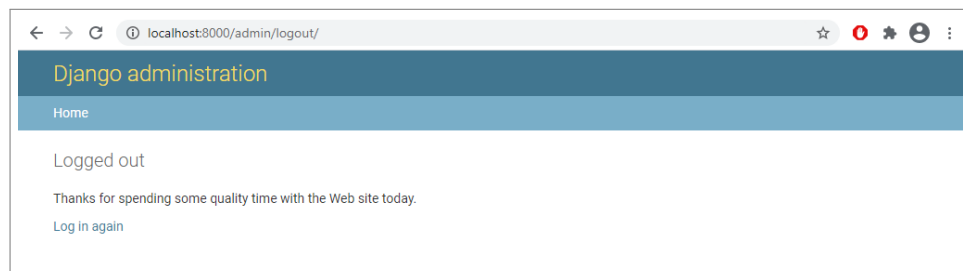
Log in to the admin site at `http://localhost:8000/admin/` using the admin user account created:



The screenshot shows the Django administration login page. The URL is `ronkow.com/admin/login/?next=/admin/`. The page has a header 'Django administration' and a login form with fields for 'Username:' (containing 'admin') and 'Password:' (containing '.....'). There is a 'LOG IN' button.



The screenshot shows the Django administration site home page. The URL is `localhost:8000/admin/`. The page has a header 'Django administration' and a sub-header 'WELCOME, ADMIN. VIEW SITE / CHANGE PASSWORD / LOG OUT'. The main content area is titled 'Site administration' and contains a section 'AUTHENTICATION AND AUTHORIZATION' with links for 'Groups' and 'Users' (each with 'Add' and 'Change' links). There is also a 'Recent actions' section with a 'My actions' link and the text 'None available'.



The screenshot shows the Django administration site logout page. The URL is `localhost:8000/admin/logout/`. The page has a header 'Django administration' and a sub-header 'Home'. The main content area is titled 'Logged out' and contains the text 'Thanks for spending some quality time with the Web site today.' and a 'Log in again' link.

3.3 Templates and Static Files

We will now create a “hello world” page and a CSS file. Create the directories as follows:

```
(grammar) PS D:\grammar\django> mkdir static_files/css
(grammar) PS D:\grammar\django> mkdir templates
```

Create a CSS file with the following code and save the file as `style.css` in `/grammar/django/static_files/css`:

```
p {
    color: blue;
}
```

Create the a HTML file for a “hello world” page and save it as `helloworld.html` in `/grammar/django/templates`:

```
{% load static %}
<!doctype html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Hello World</title>
    <link rel="stylesheet" href="{% static 'css/style.css' %}">
  </head>
  <body>
    <p>Hello world!</p>
  </body>
</html>
```

In `helloworld.html`, we use the Django Template Language with HTML:

<https://docs.djangoproject.com/en/3.1/topics/templates/#the-django-template-language>

Next, in `settings.py`, add the path of the directory `templates` to the list `TEMPLATES`. At the last line of `settings.py`, add the path of the directory `static_files`:

```
TEMPLATES = [
    {
        'DIRS': [os.path.join(BASE_DIR, 'templates')],
    },
]

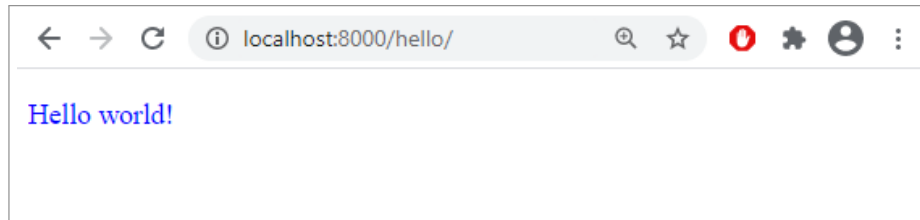
STATICFILES_DIRS = [os.path.join(BASE_DIR, 'static_files')]
```

In `urls.py`, add the URL path for the “hello world” page:

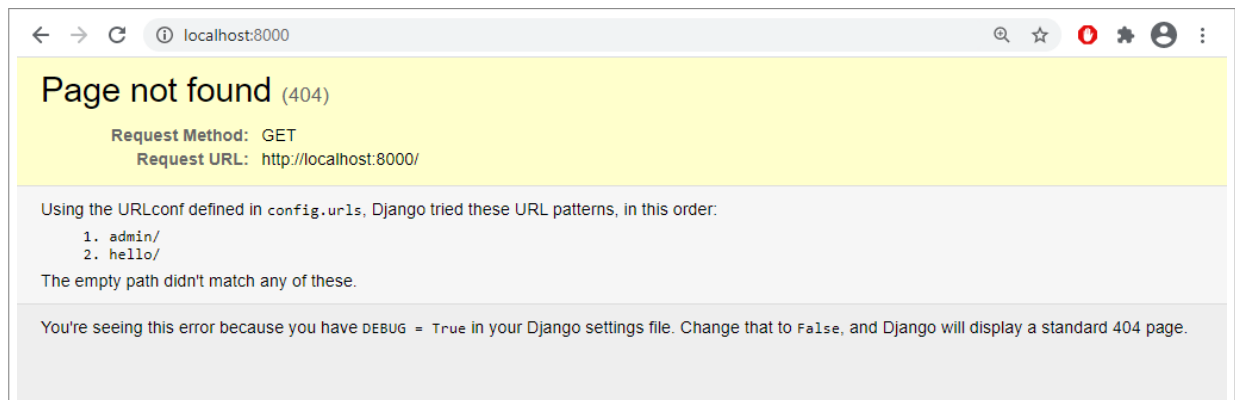
```
from django.views.generic.base import TemplateView

urlpatterns = [
    path('hello/', TemplateView.as_view(template_name = 'helloworld.html')),
]
```

Check that the “hello world” page loads at `http://localhost:8000/hello/` and that the text color is blue:



The default Django page at `http://localhost:8000/` will no longer load. Instead, we will see an error page:



If there are errors while loading pages, this error page will be displayed if the variable `DEBUG` is set to `True` in `settings.py`. The error messages will also be displayed in Anaconda Powershell.

3.4 Django Directory Structure

At this point, we have the following Django directory structure:

```
django/  
  config/  
    __pycache__/  
      __init__.cpython-37.pyc  
      settings.cpython-37.pyc  
      urls.cpython-37.pyc  
      wsgi.cpython-37.pyc  
    __init__.py  
    settings.py  
    urls.py  
    wsgi.py  
  static_files/  
    css/  
      style.css  
  templates/  
    helloworld.html  
  db.sqlite3  
  manage.py
```


Chapter 4

Grammar Practice Website

We will build the grammar practice website step-by-step, by adding files or directories from the repository directory to the project file system, or by adding code to existing files.

The setup of user and authentication features (i.e., user sign up and sign in, password features, etc.) in <http://ronkow.com/grammar/> are omitted here. I will only describe the setup of the search system and any features linked to it, such as quizzes.

4.1 settings.py and urls.py

We will organize the website features (i.e., quizzes, search, and other pages) into three Django applications: `apppage`, `appquiz`, and `appsearch`. These must be declared in the list `INSTALLED_APPS` in `settings.py`. We also declare the application `crispy_forms` and the variable `CRISPY_TEMPLATE_PACK`:

```
INSTALLED_APPS = [  
    # local apps  
    'apppage.apps.ApppageConfig',  
    'appquiz.apps.AppquizConfig',  
    'appsearch.apps.AppsearchConfig',  
  
    # Third-party apps  
    'crispy_forms',  
]  
  
CRISPY_TEMPLATE_PACK = 'bootstrap4'
```

Next, we define the URL paths in `urls.py`:

```
from django.urls import include
urlpatterns = [
    path('', include('apppage.urls')),
    path('quiz/', include('appquiz.urls')),
    path('search/', include('appsearch.urls')),
]
```

4.2 Source Files

Copy `base.html` (which define the common navigation header on every page) and `base.css` from the repository directory:

- `/grammar/django/templates/base.html`
- `/grammar/django/static_files/css/base.css`

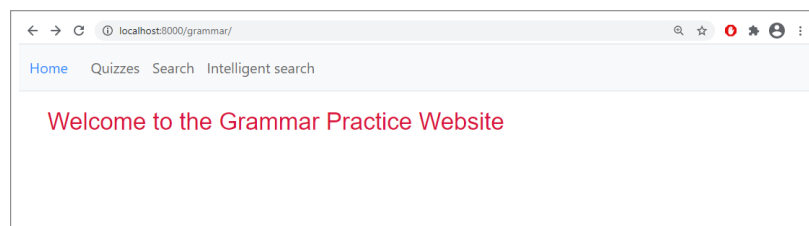
Copy these application and template directories:

- `/grammar/django/apppage`
- `/grammar/django/appquiz`
- `/grammar/django/appsearch`
- `/grammar/django/templates/page`
- `/grammar/django/templates/quiz`
- `/grammar/django/templates/search`

In `/grammar/django/apppage/urls.py`, note that we have defined the URL path of the home page:

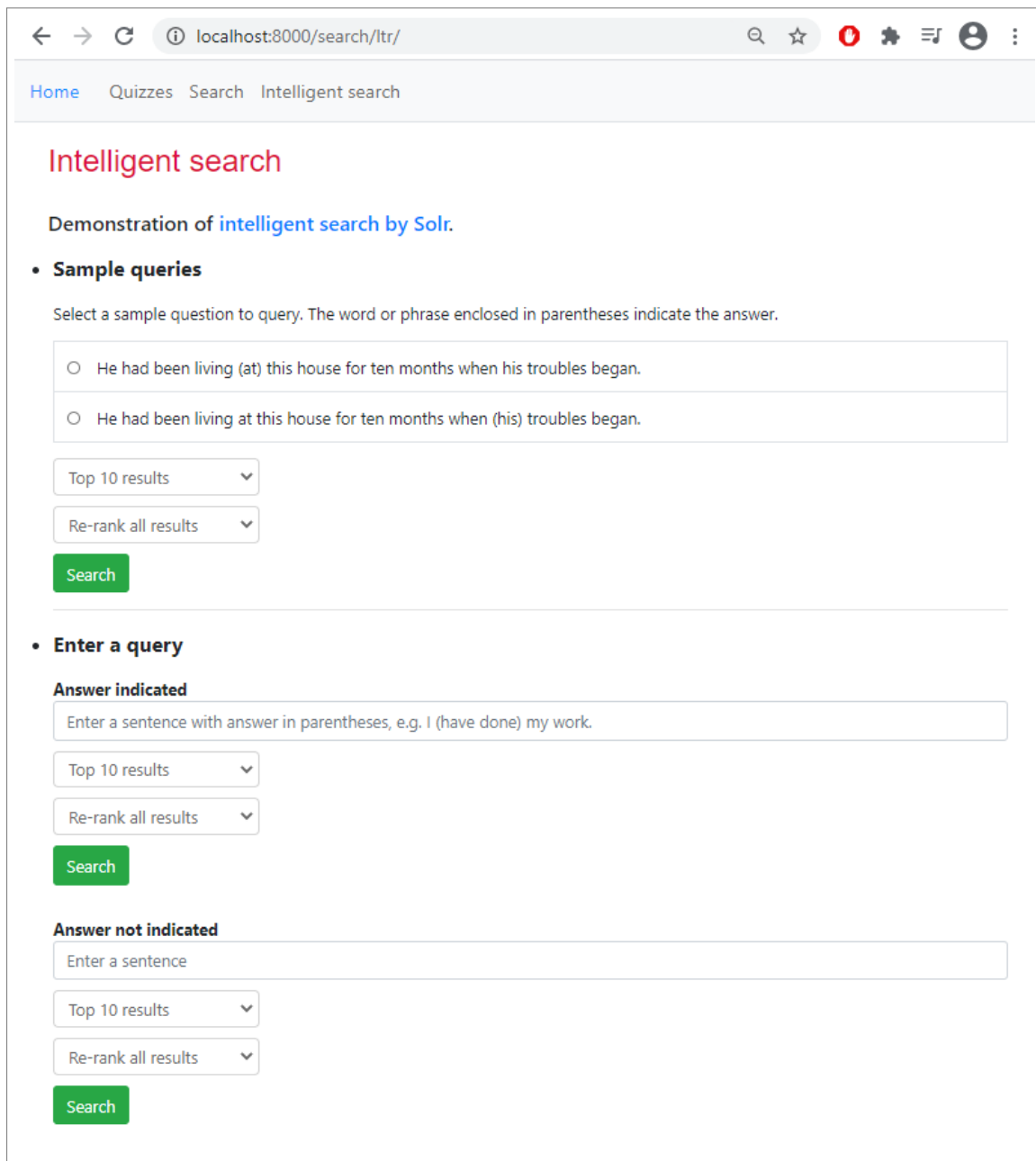
```
urlpatterns = [
    path('grammar/',
        views.HomeView.as_view(),
        name = 'homeview'),
```

Connect to the internet and run the development server. Check that the home page loads at `http://localhost:8000/grammar/`:



The template `base.html` references the **Bootstrap** stylesheet and JavaScript code at the Bootstrap repository, and also the JavaScript libraries **JQuery** and **Popper** at their respective websites. Therefore we must be connected to the internet in order for the development server to run.

If we click on the links for **Quizzes** and **Search**, we will see an error page because we have not yet created the database tables which these two pages depend on. However the **Intelligent Search** page will load, since it does not depend on the database:



The screenshot shows a web browser window with the address bar displaying `localhost:8000/search/ltr/`. The browser's navigation bar includes links for [Home](#), [Quizzes](#), [Search](#), and [Intelligent search](#). The main content area is titled "Intelligent search" in red. Below the title, it says "Demonstration of intelligent search by Solr." and lists a section "Sample queries". This section contains two radio button options: "He had been living (at) this house for ten months when his troubles began." and "He had been living at this house for ten months when (his) troubles began." Below these options are two dropdown menus labeled "Top 10 results" and "Re-rank all results", followed by a green "Search" button. The next section is "Enter a query", which is divided into two parts: "Answer indicated" and "Answer not indicated". Each part has a text input field with a placeholder sentence, the same two dropdown menus, and a green "Search" button.

Chapter 5

Quizzes and Database Search

5.1 Raw Data

Copy the raw data directory from the repository directory:

- `/grammar/data/raw`

This directory contains the following four datasets:

- `rawdata_quiz.csv` (46 *quiz questions*, for the database table `appquiz_modelquestion`)
- `rawdata_doc.csv` (850 *question bank questions*, for the database table `appsearch_modelquestion`, also used as documents for training of ranking models)
- `rawdata_query.csv` (152 *question bank questions*, for the database table `appsearch_modelquestion`, also used as queries for training of ranking models)
- `rawdata_query_validate_test.csv` (152 questions used as queries for validation and testing of ranking models)

We will import the first three datasets to the database. Thus we will have 46 quiz questions and a total of 1002 question bank questions. First, we need to create the database tables, as described in the next section.

5.2 Data Models

Django automatically creates database tables according to *data models* we define in the script `models.py` for a particular application.

In `/grammar/django/appquiz/models.py`, we have defined the following data model. Django will create the database tables `appquiz_modeltopic`, `appquiz_modelquiz`, and `appquiz_modelquestion` according to this data model definition:

```
class ModelTopic(models.Model):
    topic_name = models.CharField(max_length=100, unique=True)
    topic_examples = models.TextField(null=True)
    topic_slug = models.SlugField()

class ModelQuiz(models.Model):
    QUIZ_NUMBER = ((1, 'Quiz 1'), (2, 'Quiz 2'),)
    quiz_topic = models.ForeignKey(ModelTopic, on_delete=models.CASCADE, related_name='modelquiztopic')
    quiz_number = models.IntegerField(choices=QUIZ_NUMBER)

class ModelQuestion(models.Model):
    q_topic = models.ForeignKey(ModelTopic, on_delete=models.CASCADE, related_name='modelquestiontopic')
    q_quiz = models.ForeignKey(ModelQuiz, on_delete=models.CASCADE, related_name='modelquestionquiz')
    q_question = models.TextField(unique=True)
    q_answer = models.CharField(max_length=50)
    q_choice1 = models.CharField(max_length=50)
    q_choice2 = models.CharField(max_length=50)
    q_choice3 = models.CharField(max_length=50)
```

In `/grammar/django/appsearch/models.py`, we have defined the following data model. Django will create the database table `appsearch_modelquestionbank` accordingly:

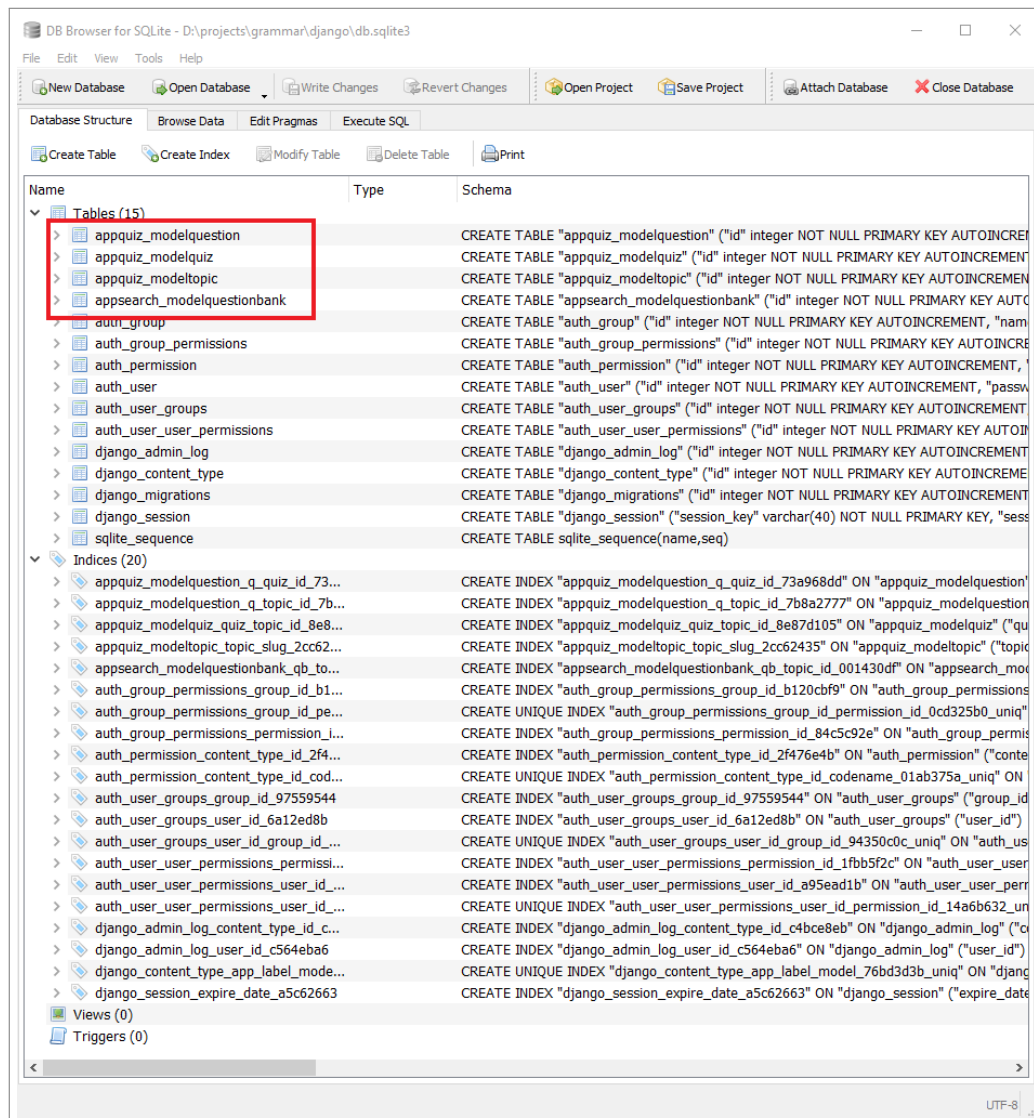
```
class ModelQuestionbank(models.Model):
    qb_topic = models.ForeignKey('appquiz.ModelTopic',
                                on_delete=models.CASCADE, related_name='modelqbtopic')
    qb_question = models.TextField(unique=True)
    qb_answer = models.CharField(max_length=50)
    qb_choice1 = models.CharField(max_length=50)
    qb_choice2 = models.CharField(max_length=50)
    qb_choice3 = models.CharField(max_length=50)
```

5.3 Database Tables

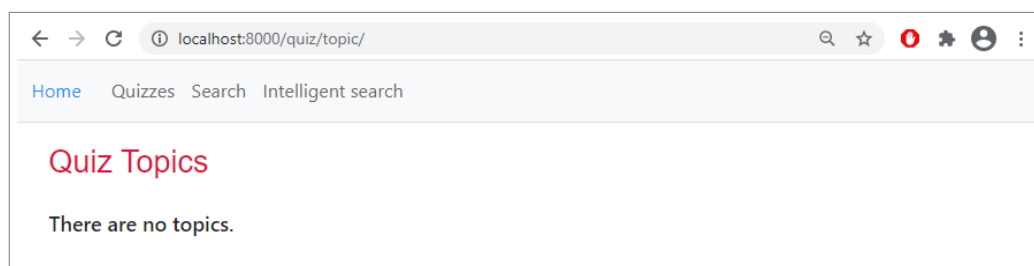
We now tell Django to create the database tables which the **Quizzes** and **Search** pages depend on. Run the command `migrate`:

```
(grammar) PS D:\grammar\django> python manage.py migrate
Operations to perform:
  Apply all migrations: admin, appquiz, appsearch, auth, contenttypes, sessions
Running migrations:
  Applying appquiz.0001_initial... OK
  Applying appsearch.0001_initial... OK
```

Four tables will be created in the database:



As there are no data in the database, the **Quizzes** page will now load showing no information:



The **Search** page will also load:

The screenshot shows a web browser window with the address bar displaying 'localhost:8000/search/'. The browser's navigation bar includes back, forward, and refresh buttons, along with search, star, and user profile icons. Below the address bar is a navigation menu with links for 'Home', 'Quizzes', 'Search', and 'Intelligent search'. The main content area is titled 'Search' in a large red font. Underneath, it says 'Search for questions from Solr.' followed by a section header '• Solr search'. This section contains a text input field with the placeholder 'Enter your query text', a dropdown menu set to 'Top 10 results', and a green 'Search' button. A horizontal line separates this from the next section, which is titled 'Search for questions from the database.' followed by a section header '• Search by topic'. This section has a dropdown menu labeled 'Select a topic' and a green 'Search' button. Below that is a section header '• Search by question text' with a text input field and a green 'Search' button. The final section is '• Search by answer choice text', also featuring a text input field and a green 'Search' button.

← → ↻ ⓘ localhost:8000/search/ 🔍 ☆ 🔴 ⚙️ 👤 ⋮

[Home](#) [Quizzes](#) [Search](#) [Intelligent search](#)

Search

Search for questions from Solr.

- **Solr search**

Enter your query text

Top 10 results ▼

Search

Search for questions from the database.

- **Search by topic**

Select a topic ▼

Search

- **Search by question text**

Enter your query text

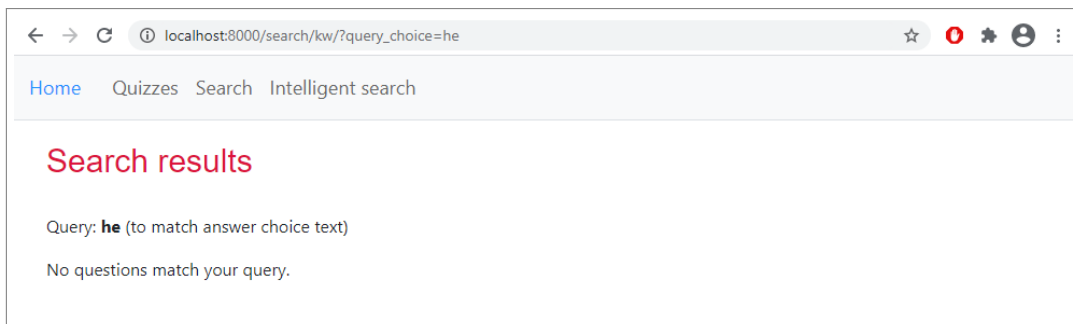
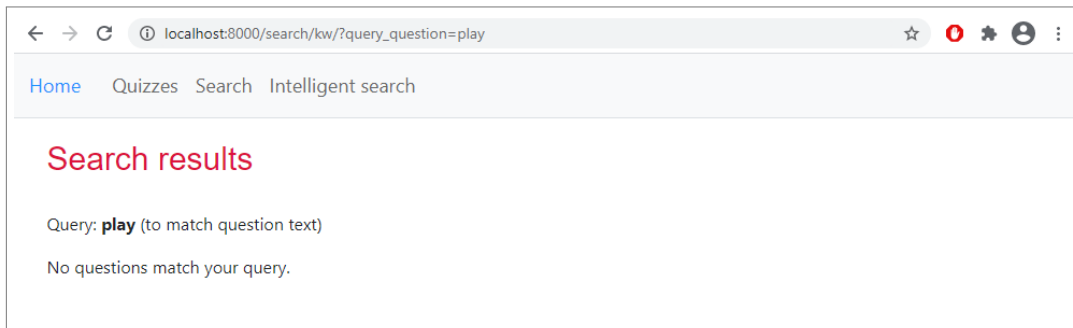
Search

- **Search by answer choice text**

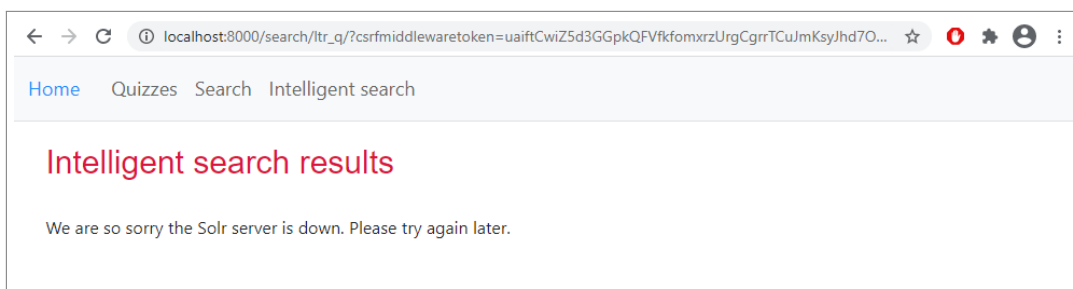
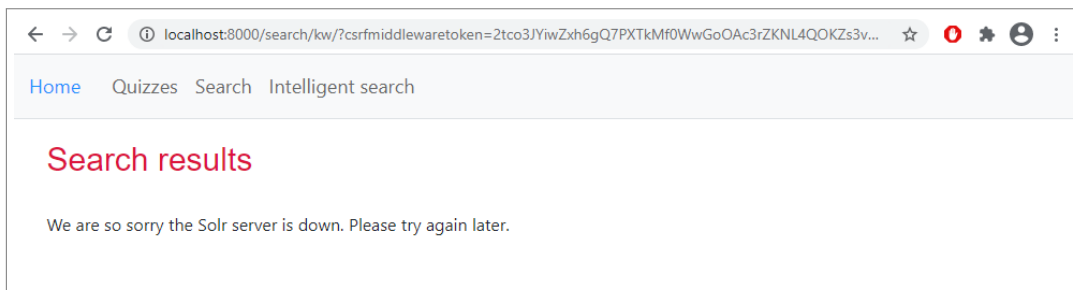
Enter your query text

Search

If we try to do a database search, there will be no results, as the database has no data:



If we try to do a Solr search on the **Search** page and **Intelligent search** page, there will also be no results as we have not yet set up the Solr server:



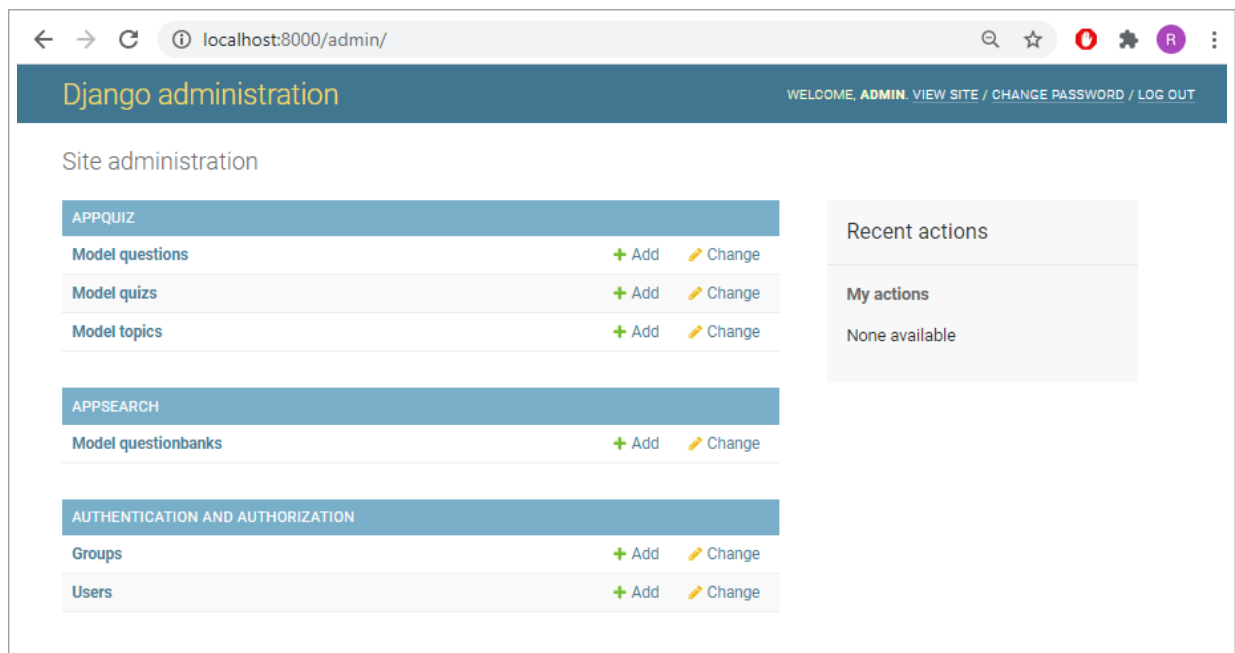
5.4 Populating the Database

There are two ways to populate the database. We can either create new data row by row from the Django admin site, or we can import data to the database from a CSV file.

We will use the Django admin site to create the topics in the table `appquiz_modeltopic` and the quiz numbers in the table `appquiz_modelquiz`. Then we will import the remaining data from CSV files to the other two tables `appquiz_modelquestion` and `appsearch_modelquestionbank`.

5.4.1 Django Admin Site

In the admin site, we will see the four database tables:



Enter data for the five topics in the table `appquiz_modeltopic`:

The screenshot shows the Django administration interface for adding a new model topic. The browser address bar shows `localhost:8000/admin/appquiz/modeltopic/add/`. The page title is "Django administration" and the user is logged in as "ADMIN". The breadcrumb trail is "Home > Appquiz > Model topics > Add model topic".

The form is titled "Add model topic" and contains the following fields:

- Topic name:** A text input field containing "Prepositions".
- Topic examples:** A text area containing "at, in, about".
- Topic slug:** A text input field containing "prepositions".

At the bottom of the form, there are three buttons: "Save and add another", "Save and continue editing", and "SAVE".

The screenshot shows the Django administration interface for the "Model topics" app. The browser address bar shows `localhost:8000/admin/appquiz/modeltopic/`. The page title is "Django administration" and the user is logged in as "ADMIN". The breadcrumb trail is "Home > Appquiz > Model topics".

A green message bar at the top indicates: "The model topic 'Pronouns' was added successfully."

The main content area is titled "Select model topic to change" and includes an "ADD MODEL TOPIC +" button. Below this, there is a table with 5 rows and 4 columns: ID, TOPIC NAME, TOPIC EXAMPLES, and TOPIC SLUG. The table is sorted by ID in descending order.

	ID	TOPIC NAME	TOPIC EXAMPLES	TOPIC SLUG
<input type="checkbox"/>	5	Pronouns	he, her, yours	pronouns
<input type="checkbox"/>	4	Verb tenses	ate, has eaten, is eating	verb-tenses
<input type="checkbox"/>	3	Phrasal verbs	break up, look after	phrasal-verbs
<input type="checkbox"/>	2	Conjunctions	because, after	conjunctions
<input type="checkbox"/>	1	Prepositions	at, in, about	prepositions

Below the table, it says "5 model topics".

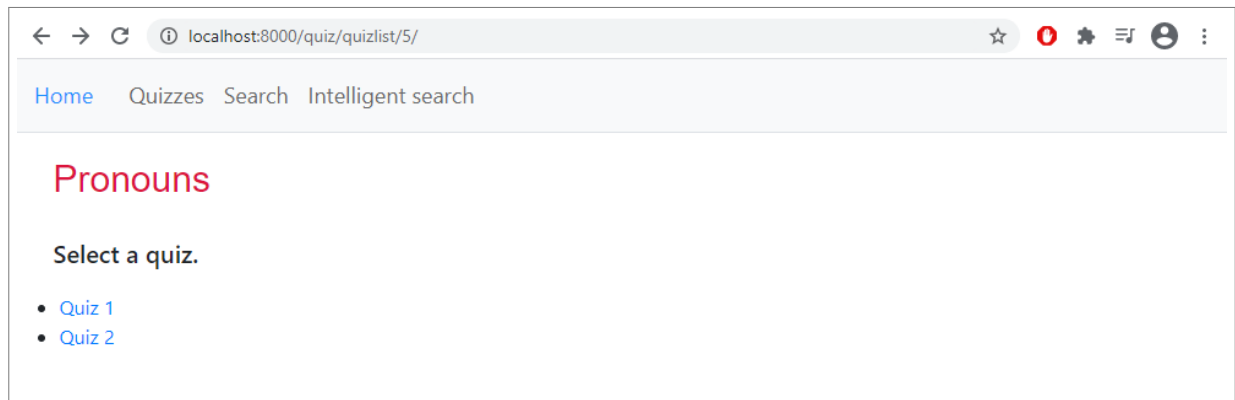
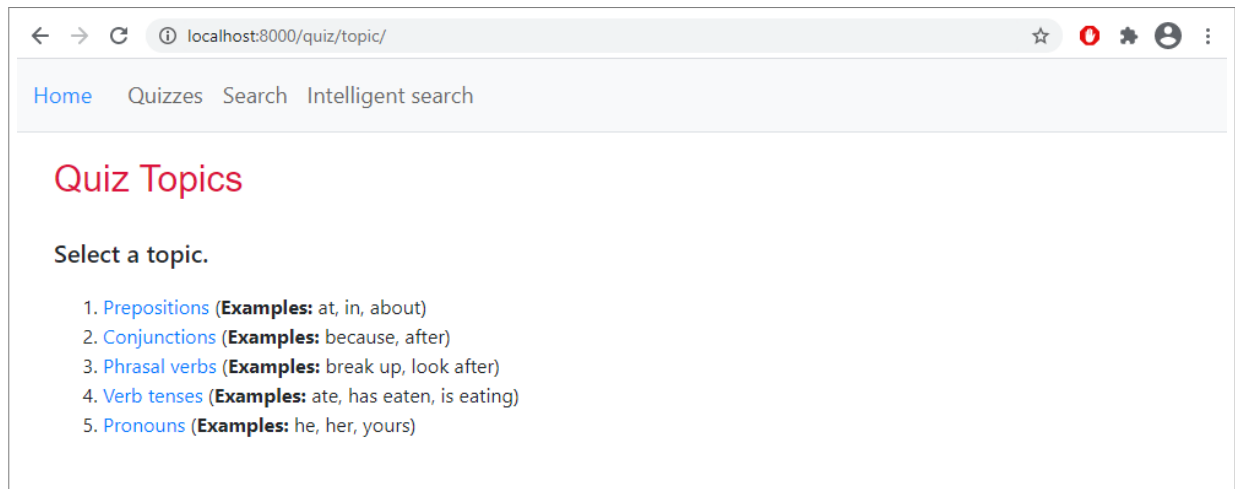
Next, we create two quizzes for each topic in the table `appquiz_modelquiz`:

The screenshot shows the 'Add model quiz' form in the Django administration interface. The browser address bar indicates the URL is `localhost:8000/admin/appquiz/modelquiz/add/`. The page header includes the Django logo and the text 'Django administration', along with a welcome message for 'ADMIN' and links for 'VIEW SITE', 'CHANGE PASSWORD', and 'LOG OUT'. The breadcrumb trail is 'Home > Appquiz > Model quizzes > Add model quiz'. The form contains two dropdown menus: 'Quiz topic:' with 'Prepositions' selected, and 'Quiz number:' with 'Quiz 1' selected. At the bottom right, there are three buttons: 'Save and add another', 'Save and continue editing', and 'SAVE'.

The screenshot shows the 'Select model quiz to change' view in the Django administration interface. The browser address bar indicates the URL is `localhost:8000/admin/appquiz/modelquiz/`. The page header is the same as the previous screenshot. The breadcrumb trail is 'Home > Appquiz > Model quizzes'. A green success message at the top states: 'The model quiz "2:Pronouns" was added successfully.' On the right side, there is a button labeled 'ADD MODEL QUIZ +'. Below this, there is an 'Action:' dropdown menu and a 'Go' button, followed by the text '0 of 10 selected'. The main content is a table with 10 rows, each representing a model quiz. The table has three columns: 'ID', 'QUIZ TOPIC', and 'QUIZ NUMBER'. Each row has a checkbox in the first column. Below the table, it says '10 model quizzes'.

<input type="checkbox"/>	ID	QUIZ TOPIC	QUIZ NUMBER
<input type="checkbox"/>	10	Pronouns	Quiz 2
<input type="checkbox"/>	9	Pronouns	Quiz 1
<input type="checkbox"/>	8	Verb tenses	Quiz 2
<input type="checkbox"/>	7	Verb tenses	Quiz 1
<input type="checkbox"/>	6	Phrasal verbs	Quiz 2
<input type="checkbox"/>	5	Phrasal verbs	Quiz 1
<input type="checkbox"/>	4	Conjunctions	Quiz 2
<input type="checkbox"/>	3	Conjunctions	Quiz 1
<input type="checkbox"/>	2	Prepositions	Quiz 2
<input type="checkbox"/>	1	Prepositions	Quiz 1

The **Quizzes** page will now show the list of topics. Selecting a topic will lead to the next page to select a quiz number.



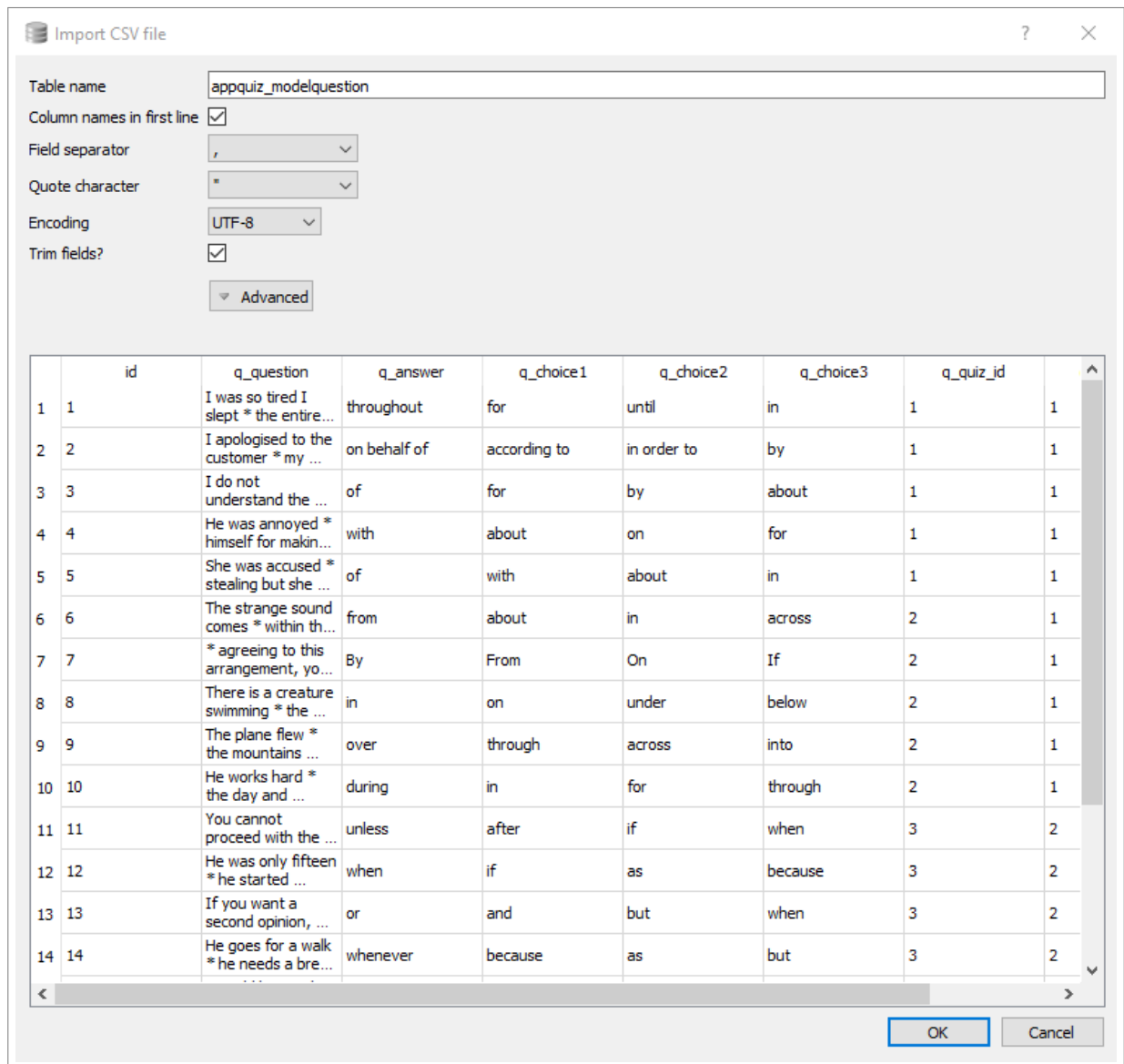
5.4.2 Importing Data

Earlier, we copied the following three datasets of raw data:

- `rawdata_quiz.csv` (46 *quiz questions*, for the database table `appquiz_modelquestion`)
- `rawdata_doc.csv` (850 *question bank questions*, for the database table `appsearch_modelquestionbank`)
- `rawdata_query.csv` (152 *question bank questions*, for the database table `appsearch_modelquestionbank`)

We will import the data in these three datasets to the database.

Using **DB Browser**, import the data (File > Import > Table from CSV file...) from `rawdata_quiz.csv` to the table `appquiz_modelquestion`. We need to amend the **Table name** field:



Import CSV file

Table name:

Column names in first line: ☒

Field separator: ▼

Quote character: ▼

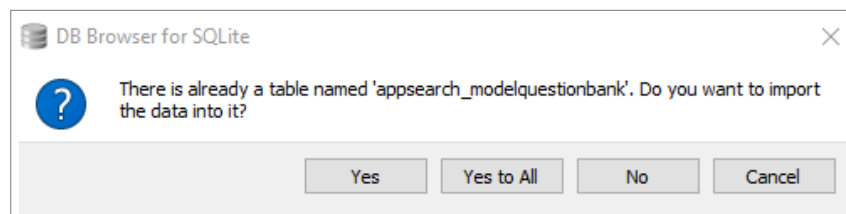
Encoding: ▼

Trim fields?: ☒

▼ Advanced

	id	q_question	q_answer	q_choice1	q_choice2	q_choice3	q_quiz_id	
1	1	I was so tired I slept * the entire...	throughout	for	until	in	1	1
2	2	I apologised to the customer * my ...	on behalf of	according to	in order to	by	1	1
3	3	I do not understand the ...	of	for	by	about	1	1
4	4	He was annoyed * himself for makin...	with	about	on	for	1	1
5	5	She was accused * stealing but she ...	of	with	about	in	1	1
6	6	The strange sound comes * within th...	from	about	in	across	2	1
7	7	* agreeing to this arrangement, yo...	By	From	On	If	2	1
8	8	There is a creature swimming * the ...	in	on	under	below	2	1
9	9	The plane flew * the mountains ...	over	through	across	into	2	1
10	10	He works hard * the day and ...	during	in	for	through	2	1
11	11	You cannot proceed with the ...	unless	after	if	when	3	2
12	12	He was only fifteen * he started ...	when	if	as	because	3	2
13	13	If you want a second opinion, ...	or	and	but	when	3	2
14	14	He goes for a walk * he needs a bre...	whenever	because	as	but	3	2

OK Cancel



DB Browser for SQLite

There is already a table named 'appsearch_modelquestionbank'. Do you want to import the data into it?

Yes Yes to All No Cancel

Now we can browse the data:

DB Browser for SQLite - D:\projects\grammar\django\db.sqlite3

File Edit View Tools Help

New Database Open Database Write Changes Revert Changes Open Project Save Project Attach Database Close Database

Database Structure Browse Data Edit Pragma Execute SQL

Table: appquiz_modelquestion

	id	q_question	q_answer	q_choice1	q_choice2	q_choice3	q_quiz_id	q_topic_id
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	I was so tired I slept * the entire movie.	throughout	for	until	in	1	1
2	2	I apologised to the customer * my company.	on behalf of	according to	in order to	by	1	1
3	3	I do not understand the cause * his anger.	of	for	by	about	1	1
4	4	He was annoyed * himself for making the same ...	with	about	on	for	1	1
5	5	She was accused * stealing but she claimed ...	of	with	about	in	1	1
6	6	The strange sound comes * within the hall.	from	about	in	across	2	1
7	7	* agreeing to this arrangement, you will need to ...	By	From	On	If	2	1
8	8	There is a creature swimming * the water.	in	on	under	below	2	1
9	9	The plane flew * the mountains towards the ...	over	through	across	into	2	1
10	10	He works hard * the day and parties even hard a...	during	in	for	through	2	1
11	11	You cannot proceed with the work * you get a ...	unless	after	if	when	3	2
12	12	He was only fifteen * he started supporting his ...	when	if	as	because	3	2

1 - 13 of 46

Go to: 1

UTF-8

Next, import the data from `rawdata.doc.csv` and `rawdata.query.csv` to the table `appsearch_modelquestionbank`:

Import CSV file

Table name: appsearch_modelquestionbank

Column names in first line: ☒

Field separator: ,

Quote character: "

Encoding: UTF-8

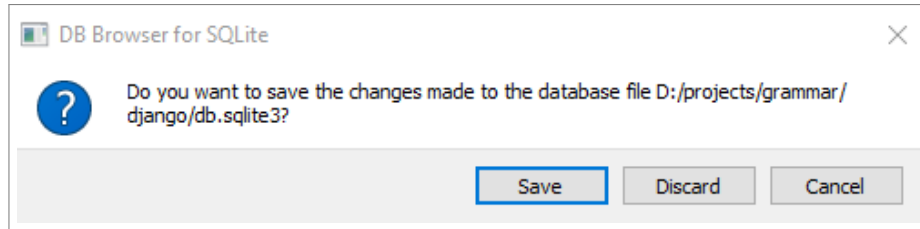
Trim fields?: ☒

Advanced

	id	qb_question	qb_answer	qb_choice1	qb_choice2	qb_choice3	qb_topic_id
1	1	The Olympics team is aiming * a gold ...	for	about	with	after	1
2	2	She has a talent * art.	for	about	with	of	1
3	3	I felt sorry * my colleagues who I...	for	about	with	by	1
4	4	Everybody knows the reason * his ...	for	about	with	by	1

OK Cancel

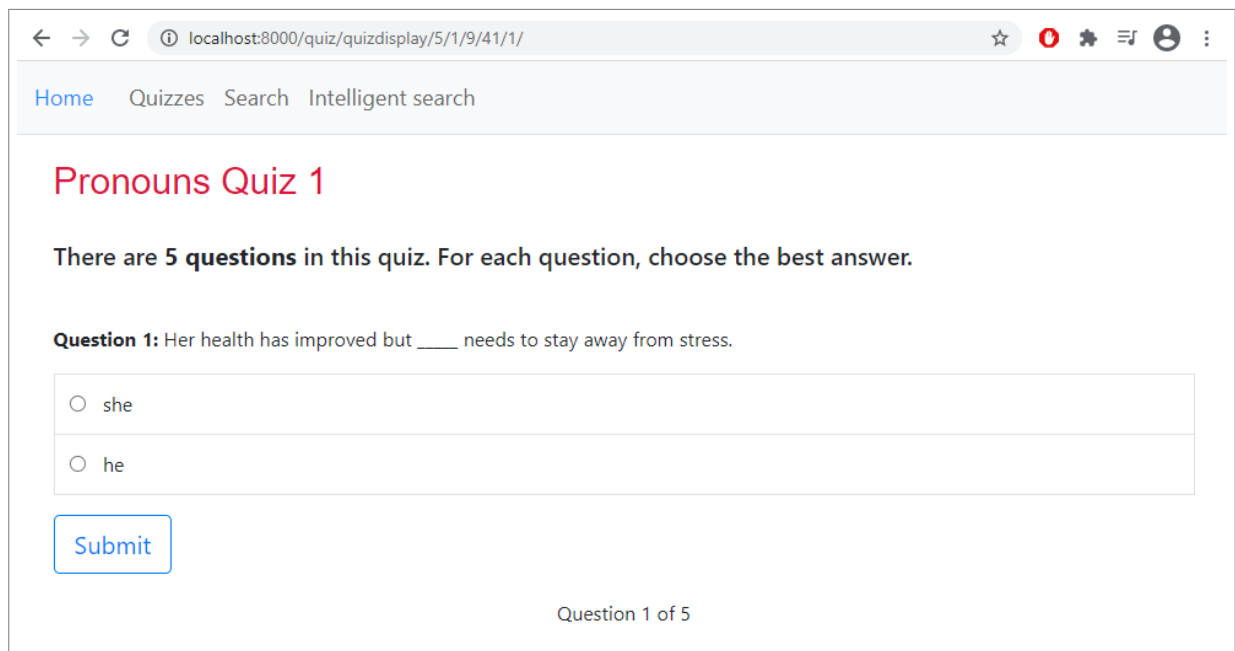
We need to close and save the database (File > Close Database):



5.5 Quizzes and Search

5.5.1 Quizzes Page

On the **Quizzes** page, we will be able to select a quiz number and do the quiz questions, leading to a page showing the quiz results and answers:



← → ↻ ⓘ localhost:8000/quiz/quizdisplay/5/1/9/41/1/ ☆ 0 ⚙ ⌵ 👤 ⋮

[Home](#) [Quizzes](#) [Search](#) [Intelligent search](#)

Pronouns Quiz 1

There are **5 questions** in this quiz. For each question, choose the best answer.

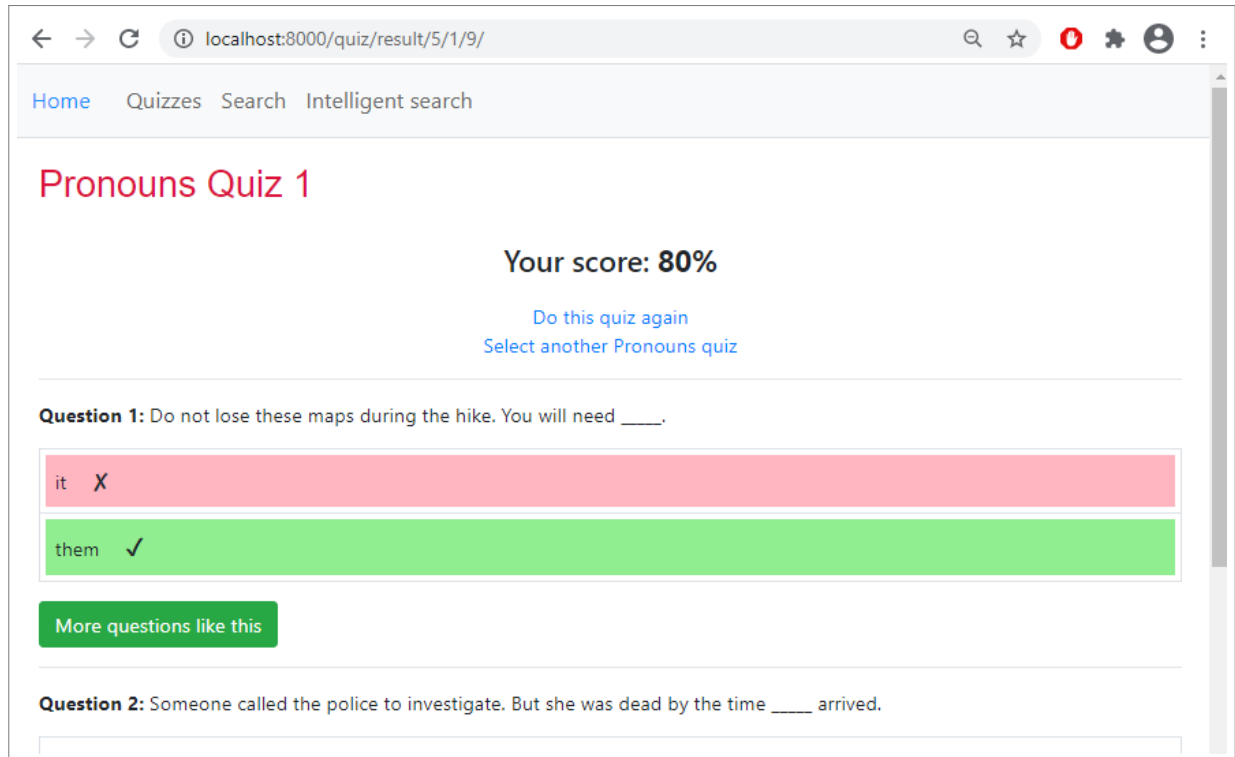
Question 1: Her health has improved but ____ needs to stay away from stress.

☐ she

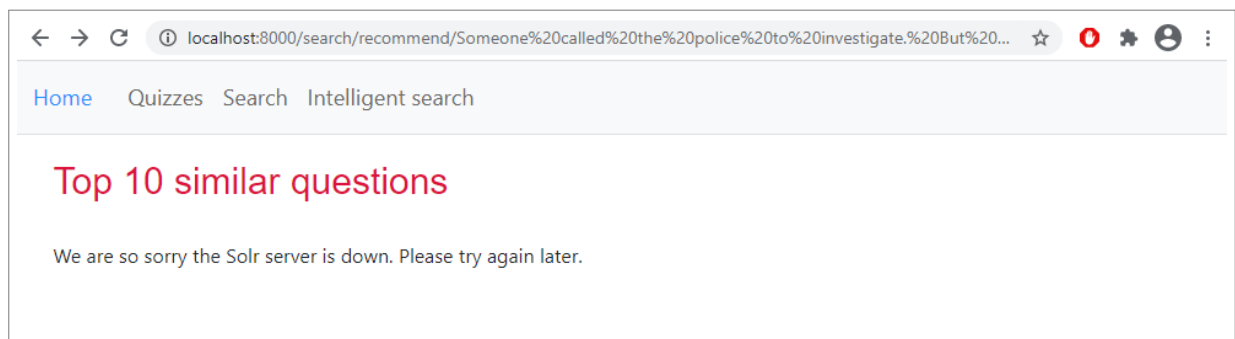
☐ he

[Submit](#)

Question 1 of 5



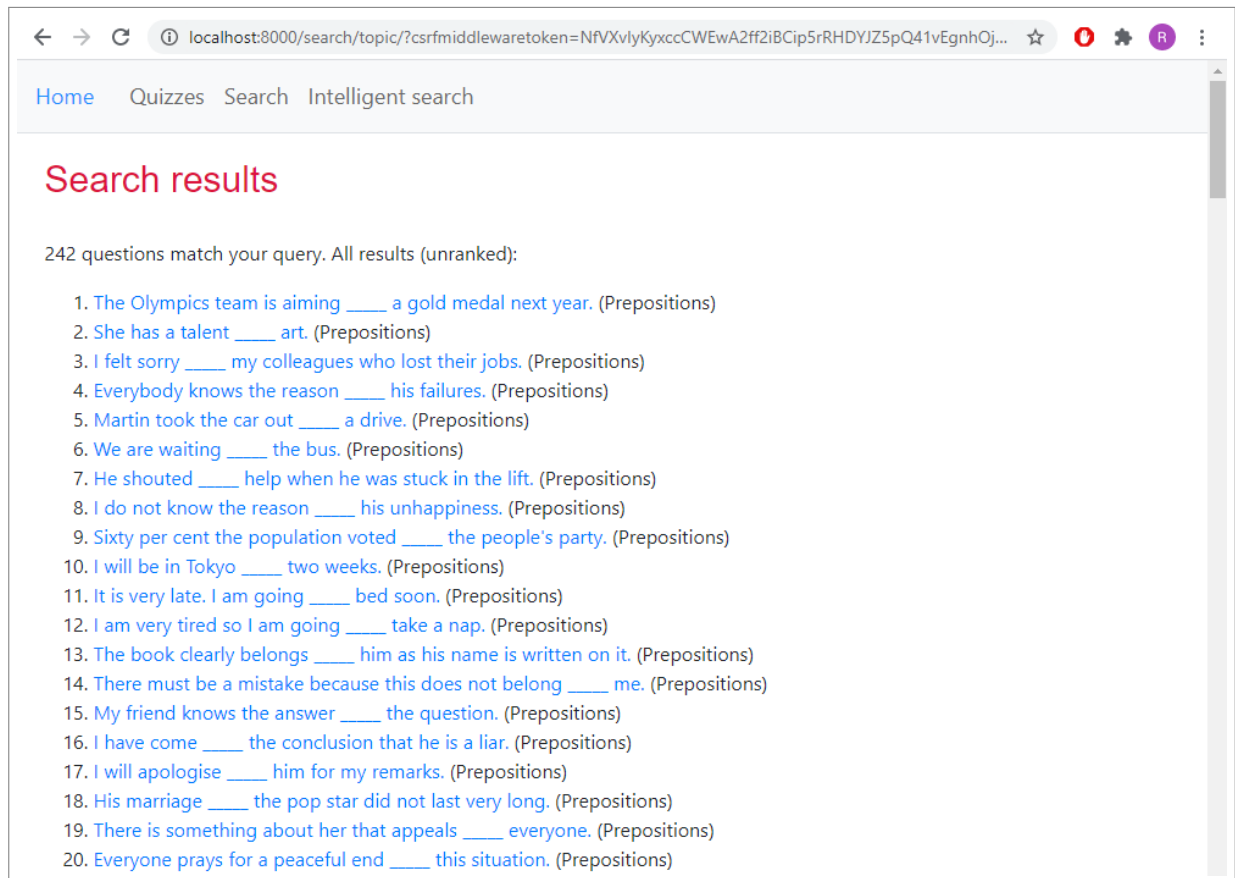
Clicking **More questions like this** on the quiz results page will send a query to the Solr server which will return the top 10 similar questions. As we have not set up the Solr server yet, we will again see the following message:



5.5.2 Search Page

Now that the database is fully populated, we will be able to use these three search features on the **Search** page. The system will return grammar questions from the table `appsearch_modelquestionbank`:

- **Search by topic**



- Search by question text

The screenshot shows a web browser window with the address bar displaying `localhost:8000/search/kw/?query_question=play`. The browser's navigation bar includes links for Home, Quizzes, Search, and Intelligent search. The main content area is titled "Search results" in red. Below the title, it states "Query: **play** (to match question text)" and "9 questions match your query. All results (unranked):". A list of 9 questions follows, each with a blank space for an answer and a category in parentheses:

1. There is much team spirit ____ the players. (Prepositions)
2. This is not going to work ____ everyone plays his part. (Conjunctions)
3. ____ we had played with more guile we could have won. (Conjunctions)
4. I will ____ the game if any player is not dress appropriately. (Phrasal verbs)
5. She ____ as a waiter during the day and plays the piano at the club at night. (Verb tenses)
6. The children stopped playing ____ game and came running for the snacks. (Pronouns)
7. I asked her ____ she was keen to play the game. (Conjunctions)
8. There was a good coach in the team who helped ____ become better players. (Pronouns)
9. Look at those two dogs. I have never seen ____ so playful before. (Pronouns)

- Search by answer choice text

The screenshot shows a web browser window with the address bar displaying `localhost:8000/search/kw/?query_choice=break`. The browser's navigation bar includes links for Home, Quizzes, Search, and Intelligent search. The main content area is titled "Search results" in red. Below the title, it states "Query: **break** (to match answer choice text)" and "4 questions match your query. All results (unranked):". A list of 4 questions follows, each with a blank space for an answer and a category in parentheses:

1. In musicals, actors ____ songs like they are having a conversation. (Phrasal verbs)
2. The company is trying to ____ the clean energy industry. (Phrasal verbs)
3. Her training involves ____ sprints in between slow jogs. (Phrasal verbs)
4. War will ____ if the two countries do not compromise. (Phrasal verbs)

In the next chapter, we will set up the Solr server and upload data to the Solr server so that the Solr search features will work. Then we will set up Learning To Rank so that the intelligent Solr search features will work.

Chapter 6

Solr Search

6.1 Solr Version and Documentation

We will use Solr 7.7.3. However, if you wish to use Haystack, Solr 6.6.6 is officially the latest version that Haystack 2.8.1 supports. We will not use Haystack, as I will explain in Section 6.5. Solr documentation archives are at:

<https://lucene.apache.org/solr/guide/>

The glossary is useful for understanding the components of Solr:

https://lucene.apache.org/solr/guide/7_7/solr-glossary.html

6.2 Java Runtime Environment

The only requirement for running Solr is Java Runtime Environment (JRE) version 1.8 or later (https://lucene.apache.org/solr/guide/7_7/solr-system-requirements.html). Download and install JRE from:

<https://www.java.com/en/download/>.

Check that the installation is successful:

```
PS C:\> java -version
java version "1.8.0_271"
Java(TM) SE Runtime Environment (build 1.8.0_271-b09)
Java HotSpot(TM) 64-Bit Server VM (build 25.271-b09, mixed mode)
```

6.3 Code

Copy the following files from the repository directory:

- /grammar/setup_solr_upload_data.ipynb
- /grammar/setup_solr_upload_feature.ipynb
- /grammar/setup_solr_upload_model.ipynb
- /grammar/solr.py

6.4 Solr Server

6.4.1 Solr Node

We will now set up a Solr server, also called a *Solr node*. This is a three-step process. We download the ZIP file of the required version of Solr, decompress the file to extract the Solr directory, and start the server from the Solr directory. First, download `solr-7.7.3.zip` from:

<https://archive.apache.org/dist/lucene/solr/7.7.3/>

The ZIP file contains a single directory named `solr-7.7.3`. Move this directory to the project directory. At this point, it would be good to explore the Solr file system. The location that we will need to access most is `solr-7.7.3/server/solr`. When we create a new *core*, a directory for the core will be created there. Let's rename the Solr directory to `solr7`. In Windows PowerShell, start the Solr server:

```
PS D:\grammar\solr7> bin\solr start
Waiting up to 30 to see Solr running on port 8983
Started Solr server on port 8983. Happy searching!
```

Other than `start`, other Solr commands are `stop`, `restart`, and `status`. The following shows the output of `status` if a Solr node is running:

```
PS D:\grammar\solr7> bin\solr status

Found Solr process 5424 running on port 8983
{
  "solr_home":"D:\\projects\\grammar\\solr7\\server\\solr",
  "version":"7.7.3 1a0d2a901dfec93676b0fe8be425101ceb754b85 - noble - 2020-04-21 10:37:32",
  "startTime":"2021-01-27T22:23:12.354Z",
  "uptime":"0 days, 0 hours, 28 minutes, 6 seconds",
  "memory":"40.7 MB (%8.3) of 490.7 MB"}
```

We will see the Solr Administration User Interface (Solr Admin UI) at <http://localhost:8983/>:

The screenshot displays the Solr Administration User Interface (Solr Admin UI) in a web browser. The browser's address bar shows the URL `localhost:8983/solr/#/`. The interface is divided into several sections:

- Instance:** Shows the status of the Solr instance, including the start time (about a minute ago).
- Versions:** Lists the versions of the Solr components (solr-spec, solr-impl, lucene-spec, lucene-impl) and their corresponding build numbers and timestamps.
- JVM:** Displays the Java runtime environment details, including the Oracle Corporation Java HotSpot(TM) 64-Bit Server VM 1.8.0_271, 25.271-b09, and the number of processors (12). It also lists various JVM arguments (e.g., `-DSTOPKEY=solrrocks`, `-DSTOPPORT=7983`, `-Djava.io.tmpdir=D:\projects\grammar\solr7\server\tmp`, etc.).
- System:** Shows system-level metrics, including Physical Memory (29.8%), Swap Space (47.0%), and JVM-Memory (10.4%).
- JVM-Memory:** Displays the memory usage of the JVM, showing 51.08 MB used out of 490.69 MB available.

The interface also includes a sidebar with navigation links (Dashboard, Logging, Core Admin, Java Properties, Thread Dump) and a footer with links to Documentation, Issue Tracker, IRC Channel, Community forum, and Solr Query Syntax.

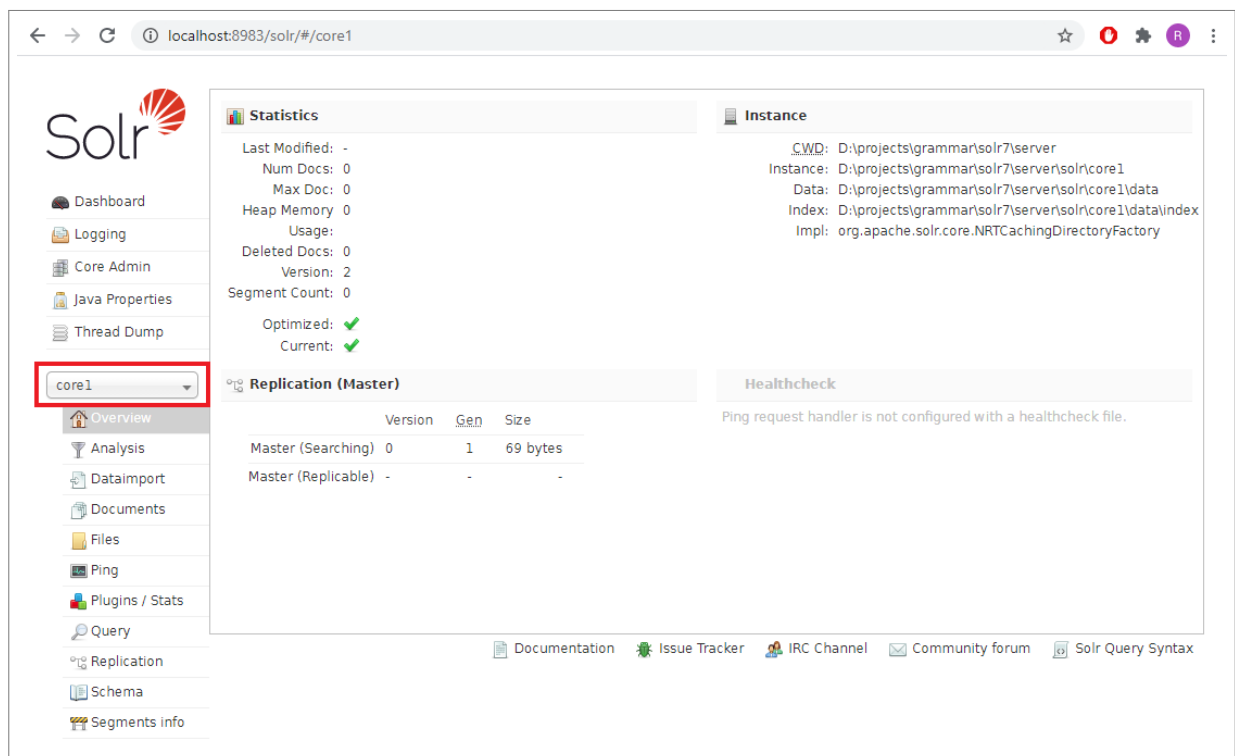
6.4.2 Creating a Core

A *Solr instance*, representing a logical index, is called a *core*. We can create and run multiple cores in a single Solr node. Let's create a core named `core1`:

```
PS D:\grammar\solr7> bin\solr create -c core1
WARNING: Using _default configset with data driven schema functionality.
NOT RECOMMENDED for production use.
To turn off: bin\solr config -c core1 -p 8983 -action set-user-property
-property update.autoCreateFields -value false
```

Created new core 'core1'

Refresh the Solr Admin UI and we will now be able to select the newly created core:



The screenshot shows the Solr Admin UI in a web browser. The address bar indicates the URL is `localhost:8983/solr/#/core1`. The left sidebar contains a menu with options: Dashboard, Logging, Core Admin, Java Properties, Thread Dump, and a dropdown menu currently showing `core1`. Below the dropdown is a list of links: Overview, Analysis, Dataimport, Documents, Files, Ping, Plugins / Stats, Query, Replication, Schema, and Segments info. The main content area is divided into several sections: **Statistics** (Last Modified: -, Num Docs: 0, Max Doc: 0, Heap Memory: 0, Usage: -, Deleted Docs: 0, Version: 2, Segment Count: 0, Optimized: ✓, Current: ✓), **Instance** (CWD: D:\projects\grammar\solr7\server, Instance: D:\projects\grammar\solr7\server\solr\core1, Data: D:\projects\grammar\solr7\server\solr\core1\data, Index: D:\projects\grammar\solr7\server\solr\core1\data\index, Impl: org.apache.solr.core.NRTCachingDirectoryFactory), **Replication (Master)** (a table with columns Version, Gen, and Size), and **Healthcheck** (Ping request handler is not configured with a healthcheck file.). At the bottom of the main panel, there are links to Documentation, Issue Tracker, IRC Channel, Community forum, and Solr Query Syntax.

	Version	Gen	Size
Master (Searching)	0	1	69 bytes
Master (Replicable)	-	-	-

6.4.3 Deleting a Core

To delete an existing core:

```
PS D:\grammar\solr7> bin\solr delete -c core_name
```

6.4.4 Core Directory

It is important to be familiar with the files in the directory created for the core:

```
solr7/
  server/
    solr/
      core1/
        conf/
          lang/
            ...
            managed-schema
            params.json
            protwords.txt
            solrconfig.xml
            stopwords.txt
            synonyms.txt
        data/
          index/
            segments_1
            write.lock
            snapshot_metadata/
            tlog/
        core.properties
```

The two most important files are `solrconfig.xml` and `managed-schema.xml`. We need to add information to `solrconfig.xml` when we set up Learning To Rank. `managed-schema.xml` is generally left untouched unless we wish to customize the search behavior.

`managed-schema.xml` defines the data *fields* and *field types*, which will determine the search behavior. When we upload data to this core, Solr will index the data and update `managed-schema.xml` with new fields. Solr will also guess the field type for each new field created. We should check the field type for each field and modify it if necessary.

6.5 Uploading Data to Solr

6.5.1 Uploading Methods

We need to upload data to the Solr server in order to do a Solr search. Solr will then index the data. There are two ways to upload data to Solr:

- **Method 1:** From the database to Solr by Haystack's API
- **Method 2:** From a CSV file to Solr by Solr's REST API

While I was learning to use the Solr REST API, I experimented with Haystack. Haystack provides the API to upload data to Solr from a database in Django, by adding some scripts to the Django setup and some code to the scripts `/config/settings.py` and `/config/urls.py`.

The process might appear easier compared to writing Python code to upload data from a CSV file (our code is in `setup_solr_upload_data.ipynb`). However, I found the Haystack process to be a black box. Whenever there was an error, I could not be sure which part of the process I did wrongly. The flexibility and clarity of using Solr's API is easier in the long run.

Nevertheless, Haystack is an option worth exploring, because it also supports **Elasticsearch** and other search software. For more details, refer to the documentation:

<https://django-haystack.readthedocs.io/en/master/>

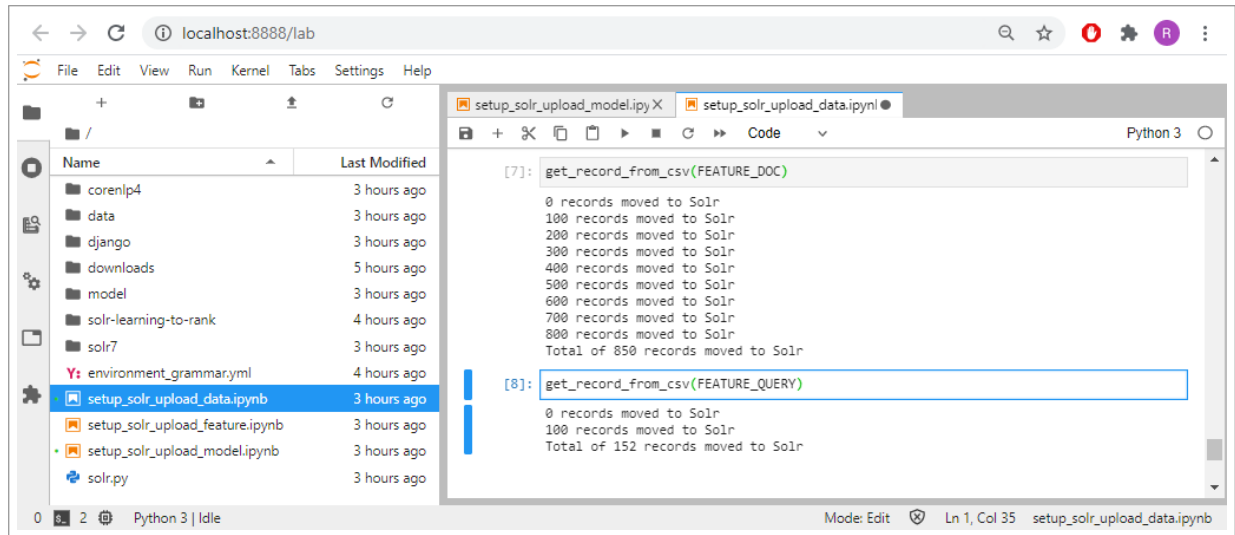
6.5.2 Uploading Data by Solr's REST API

We will upload data to Solr by Solr's REST API, because the data in the database is not what we need for intelligent Solr search. For the 1002 raw grammar questions in the question bank (see Section 5.4.2), we need to extract features and upload the features data to Solr. Copy the following directory from the repository directory:

- `/grammar/data/feature`

This directory contains the datasets `feature_doc.csv` and `feature_query.csv`.

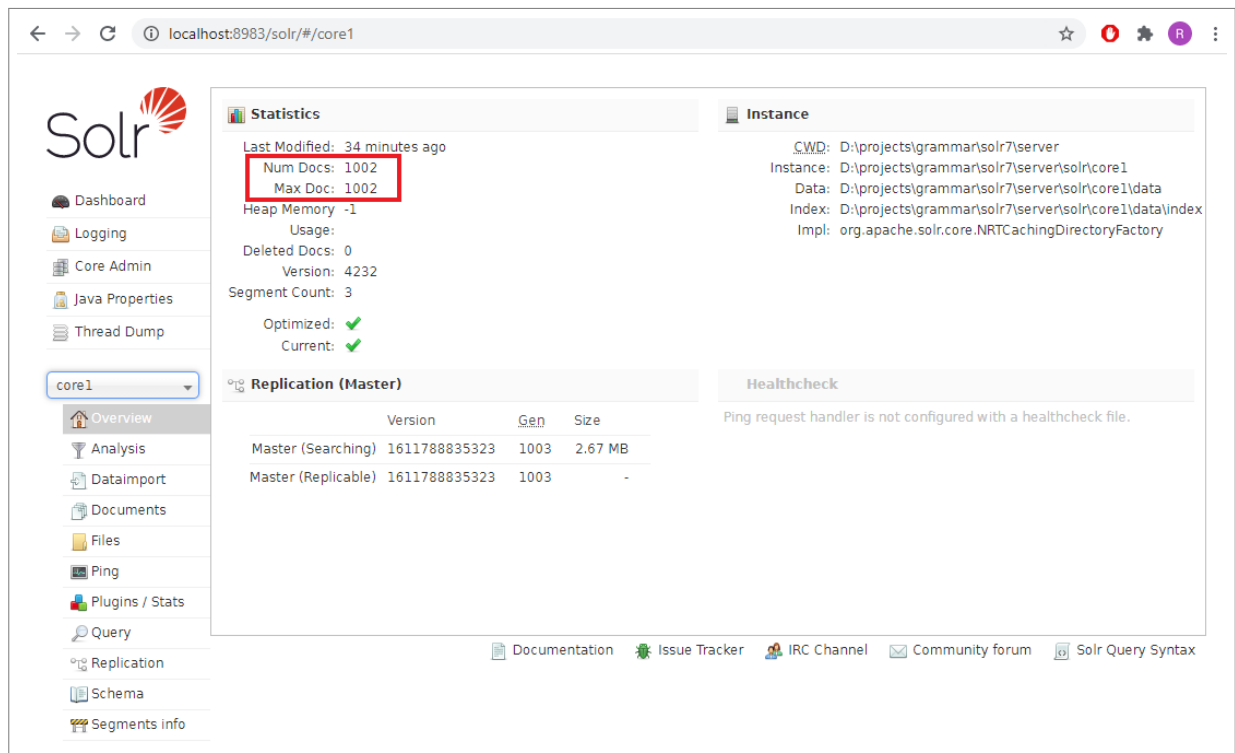
To upload data to Solr, open `setup_solr_upload_data.ipynb` and run it.



```
[7]: get_record_from_csv(FEATURE_DOC)
0 records moved to Solr
100 records moved to Solr
200 records moved to Solr
300 records moved to Solr
400 records moved to Solr
500 records moved to Solr
600 records moved to Solr
700 records moved to Solr
800 records moved to Solr
Total of 850 records moved to Solr

[8]: get_record_from_csv(FEATURE_QUERY)
0 records moved to Solr
100 records moved to Solr
Total of 152 records moved to Solr
```

When all 1002 records are uploaded (it will take a while), check the Solr Admin UI. We will now see 1002 documents in `core1`:



Solr

Dashboard
Logging
Core Admin
Java Properties
Thread Dump

core1

Overview
Analysis
Dataimport
Documents
Files
Ping
Plugins / Stats
Query
Replication
Schema
Segments info

Statistics

Last Modified: 34 minutes ago

Num Docs: 1002
Max Doc: 1002

Heap Memory -1

Usage:

Deleted Docs: 0

Version: 4232

Segment Count: 3

Optimized: ✓
Current: ✓

Replication (Master)

	Version	Gen	Size
Master (Searching)	1611788835323	1003	2.67 MB
Master (Replicable)	1611788835323	1003	-

Instance

CWD: D:\projects\grammar\solr7\server
Instance: D:\projects\grammar\solr7\server\solr\core1
Data: D:\projects\grammar\solr7\server\solr\core1\data
Index: D:\projects\grammar\solr7\server\solr\core1\data\index
Impl: org.apache.solr.core.NRTCachingDirectoryFactory

Healthcheck

Ping request handler is not configured with a healthcheck file.

[Documentation](#) [Issue Tracker](#) [IRC Channel](#) [Community forum](#) [Solr Query Syntax](#)

6.6 Deleting Data From Solr

We can delete data in Solr from the Solr Admin UI. To delete all data, on the **Documents** interface, select the **Document Type** Solr Command (raw XML or JSON) and enter the XML command:

```
<delete><query>*:*</query></delete>
```

or the JSON command:

```
{'delete': {'query': '*:*'}}
```

Click **Submit Document** to delete all data:

The screenshot shows the Solr Admin UI at the URL `localhost:8983/solr/#/core1/documents`. The interface includes a sidebar with navigation links: Dashboard, Logging, Core Admin, Java Properties, Thread Dump, core1 (selected), Overview, Analysis, Dataimport, Documents (highlighted), Files, Ping, Plugins / Stats, Query, Replication, Schema, and Segments info. The main content area is titled "Request-Handler (qt)" and contains the following fields:

- Request-Handler (qt)**: `/update`
- Document Type**: `Solr Command (raw XML or JSON)` (selected from a dropdown)
- Document(s)**: `{'delete': {'query': '*:*'}}`
- Commit Within**: `1000`
- Overwrite**: `true`
- Submit Document**: A blue button to execute the command.

A [Documentation](#) link is visible in the bottom right corner.

6.7 Solr Search

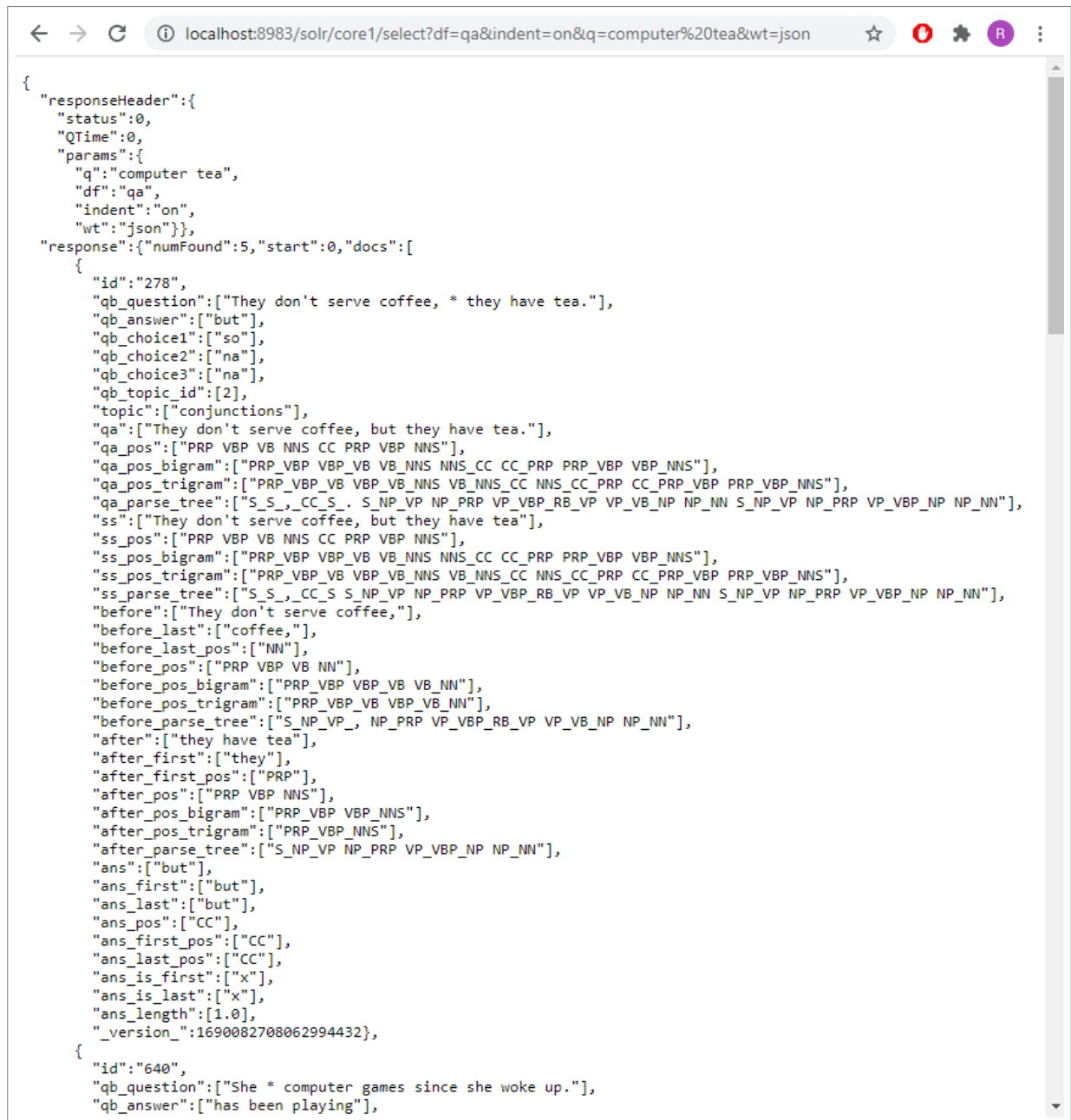
6.7.1 Solr Admin UI

After data is uploaded to Solr, we can search for documents in the Solr Admin UI. The data in each document is organized into 40 fields. In the screenshot below, the query is “computer tea”. We specify the default field (df) as qa, so that Solr knows which field to match the query with. Solr returns five documents matching the query. In other words, at least one of the words in the query (“computer” or “tea”) is found in the field qa in each of the five documents:

The screenshot shows the Solr Admin UI interface. On the left is a sidebar with navigation links: Dashboard, Logging, Core Admin, Java Properties, Thread Dump, core1 (selected), Overview, Analysis, Dataimport, Documents, Files, Ping, Plugins / Stats (highlighted), Replication, Schema, and Segments info. The main area is titled 'Request-Handler (qt)' and shows a search form. The 'q' field contains 'computer tea'. The 'df' field is set to 'qa'. The 'wt' field is set to 'json'. The 'indent' checkbox is checked. Below the form is an 'Execute Query' button. To the right of the form, the URL bar shows the request: `http://localhost:8983/solr/core1/select?df=qa&indent=on&q=computer%20tea&wt=json`. The response is a JSON object showing the search results, including the number of documents found (5) and the details of the first document.

```
{
  "responseHeader": {
    "status": 0,
    "QTime": 0,
    "params": {
      "q": "computer tea",
      "df": "qa",
      "indent": "on",
      "wt": "json",
      "_: "1611788447112"
    }
  },
  "response": {
    "numFound": 5,
    "start": 0,
    "docs": [
      {
        "id": "278",
        "qb_question": ["They don't serve coffee, * they have tea."],
        "qb_answer": ["but"],
        "qb_choice1": ["so"],
        "qb_choice2": ["na"],
        "qb_choice3": ["na"],
        "qb_topic_id": [2],
        "topic": ["conjunctions"],
        "qa": ["They don't serve coffee, but they have tea."],
        "qa_pos": ["PRP VBP VB NNS CC PRP VBP NNS"],
        "qa_pos_bigram": ["PRP_VBP VBP_VB VB_NNS NNS_CC CC_PRP PRP_VBP VBP_NNS"],
        "qa_pos_trigram": ["PRP_VBP_VB VBP_VB_NNS VB_NNS_CC NNS_CC_PRP CC_PRP_VBP PRP_VBP_NNS"],
        "qa_parse_tree": ["S_S_,CC_S_ S_NP_VP NP_PRP VP_VBP_RB_VP VP_VB_NP NP_NN S_NP_VP NP_PRP VP_VBP_NP NP_NN"],
        "ss": ["They don't serve coffee, but they have tea"],
        "ss_pos": ["PRP VBP VB NNS CC PRP VBP NNS"],
        "ss_pos_bigram": ["PRP_VBP VBP_VB VB_NNS NNS_CC CC_PRP PRP_VBP VBP_NNS"],
        "ss_pos_trigram": ["PRP_VBP_VB VBP_VB_NNS VB_NNS_CC NNS_CC_PRP CC_PRP_VBP PRP_VBP_NNS"],
        "ss_parse_tree": ["S_S_,CC_S_ S_NP_VP NP_PRP VP_VBP_RB_VP VP_VB_NP NP_NN S_NP_VP NP_PRP VP_VBP_NP NP_NN"],
        "before": ["They don't serve coffee,"],
        "before_last": ["coffee,"],
        "before_last_pos": ["NN"],
        "before_pos": ["PRP VBP VB NN"],
        "before_pos_bigram": ["PRP_VBP VBP_VB VB_NN"],
        "before_pos_trigram": ["PRP_VBP_VB VBP_VB_NN"],
        "before_parse_tree": ["S_NP_VP_, NP_PRP VP_VBP_RB_VP VP_VB_NP NP_NN"],
        "after": ["they have tea"],
      }
    ]
  }
}
```

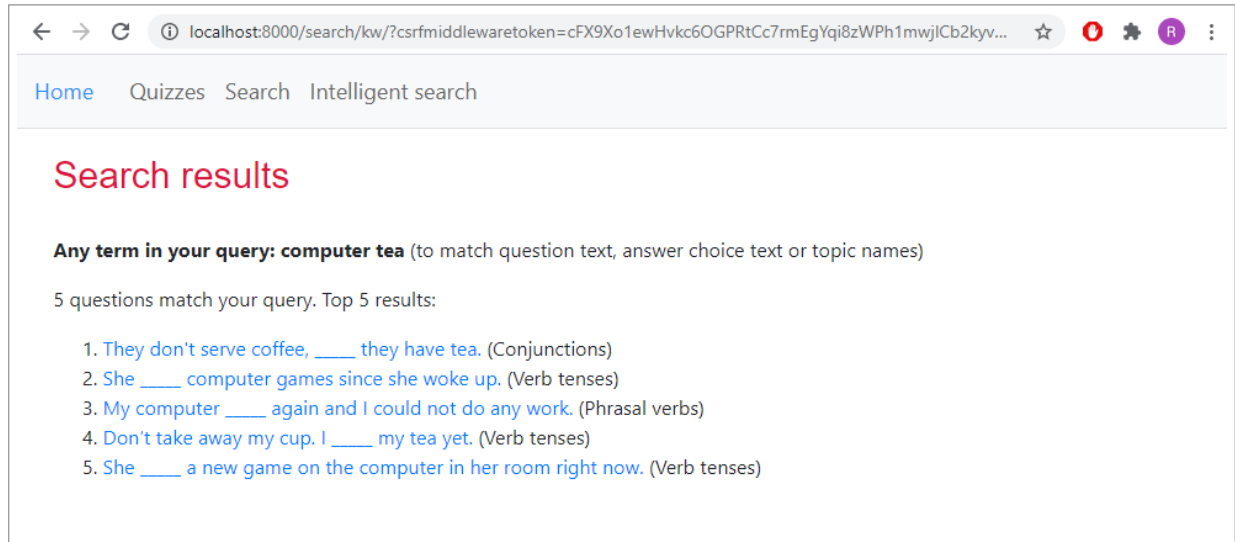
At the top of the query interface on the previous page, the query is given as a URL. Click on this URL and we will see the results on a web page:



```
{
  "responseHeader":{
    "status":0,
    "QTime":0,
    "params":{
      "q":"computer tea",
      "df":"qa",
      "indent":"on",
      "wt":"json"}},
  "response":{"numFound":5,"start":0,"docs":[
    {
      "id":"278",
      "qb_question":["They don't serve coffee, * they have tea."],
      "qb_answer":["but"],
      "qb_choice1":["so"],
      "qb_choice2":["na"],
      "qb_choice3":["na"],
      "qb_topic_id":[2],
      "topic":["conjunctions"],
      "qa":["They don't serve coffee, but they have tea."],
      "qa_pos":["PRP VBP VB NNS CC PRP VBP NNS"],
      "qa_pos_bigram":["PRP_VBP VBP_VB VB_NNS NNS_CC CC_PRP PRP_VBP VBP_NNS"],
      "qa_pos_trigram":["PRP_VBP_VB VBP_VB_NNS VB_NNS_CC NNS_CC_PRP CC_PRP_VBP PRP_VBP_NNS"],
      "qa_parse_tree":["S_S_,CC_S_ S_NP_VP NP_PRP VP_VBP_RB_VP VP_VB_NP NP_NN S_NP_VP NP_PRP VP_VBP_NP NP_NN"],
      "ss":["They don't serve coffee, but they have tea"],
      "ss_pos":["PRP VBP VB NNS CC PRP VBP NNS"],
      "ss_pos_bigram":["PRP_VBP VBP_VB VB_NNS NNS_CC CC_PRP PRP_VBP VBP_NNS"],
      "ss_pos_trigram":["PRP_VBP_VB VBP_VB_NNS VB_NNS_CC NNS_CC_PRP CC_PRP_VBP PRP_VBP_NNS"],
      "ss_parse_tree":["S_S_,CC_S S_NP_VP NP_PRP VP_VBP_RB_VP VP_VB_NP NP_NN S_NP_VP NP_PRP VP_VBP_NP NP_NN"],
      "before":["They don't serve coffee,"],
      "before_last":["coffee,"],
      "before_last_pos":["NN"],
      "before_pos":["PRP VBP VB NN"],
      "before_pos_bigram":["PRP_VBP VBP_VB VB_NN"],
      "before_pos_trigram":["PRP_VBP_VB VBP_VB_NN"],
      "before_parse_tree":["S_NP_VP_, NP_PRP VP_VBP_RB_VP VP_VB_NP NP_NN"],
      "after":["they have tea"],
      "after_first":["they"],
      "after_first_pos":["PRP"],
      "after_pos":["PRP VBP NNS"],
      "after_pos_bigram":["PRP_VBP VBP_NNS"],
      "after_pos_trigram":["PRP_VBP_NNS"],
      "after_parse_tree":["S_NP_VP NP_PRP VP_VBP_NP NP_NN"],
      "ans":["but"],
      "ans_first":["but"],
      "ans_last":["but"],
      "ans_pos":["CC"],
      "ans_first_pos":["CC"],
      "ans_last_pos":["CC"],
      "ans_is_first":["x"],
      "ans_is_last":["x"],
      "ans_length":[1.0],
      "_version_":1690082708062994432},
    {
      "id":"640",
      "qb_question":["She * computer games since she woke up."],
      "qb_answer":["has been playing"],
```

6.7.2 Website

On the **Search** page, do a Solr search. The query "computer tea" will return the same five documents in the same ranking order:



Chapter 7

Intelligent Solr Search

7.1 Learning To Rank Plugin

To enable Learning To Rank (LTR) for `core1`, we need to add the following configuration information to the file `solrconfig.xml` in `solr7/server/solr/core1/conf`:

- the library `solr-ltr-7.7.3.jar`, located in `solr7/dist/`
- the Learning To Rank *query parser* named `ltr`
- the *feature values cache* named `QUERY_DOC_FV`
- the *transformer* named `features`

Put the following code before `</config>` at the end of `solrconfig.xml`:

```
<lib dir="${solr.install.dir:../../..}/dist/" regex="solr-ltr-\\d.*\\.jar" />

<queryParser name="ltr" class="org.apache.solr.ltr.search.LTRQParserPlugin"/>

<cache name="QUERY_DOC_FV"
  class="solr.search.LRUCache"
  size="4096"
  initialSize="2048"
  autowarmCount="4096"
  regenerator="solr.search.NoOpRegenerator" />

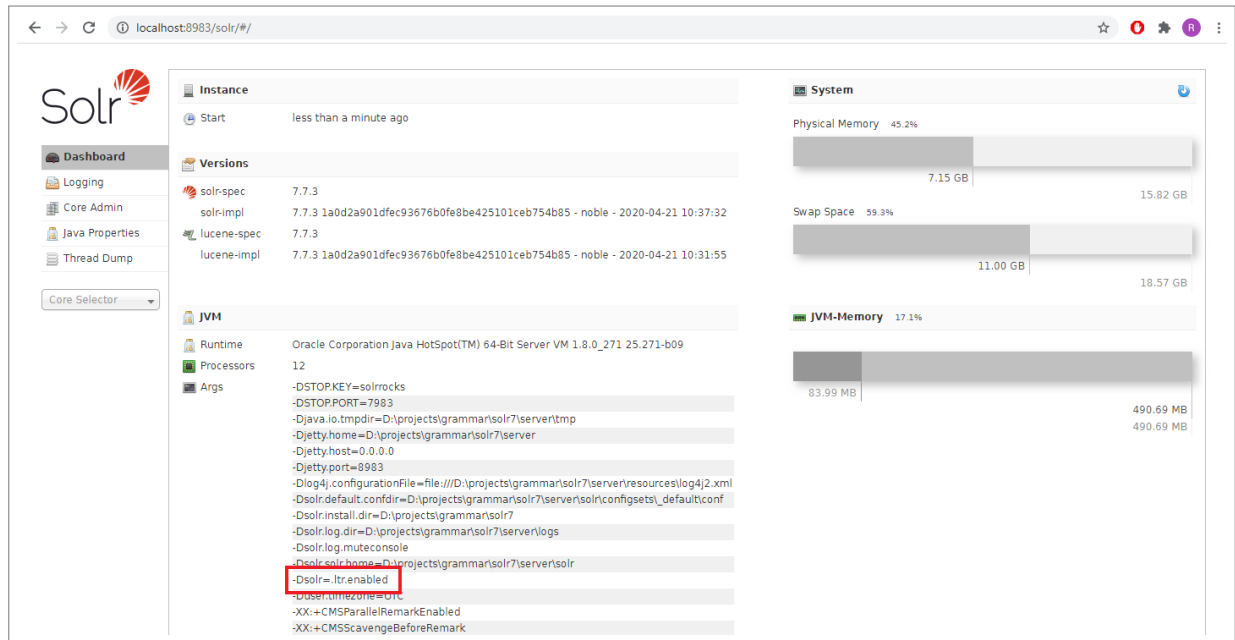
<transformer name="features"
  class="org.apache.solr.ltr.response.transform.LTRFeatureLoggerTransformerFactory">
  <str name="fvCacheName">QUERY_DOC_FV</str>
</transformer>
```

Restart Solr with LTR enabled:

```
PS D:\grammar\solr7> bin\solr restart -Dsolr.ltr.enabled -p 8983
Stopping Solr process 19952 running on port 8983
```

```
Waiting for 0 seconds, press a key to continue ...
Waiting up to 30 to see Solr running on port 8983
Started Solr server on port 8983. Happy searching!
```

Refresh the Solr Admin UI. The Dashboard will show that LTR is now enabled:



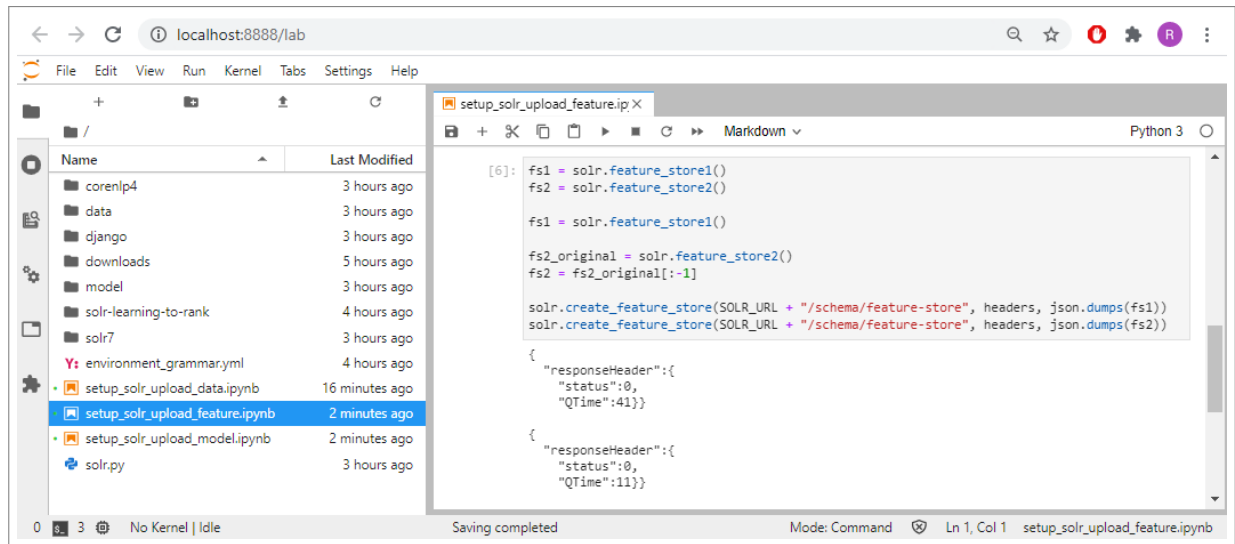
In the list of *query parser plugins* for *core1*, we will see the LTR plugin:

The screenshot shows the Solr Admin interface in a web browser. The address bar indicates the URL is `localhost:8983/solr/#/core1/plugins?type=queryparser`. The left sidebar contains navigation links such as Dashboard, Logging, Core Admin, Java Properties, Thread Dump, and a dropdown menu for 'core1' with options like Overview, Analysis, Dataimport, Documents, Files, Ping, Plugins / Stats, Query, Replication, Schema, and Segments info. The main content area is divided into two columns. The left column lists categories: ADMIN, CORE, QUERY, UPDATE, CACHE, HIGHLIGHTER, QUERYPARSER (selected), SPELLCHECKER, REPLICATION, and OTHER. Below these is a note: "NOTE: Only selected metrics are shown here. Full metrics can be accessed via /admin/metrics handler." The right column displays a list of query parser plugins, each with a checkbox and a class name. The plugin `org.apache.solr.search.LTRQParserPlugin` is highlighted with a red rectangular box. Other visible plugins include `org.apache.solr.search.BoostQParserPlugin`, `org.apache.solr.search.GraphTermsQParserPlugin`, `org.apache.solr.search.ComplexPhraseQParserPlugin`, `org.apache.solr.search.PayloadScoreQParserPlugin`, `org.apache.solr.search.HashQParserPlugin`, `org.apache.solr.search.CollapsingQParserPlugin`, `org.apache.solr.search.TermQParserPlugin`, `org.apache.solr.search.ExportQParserPlugin`, `org.apache.solr.search.SpatialFilterQParserPlugin`, `org.apache.solr.search.IGainTermsQParserPlugin`, `org.apache.solr.search.join.FiltersQParserPlugin`, `org.apache.solr.search.ExtendedDismaxQParserPlugin`, `org.apache.solr.search.join.BlockJoinChildQParserPlugin`, `org.apache.solr.search.LuceneQParserPlugin`, `org.apache.solr.search.FunctionQParserPlugin`, `org.apache.solr.search.DisMaxQParserPlugin`, `org.apache.solr.search.XmlQParserPlugin`, `org.apache.solr.search.JoinQParserPlugin`, `org.apache.solr.search.SimpleQParserPlugin`, `org.apache.solr.search.SignificantTermsQParserPlugin`, `org.apache.solr.search.SwitchQParserPlugin`, `org.apache.solr.search.TextLogisticRegressionQParserPlugin`, `org.apache.solr.search.join.BlockJoinParentQParserPlugin`, `org.apache.solr.search.MaxScoreQParserPlugin`, `org.apache.solr.search.join.GraphQParserPlugin`, `org.apache.solr.search.PrefixQParserPlugin`, `org.apache.solr.search.NestedQParserPlugin`, `org.apache.solr.search.SpatialBoxQParserPlugin`, `org.apache.solr.search.FieldQParserPlugin`, `org.apache.solr.search.TermsQParserPlugin`, `org.apache.solr.search.RawQParserPlugin`, `org.apache.solr.search.ReRankQParserPlugin`, `org.apache.solr.search.PayloadCheckQParserPlugin`, `org.apache.solr.search.mlt.MLTQParserPlugin`, `org.apache.solr.search.SurroundQParserPlugin`, `org.apache.solr.search.FunctionRangeQParserPlugin`, and `org.apache.solr.search.BoolQParserPlugin`.

7.2 Feature Store and Model Store

7.2.1 Uploading Features

First, we upload the feature definitions for all features to Solr. To upload feature definitions to Solr, open `setup_solr_upload_feature.ipynb` and run it:



```
[6]: fs1 = solr.feature_store1()
fs2 = solr.feature_store2()

fs1 = solr.feature_store1()

fs2_original = solr.feature_store2()
fs2 = fs2_original[:-1]

solr.create_feature_store(SOLR_URL + "/schema/feature-store", headers, json.dumps(fs1))
solr.create_feature_store(SOLR_URL + "/schema/feature-store", headers, json.dumps(fs2))

{
  "responseHeader": {
    "status": 0,
    "QTime": 41
  }
}

{
  "responseHeader": {
    "status": 0,
    "QTime": 11
  }
}
```

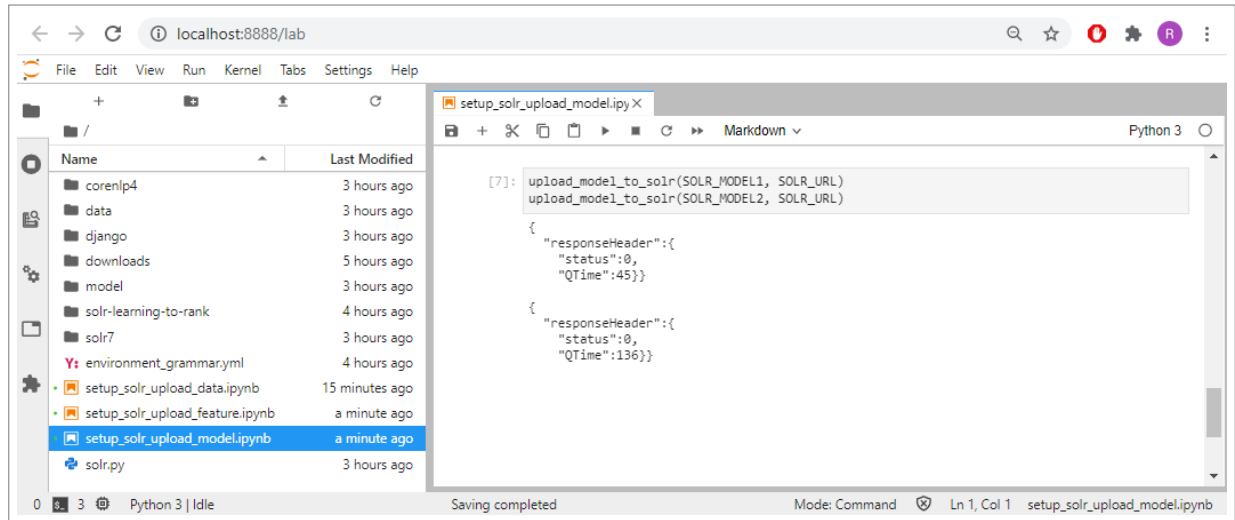
Solr will create the JSON file `_schema_feature-store.json` to store the feature definitions.

7.2.2 Uploading Models

Next, we upload the trained models to Solr. Copy the following models from the repository directory:

- `/grammar/model/model11.json`
- `/grammar/model/model12.json`

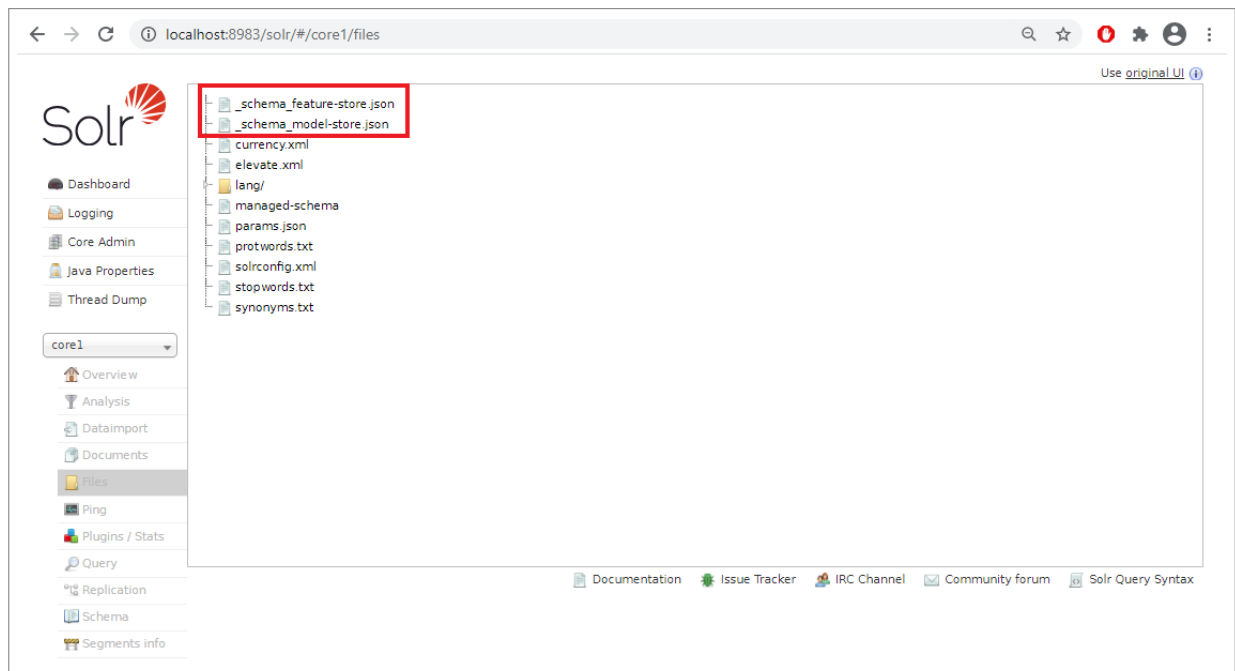
We upload Model 1 and Model 2 in JSON format to Solr. Open `setup_solr_upload_model.ipynb` and run it:



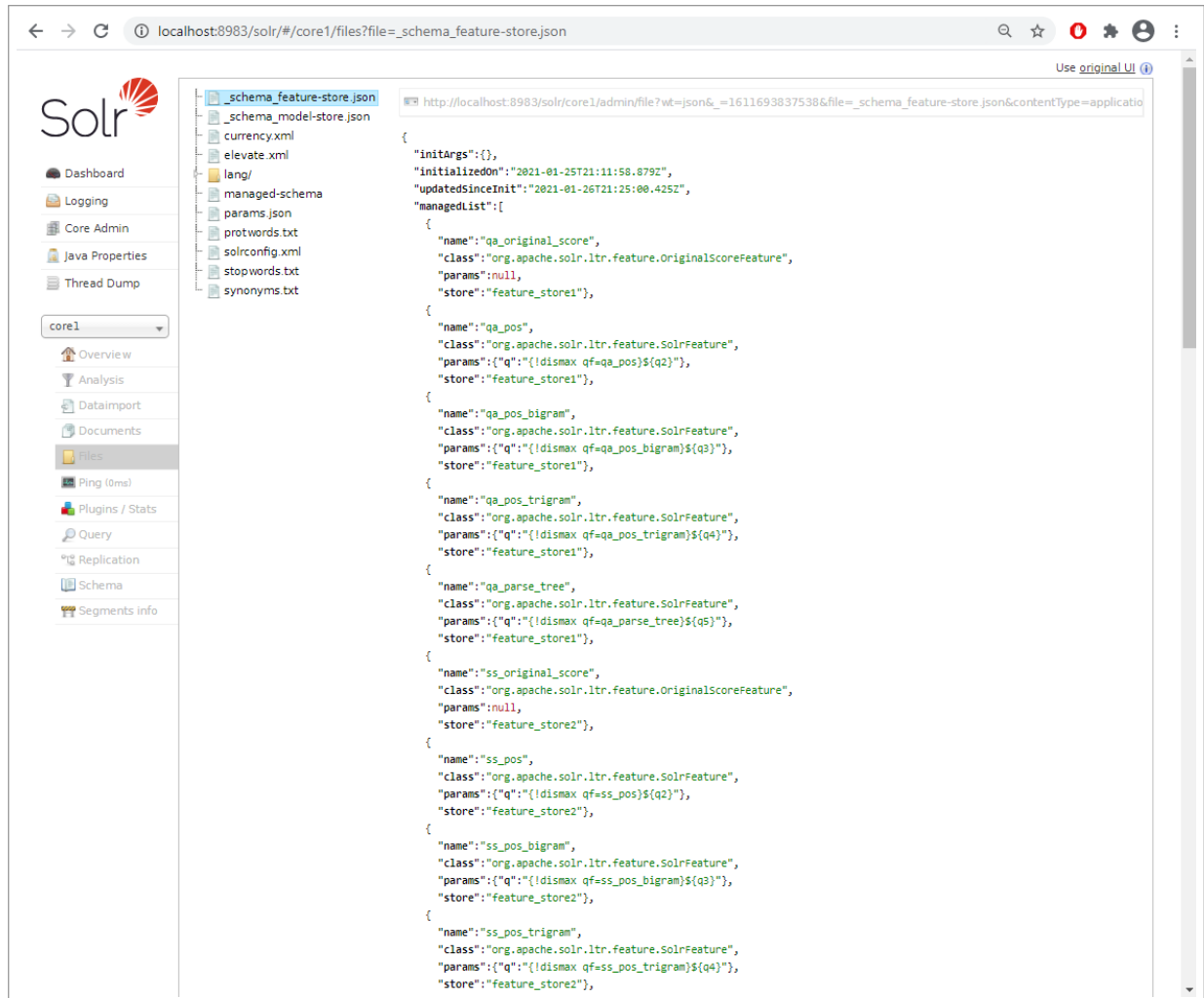
Solr will create the JSON file `_schema_model-store.json` to store the information uploaded.

7.2.3 Solr Admin UI

In Solr Admin UI, go to the **Files** interface. We will see the two JSON files created by Solr:



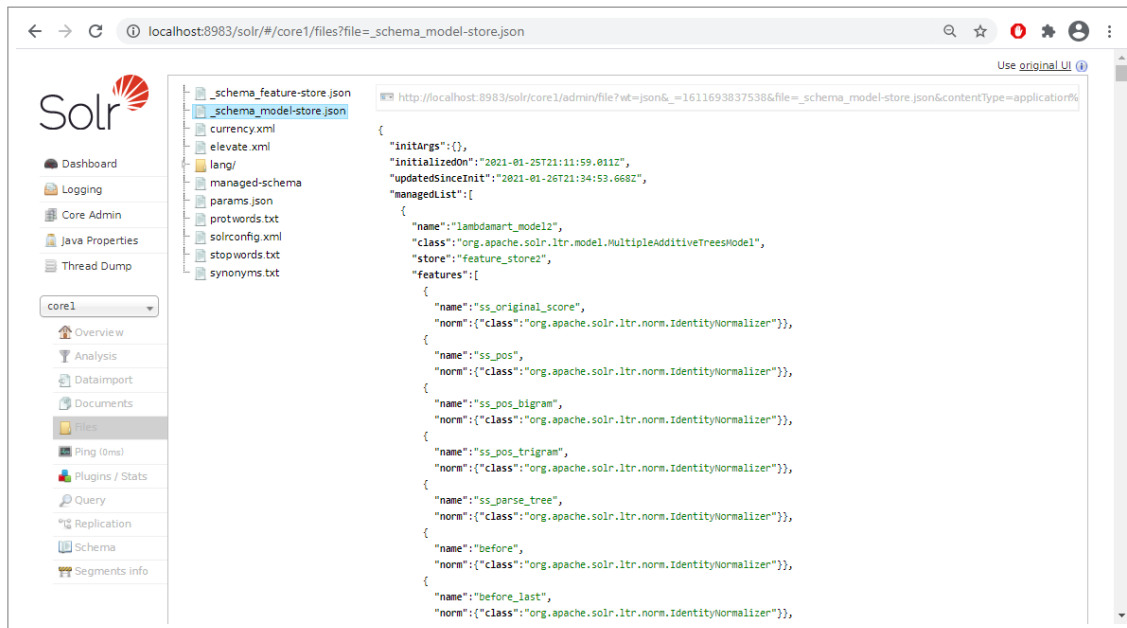
Click on each file to see its contents. In `_schema_feature-store.json`, the variable `managedList` is a list of feature definitions. Each feature is stored in either `feature_store1` (Model 1 features) or `feature_store2` (Model 2 features):



The screenshot displays the Solr Admin UI in a web browser. The address bar shows the URL: `localhost:8983/solr/#/core1/files?file=_schema_feature-store.json`. The left sidebar contains the Solr logo and a navigation menu with options: Dashboard, Logging, Core Admin, Java Properties, Thread Dump, Overview, Analysis, Dataimport, Documents, Files (selected), Ping (0ms), Plugins / Stats, Query, Replication, Schema, and Segments Info. The main content area shows a file tree on the left with `_schema_feature-store.json` selected. The right pane displays the JSON content of this file, which defines a list of features for two different models.

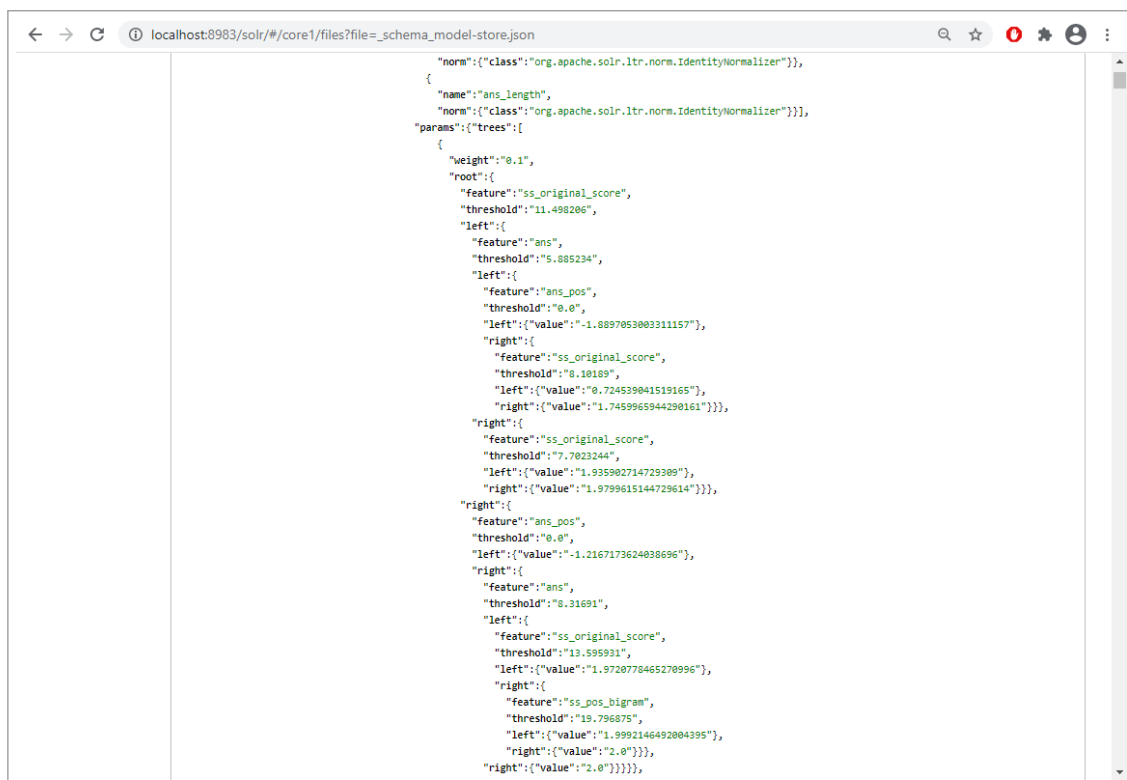
```
{
  "initargs": {},
  "initializedOn": "2021-01-25T21:11:58.879Z",
  "updatedSinceInit": "2021-01-26T21:25:00.425Z",
  "managedList": [
    {
      "name": "qa_original_score",
      "class": "org.apache.solr.ltr.feature.OriginalScoreFeature",
      "params": null,
      "store": "feature_store1",
    },
    {
      "name": "qa_pos",
      "class": "org.apache.solr.ltr.feature.SolrFeature",
      "params": {
        "q": "{!dismax qf=qa_pos}${q2}"
      },
      "store": "feature_store1",
    },
    {
      "name": "qa_pos_bigram",
      "class": "org.apache.solr.ltr.feature.SolrFeature",
      "params": {
        "q": "{!dismax qf=qa_pos_bigram}${q3}"
      },
      "store": "feature_store1",
    },
    {
      "name": "qa_pos_trigram",
      "class": "org.apache.solr.ltr.feature.SolrFeature",
      "params": {
        "q": "{!dismax qf=qa_pos_trigram}${q4}"
      },
      "store": "feature_store1",
    },
    {
      "name": "qa_parse_tree",
      "class": "org.apache.solr.ltr.feature.SolrFeature",
      "params": {
        "q": "{!dismax qf=qa_parse_tree}${q5}"
      },
      "store": "feature_store1",
    },
    {
      "name": "ss_original_score",
      "class": "org.apache.solr.ltr.feature.OriginalScoreFeature",
      "params": null,
      "store": "feature_store2",
    },
    {
      "name": "ss_pos",
      "class": "org.apache.solr.ltr.feature.SolrFeature",
      "params": {
        "q": "{!dismax qf=ss_pos}${q2}"
      },
      "store": "feature_store2",
    },
    {
      "name": "ss_pos_bigram",
      "class": "org.apache.solr.ltr.feature.SolrFeature",
      "params": {
        "q": "{!dismax qf=ss_pos_bigram}${q3}"
      },
      "store": "feature_store2",
    },
    {
      "name": "ss_pos_trigram",
      "class": "org.apache.solr.ltr.feature.SolrFeature",
      "params": {
        "q": "{!dismax qf=ss_pos_trigram}${q4}"
      },
      "store": "feature_store2",
    }
  ]
}
```

In `_schema_model-store.json`, the variable `managedList` is a list of models. The screenshot below shows the information stored for Model 2: the model name (`lambdamart_model2`), the model class, the feature store name, the list of features, and the LambdaMART model.



The screenshot shows the Solr Admin UI at `localhost:8983/solr/#/core1/files?file=_schema_model-store.json`. The left sidebar shows the file tree with `_schema_model-store.json` selected. The main content area displays the JSON content of the file:

```
{
  "initArgs": {},
  "initializedOn": "2021-01-25T21:11:59.011Z",
  "updatedSinceInit": "2021-01-26T21:34:53.668Z",
  "managedList": [
    {
      "name": "lambdamart_model2",
      "class": "org.apache.solr.ltr.model.MultipleAdditiveTreesModel",
      "store": "feature_store2",
      "features": [
        {
          "name": "ss_original_score",
          "norm": {
            "class": "org.apache.solr.ltr.norm.IdentityNormalizer"
          }
        },
        {
          "name": "ss_pos",
          "norm": {
            "class": "org.apache.solr.ltr.norm.IdentityNormalizer"
          }
        },
        {
          "name": "ss_pos_bigram",
          "norm": {
            "class": "org.apache.solr.ltr.norm.IdentityNormalizer"
          }
        },
        {
          "name": "ss_pos_trigram",
          "norm": {
            "class": "org.apache.solr.ltr.norm.IdentityNormalizer"
          }
        },
        {
          "name": "ss_parse_tree",
          "norm": {
            "class": "org.apache.solr.ltr.norm.IdentityNormalizer"
          }
        },
        {
          "name": "before",
          "norm": {
            "class": "org.apache.solr.ltr.norm.IdentityNormalizer"
          }
        },
        {
          "name": "before_last",
          "norm": {
            "class": "org.apache.solr.ltr.norm.IdentityNormalizer"
          }
        }
      ]
    }
  ]
}
```



The screenshot shows the continuation of the JSON content from the previous screenshot, specifically the `params` section of the `lambdamart_model2` model:

```
    {
      "norm": {
        "class": "org.apache.solr.ltr.norm.IdentityNormalizer"
      },
      {
        "name": "ans_length",
        "norm": {
          "class": "org.apache.solr.ltr.norm.IdentityNormalizer"
        }
      },
      "params": {
        "trees": [
          {
            "weight": "0.1",
            "root": {
              "feature": "ss_original_score",
              "threshold": "11.498286",
              "left": {
                "feature": "ans",
                "threshold": "5.885234",
                "left": {
                  "feature": "ans_pos",
                  "threshold": "0.0",
                  "left": {
                    "value": "-1.889705300311157",
                    "right": {
                      "feature": "ss_original_score",
                      "threshold": "0.10189",
                      "left": {
                        "value": "0.724539041519165",
                        "right": {
                          "value": "1.7459965944290161"
                        }
                      }
                    }
                  }
                }
              }
            }
          }
        ]
      }
    }
  ]
}
```

7.3 Stanford CoreNLP

The last thing that we need to set up before we can do intelligent Solr search is the Stanford CoreNLP server. CoreNLP is used to extract grammar production rules from sentences. Download CoreNLP (`stanford-corenlp-latest-4.2.0.zip`) from:

<https://stanfordnlp.github.io/CoreNLP/>

The ZIP file contains one directory. Move it to the project directory and rename it `corenlp4`.

To start the server, the basic command is:

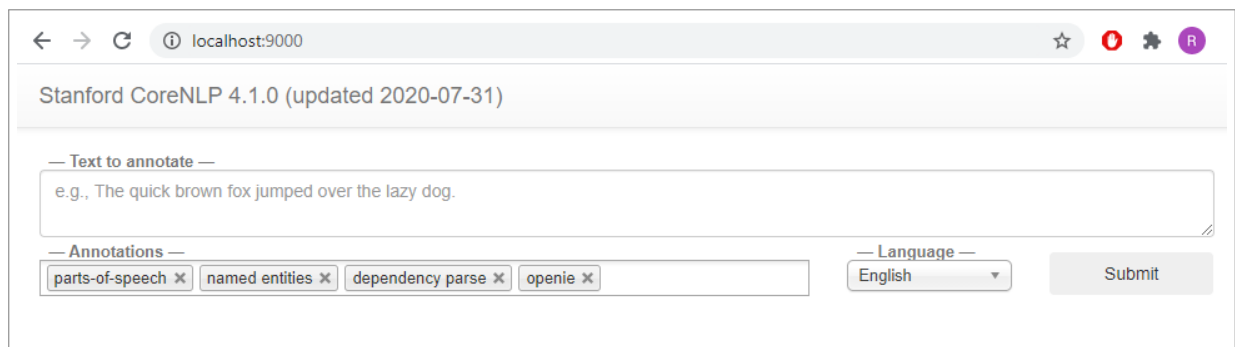
```
PS D:\grammar\corenlp4> java -mx4g -cp "*" edu.stanford.nlp.pipeline.StanfordCoreNLPServer
-timout 15000
[main] INFO CoreNLP - --- StanfordCoreNLPServer#main() called ---
[main] INFO CoreNLP - Server default properties:
                        (Note: unspecified annotator properties are English defaults)
                        inputFormat = text
                        outputFormat = json
                        prettyPrint = false
[main] INFO CoreNLP - Threads: 12
[main] INFO CoreNLP - Starting server...
[main] INFO CoreNLP - StanfordCoreNLPServer listening at /0:0:0:0:0:0:0:0:9000
```

An equivalent command, in which all *annotators* are specified, is:

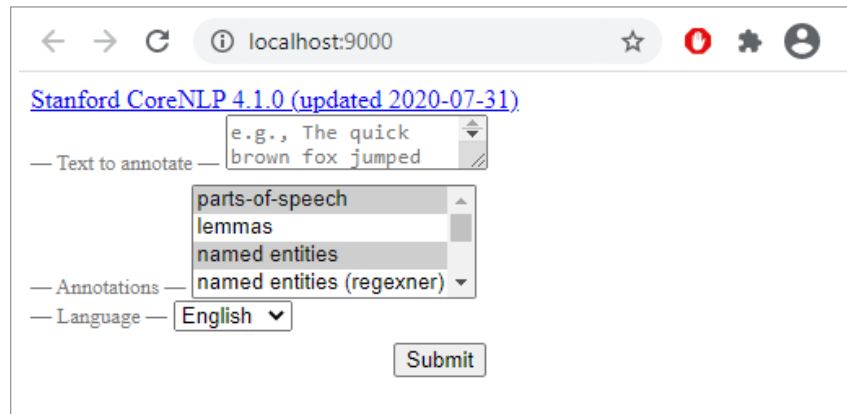
```
PS D:\grammar\corenlp4> java -mx4g -cp "*"
edu.stanford.nlp.pipeline.StanfordCoreNLPServer -preload
tokenize,ssplit,pos,lemma,ner,parse,depparse
-port 9000 -timeout 15000
```

The wildcard "*" (it must be in quotes) after `-cp` loads all JAR files in the current directory.

If the server is running, the CoreNLP page will be loaded at <http://localhost:9000>:



If you are not connected to the internet, the CoreNLP page will not be styled:



The method to shut down the server is described in CoreNLP's documentation:

<https://stanfordnlp.github.io/CoreNLP/corenlp-server.html#stopping-the-server>

To shut down the server, we pass the shutdown key in the command `wget`. In Windows 10, the key can be obtained as follows:

```
PS C:\> cat /Users/ronko/AppData/Local/Temp/corenlp.shutdown
18fbitvsk05312p26tt04tdd47
```

The command to shut down the server is:

```
wget "localhost:9000/shutdown?key=18fbitvsk05312p26tt04tdd47" -O -
```

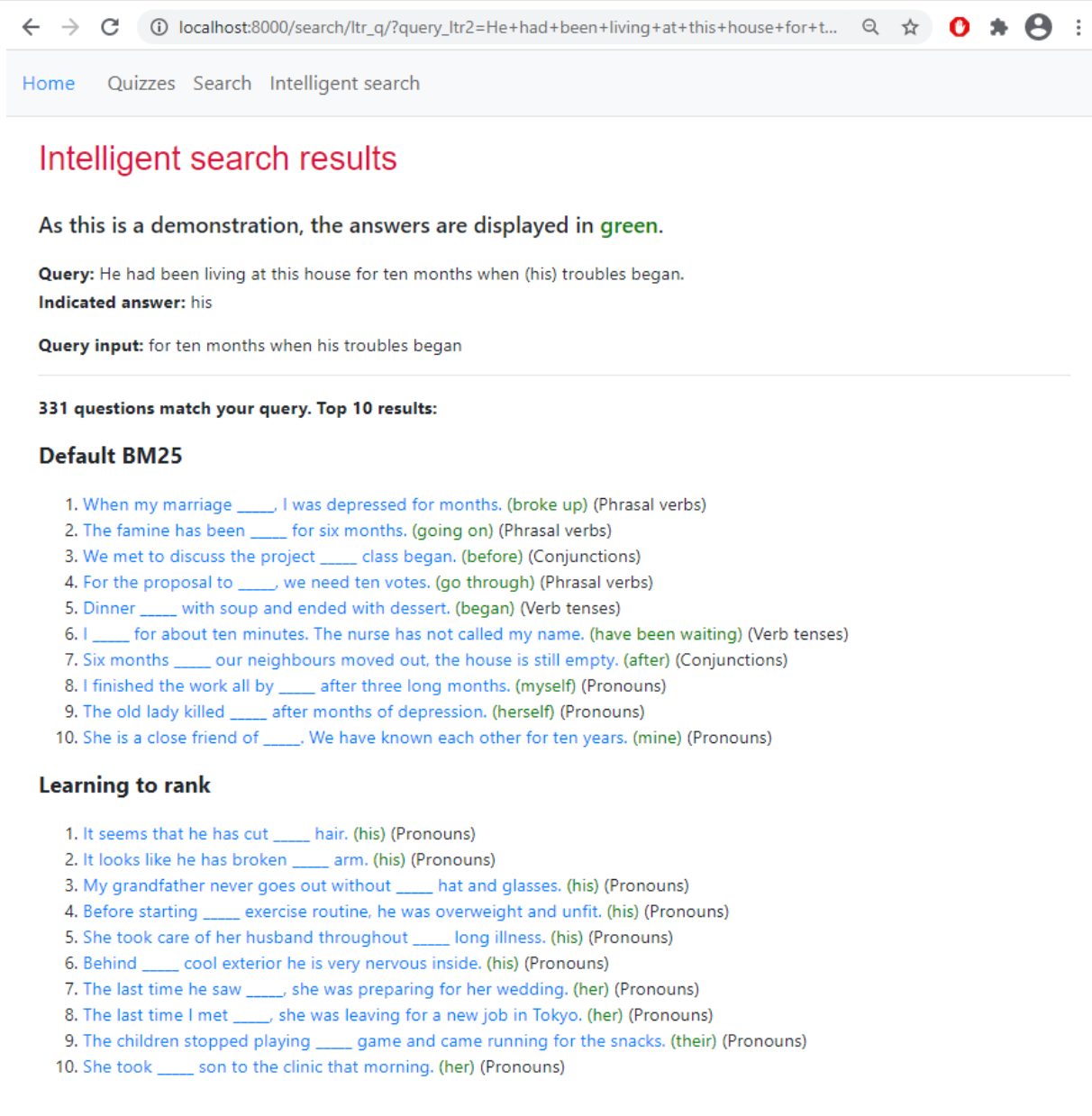
However, we can easily shut down the server by closing the command line interface in which we started the server.

7.4 Intelligent Search

7.4.1 Intelligent Search Page

Now that everything is set up, we can send queries on the website:

- Sample queries



The screenshot shows a web browser window with the address bar displaying `localhost:8000/search/ltr_q/?query_ltr2=He+had+been+living+at+this+house+for+t...`. The browser's navigation bar includes links for Home, Quizzes, Search, and Intelligent search. The main content area is titled "Intelligent search results" in red. Below the title, a message states: "As this is a demonstration, the answers are displayed in green." The query is shown as "Query: He had been living at this house for ten months when (his) troubles began." and the indicated answer is "Indicated answer: his". The query input is "for ten months when his troubles began". A horizontal line separates the query section from the results. The results section is titled "331 questions match your query. Top 10 results:" and "Default BM25". It lists 10 search results, each with a numbered sentence, a blank space for the answer, and the correct answer in green parentheses. The results are as follows:

1. When my marriage ____, I was depressed for months. (broke up) (Phrasal verbs)
2. The famine has been ____ for six months. (going on) (Phrasal verbs)
3. We met to discuss the project ____ class began. (before) (Conjunctions)
4. For the proposal to ____, we need ten votes. (go through) (Phrasal verbs)
5. Dinner ____ with soup and ended with dessert. (began) (Verb tenses)
6. I ____ for about ten minutes. The nurse has not called my name. (have been waiting) (Verb tenses)
7. Six months ____ our neighbours moved out, the house is still empty. (after) (Conjunctions)
8. I finished the work all by ____ after three long months. (myself) (Pronouns)
9. The old lady killed ____ after months of depression. (herself) (Pronouns)
10. She is a close friend of ____. We have known each other for ten years. (mine) (Pronouns)

Below the results is a section titled "Learning to rank" which lists 10 more search results, each with a numbered sentence, a blank space for the answer, and the correct answer in green parentheses. The results are as follows:

1. It seems that he has cut ____ hair. (his) (Pronouns)
2. It looks like he has broken ____ arm. (his) (Pronouns)
3. My grandfather never goes out without ____ hat and glasses. (his) (Pronouns)
4. Before starting ____ exercise routine, he was overweight and unfit. (his) (Pronouns)
5. She took care of her husband throughout ____ long illness. (his) (Pronouns)
6. Behind ____ cool exterior he is very nervous inside. (his) (Pronouns)
7. The last time he saw ____, she was preparing for her wedding. (her) (Pronouns)
8. The last time I met ____, she was leaving for a new job in Tokyo. (her) (Pronouns)
9. The children stopped playing ____ game and came running for the snacks. (their) (Pronouns)
10. She took ____ son to the clinic that morning. (her) (Pronouns)

- Answer indicated

localhost:8000/search/ltr_q/?csrfmiddlewaretoken=GbCt8AZPaVJM7nytZGq7ZEgLI8A...

Home

Quizzes

Search

Intelligent search

Intelligent search results

As this is a demonstration, the answers are displayed in **green**.

Query: You cannot proceed with the work (until) you get a permit.
Indicated answer: until

Query input: proceed with the work until you get a permit

754 questions match your query. Top 10 results:

Default BM25

1. I could ____ with a small budget last time but times have changed. (get by) (Phrasal verbs)
2. Please stop work and get ____ bed. (into) (Prepositions)
3. I was up ____ three in the morning doing my work. (until) (Conjunctions)
4. The company can ____ with two employees at the moment. (get by) (Phrasal verbs)
5. The ceremony will proceed ____ my late father's wishes. (according to) (Prepositions)
6. I need to talk to you ____ you are done with your work. (after) (Conjunctions)
7. ____ she starts her work, she will not stop until she completes it. (Once) (Conjunctions)
8. You will never know ____ you try. (until) (Conjunctions)
9. They would not back down until they get what is rightfully _____. (theirs) (Pronouns)
10. I don't ____ with most of my classmates. (get along) (Phrasal verbs)

Learning to rank

1. You will never know ____ you try. (until) (Conjunctions)
2. We will never find out the truth ____ we analyse the evidence. (until) (Conjunctions)
3. Once she starts her work, she will not stop ____ she completes it. (until) (Conjunctions)
4. I was up ____ three in the morning doing my work. (until) (Conjunctions)
5. The shop will remain open ____ all the customers have left. (until) (Conjunctions)
6. I will keep on dreaming ____ my dreams come true. (until) (Conjunctions)
7. I asked her ____ she was keen to play the game. (whether) (Conjunctions)
8. She listened to the song ____ she was having dinner. (while) (Conjunctions)
9. She is concerned ____ the state of her country. (about) (Prepositions)
10. Look for me ____ the bookshop after your meeting. (at) (Prepositions)

- Answer not indicated

[←](#)
[→](#)
[↻](#)

localhost:8000/search/ltr_q/?csrfmiddlewaretoken=GbCt8AZPaVJM7nytZGq7ZEgLi8A...

[Home](#)
[Quizzes](#)
[Search](#)
[Intelligent search](#)

Intelligent search results

As this is a demonstration, the answers are displayed in **green**.

Query: You cannot proceed with the work until you get a permit.

757 questions match your query. Top 10 results:

Default BM25

1. I need to talk to you ____ you are done with your work. (after) (Conjunctions)
2. You will never know ____ you try. (until) (Conjunctions)
3. You cannot go for the mountain hike ____ I come along. (unless) (Conjunctions)
4. You cannot go to the party unless I _____. (come along) (Phrasal verbs)
5. You must ____ this setback and start working hard again. (get over) (Phrasal verbs)
6. If you really ____ it, you will find that this is the best offer you will get. (look into) (Phrasal verbs)
7. ____ you brush your teeth every day you will not get cavities so easily. (If) (Conjunctions)
8. If you ____ too much sweet foods, you may get tooth decay. (eat) (Verb tenses)
9. ____ I have found the hotel, I will give you a call. (Once) (Conjunctions)
10. I could ____ with a small budget last time but times have changed. (get by) (Phrasal verbs)

Learning to rank

1. I need to talk to you ____ you are done with your work. (after) (Conjunctions)
2. You will never know ____ you try. (until) (Conjunctions)
3. You cannot go for the mountain hike ____ I come along. (unless) (Conjunctions)
4. You cannot go to the party unless I _____. (come along) (Phrasal verbs)
5. You must ____ this setback and start working hard again. (get over) (Phrasal verbs)
6. I felt sorry ____ my colleagues who lost their jobs. (for) (Prepositions)
7. The Olympics team is aiming ____ a gold medal next year. (for) (Prepositions)
8. He shouted ____ help when he was stuck in the lift. (for) (Prepositions)
9. Sixty per cent the population voted ____ the people's party. (for) (Prepositions)
10. I do not know the reason ____ his unhappiness. (for) (Prepositions)

7.4.2 Quizzes Page

Test the quiz feature **More questions like this:**

localhost:8000/search/recommend/Someone%20called%20the%20police%20to%20in...

Home

Quizzes

Search

Intelligent search

Top 10 similar questions

As this is a demonstration, the answers are displayed in green.

Question: Someone called the police to investigate. But she was dead by the time ____ arrived.

Answer: they

Query input: dead by the time they arrived

589 questions match the query. Top 10 results:

Default BM25

1. A medical team was called to his home, but he was unconscious by the time ____ arrived. (they) (Pronouns)
2. By the time they found ____ pet, it had died. (their) (Pronouns)
3. By the time they found their pet, it ____, (had died) (Verb tenses)
4. ____ the time he started working hard, it was too late. (By) (Prepositions)
5. They were sitting ____ the river looking at the ducks. (by) (Prepositions)
6. The flowers ____ yet. The delivery is late. (have not arrived) (Verb tenses)
7. They spent the entire day at the beach by _____. (themselves) (Pronouns)
8. They painted the whole house all by _____. (themselves) (Pronouns)
9. We asked the counter staff ____ the train has arrived. (whether) (Conjunctions)
10. I did not want to join them, so they went to the show by _____. (themselves) (Pronouns)

Learning to rank

1. A medical team was called to his home, but he was unconscious by the time ____ arrived. (they) (Pronouns)
2. People tend to take their health for granted until ____ fall sick. (they) (Pronouns)
3. My colleague and ____ attended the seminar last week. (I) (Pronouns)
4. There was a good coach in the team who helped ____ become better players. (us) (Pronouns)
5. All the toys in this room belonged to ____ and my twin brother. (me) (Pronouns)
6. She saw ____ at the meeting but he did not seem to remember her. (him) (Pronouns)
7. ____ made it clear to everyone in the room that this will be his last offer. (He) (Pronouns)
8. ____ made it clear to all of us in the meeting that this will be her final offer. (She) (Pronouns)
9. My friend and ____ have decided to sign up for the course. (I) (Pronouns)
10. My sister and ____ are the best of friends. (I) (Pronouns)

Chapter 8

Summary of Setup Process

We have completed the setup using the raw datasets, feature datasets, model datasets, and the trained models. The following is a step-by-step summary of the setup process described in the preceding six chapters:

1. Project Requirements and Virtual Environment

- Download Solr, CoreNLP, RankLib, and Apache Commons Mathematics.
- Download and install Java Runtime Environment and DB Browser.
- Download and install Miniconda.
- Set up the virtual environment.
- Install the required libraries in the virtual environment.
- Create the project file system.
- Download the project repository from GitHub.

2. Django Setup on Development Server

- Set up the default Django website (<http://localhost:8000>).
- Set up the Django admin site (<http://localhost:8000/admin>).
- Create the directories for templates and static files.

3. Grammar Practice Website

- Add the three applications `apppage`, `appquiz`, and `appsearch` to `settings.py`.
- Add the application `crispy_forms` to `settings.py`.
- Add the paths for the three apps to `urls.py`.
- Copy the source files (templates, CSS) and directories for the three applications from the repository directory.
- Run the development server and check that the grammar website home page loads (<http://localhost:8000/grammar>).

4. Quizzes and Database Search

- Copy the raw data directory from the repository directory.
- Create the database tables for `appquiz` and `appsearch`.
- Create data (for topics and quiz numbers) in database tables using the Django admin site.
- Import data (for quiz questions and question bank questions) from CSV files (`rawdata_quiz.csv`, `rawdata_doc.csv`, `rawdata_query.csv`) to database tables using **DB Browser**.
- Check that the quiz and basic search features work on the website.

5. Solr Search

- Set up a Solr node (<http://localhost:8983>).
- Create a core in the Solr node.
- Upload features data to Solr from CSV files (`feature_doc.csv`, `feature_query.csv`).
- Check that Solr search works on the website.

6. Intelligent Solr Search

- Add LTR plugin configuration information to `solrconfig.xml`.
- Upload the feature definitions to Solr.
- Upload the two LambdaMART models to Solr.
- Run Stanford CoreNLP (<http://localhost:9000>) using the command:

```
java -mx4g -cp "*" edu.stanford.nlp.pipeline.StanfordCoreNLPServer -timeout 15000
```
- Check that intelligent Solr search works and the **More questions like this** feature works in quizzes.

Chapter 9

Creating Model Datasets

I will now describe the process of creating the model datasets (training, validation, testing) from the raw data (in `/grammar/data/raw/`), and the building of the LambdaMART models. Copy the following the Jupyter Notebooks from the repository directory and run them in the following order:

- `/grammar/1_extract_feature.ipynb`

This notebook extracts features from the raw datasets in `/grammar/data/raw/` and creates the feature datasets in `/grammar/data/feature`.

- `/grammar/setup_solr_upload_data.ipynb`

Upload the features data in `feature_doc.csv` and `feature_query.csv` to Solr. This is the same as what we did in Section [6.5.2](#).

- `/grammar/2_solr_upload_feature.ipynb`

This notebook uploads the feature definitions to Solr. This is slightly different from what we did in Section [7.2.1](#), which excluded the feature `qb_topic_id`. `qb_topic_id` is only used for creating the datasets; it is not a feature in the ranking models.

- `/grammar/3_solr_upload_linear_model.ipynb`

This notebook uploads two *linear models* (one for each ranking model) used by Solr to extract feature values (which are the weights in each linear model) for all the features.

- `/grammar/4_create_model_dataset.ipynb`

This notebook creates the training, validation and testing datasets for Model 1 and Model 2.

A sample of the Model 1 dataset `model1_train.txt` is shown below:

```
1 qid:851 1:19.991014 2:4.0605597 3:11.197081 4:16.45244 5:8.824474 # docid:851
1 qid:851 1:10.315361 2:3.538503 3:6.289652 4:4.206999 5:4.2085648 # docid:554
1 qid:851 1:9.662352 2:3.5659604 3:5.561839 4:4.520902 5:4.8901253 # docid:236
```

A sample of the Model 2 dataset `model2_train.txt` is shown below:

```
3 qid:851 1:19.452402 2:4.8407536 3:11.618433 4:15.950744 5:9.296059 6:11.506929
7:1.0 8:1.0 9:5.9961243 10:8.067158 11:5.6357236 12:9.641943 13:6.9352627 14:1.0
15:1.0 16:1.4760072 17:1.5126191 18:0.0 19:1.9578518 20:4.6139307 21:1.0 22:1.0
23:1.0 24:1.0 25:1.0 26:1.0 27:1.0 28:1.0 # docid:851
0 qid:851 1:10.547468 2:4.3035083 3:6.9204445 4:4.260151 5:5.3723674 6:2.2685385
7:0.0 8:0.0 9:0.6976033 10:0.0 11:0.0 12:0.6431802 13:5.4170594 14:1.0
15:1.0 16:1.1525129 17:1.0870409 18:0.0 19:1.3429569 20:0.0 21:0.0 22:0.0
23:0.0 24:0.0 25:0.0 26:0.0 27:0.0 28:0.0 # docid:554
0 qid:851 1:9.775789 2:4.3051305 3:4.950946 4:0.0 5:4.6064773 6:1.4456632
7:0.0 8:0.0 9:2.5833564 10:2.4382665 11:0.0 12:1.4965961 13:0.0 14:0.0
15:0.0 16:0.0 17:0.0 18:0.0 19:0.0 20:0.0 21:0.0 22:0.0
23:1.0 24:1.0 25:1.0 26:0.0 27:0.0 28:1.0 # docid:236
```

- `/grammar/5_create_baseline_dataset.ipynb`

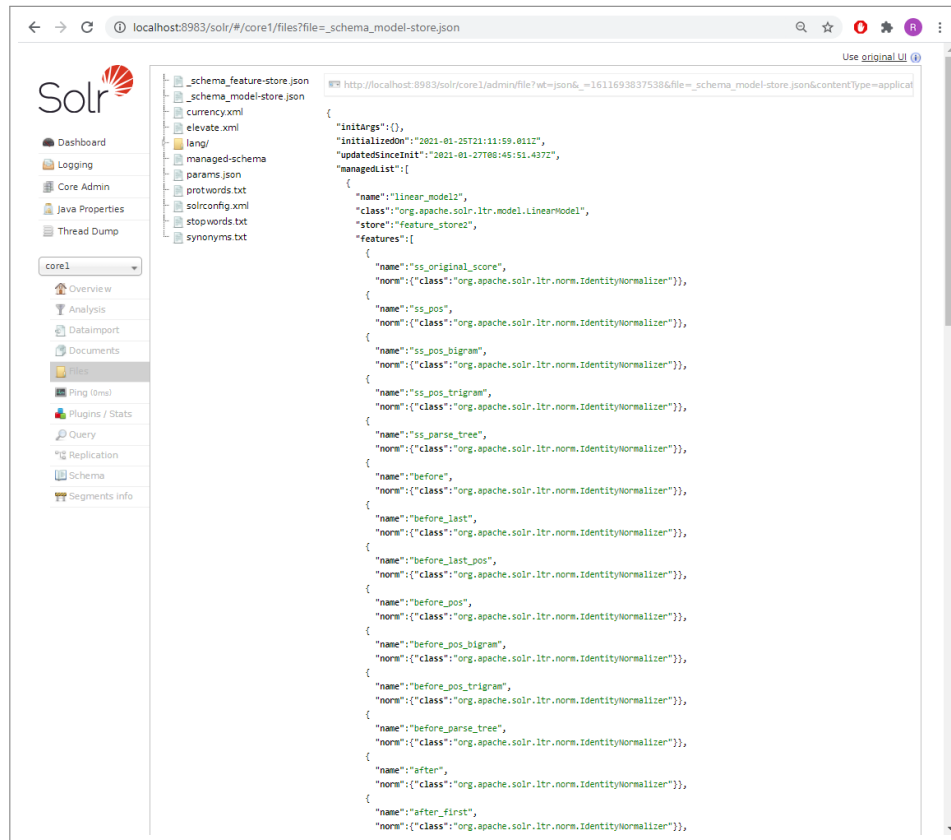
This notebook creates the training, validation and testing datasets for Baseline Model 1 and Baseline Model 2. A sample of the Baseline Model 1 dataset `baseline_model1_train.txt` is shown below:

```
1 qid:851 1:19.991014 # docid:851
1 qid:851 1:10.315361 # docid:554
1 qid:851 1:9.662352 # docid:236
```

A sample of the Baseline Model 2 dataset `baseline_model2_train.txt` is shown below:

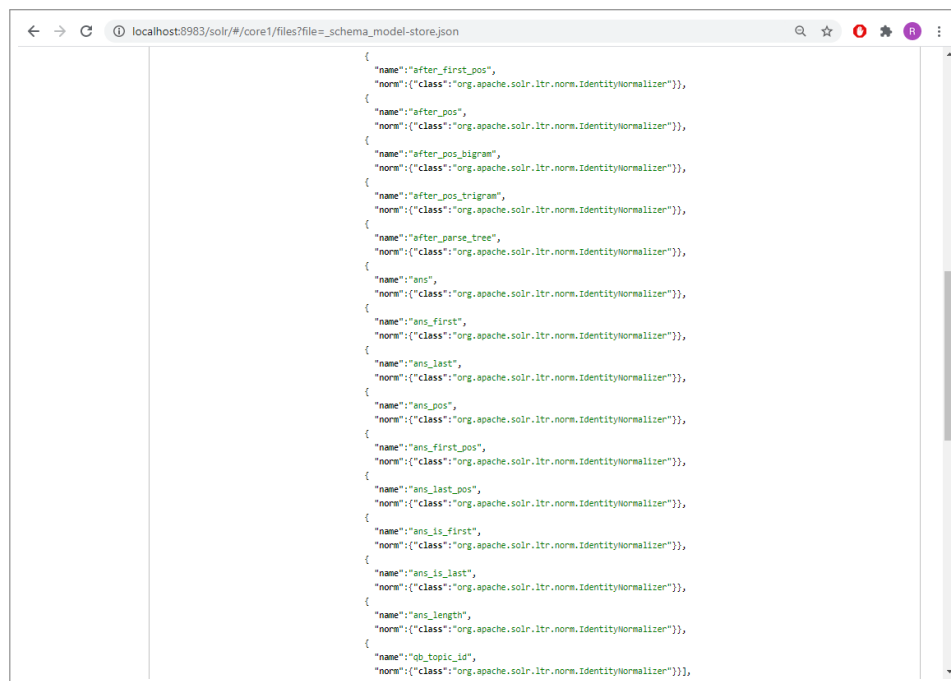
```
3 qid:851 1:19.452402 # docid:851
0 qid:851 1:10.547468 # docid:554
0 qid:851 1:9.775789 # docid:236
```

Linear model for Model 2:



The screenshot shows the Solr Admin UI for 'core1'. The left sidebar contains navigation links: Dashboard, Logging, Core Admin, Java Properties, Thread Dump, Overview, Analysis, Dataimport, Documents, Files, Ping (lms), Plugins / Stats, Query, Replication, Schema, and Segments info. The 'Files' section is selected, showing a file tree with: _schema_feature-store.json, _schema_model-store.json, currency.xml, elevate.xml, lang/, managed-schema, params.json, protwords.txt, solrconfig.xml, stopwords.txt, and synonyms.txt. The main content area displays the JSON configuration for '_schema_model-store.json'.

```
{
  "initArgs": {},
  "initializedOn": "2021-01-25T21:11:59.011Z",
  "updatedSinceInit": "2021-01-27T08:45:51.437Z",
  "managedList": [
    {
      "name": "linear_model1",
      "class": "org.apache.solr.ltr.model.linearModel",
      "store": "feature_store2",
      "features": [
        {
          "name": "ss_original_score",
          "norm": {
            "class": "org.apache.solr.ltr.norm.IdentityNormalizer"
          }
        },
        {
          "name": "ss_pos",
          "norm": {
            "class": "org.apache.solr.ltr.norm.IdentityNormalizer"
          }
        },
        {
          "name": "ss_pos_bigram",
          "norm": {
            "class": "org.apache.solr.ltr.norm.IdentityNormalizer"
          }
        },
        {
          "name": "ss_pos_trigram",
          "norm": {
            "class": "org.apache.solr.ltr.norm.IdentityNormalizer"
          }
        },
        {
          "name": "ss_parse_tree",
          "norm": {
            "class": "org.apache.solr.ltr.norm.IdentityNormalizer"
          }
        },
        {
          "name": "before",
          "norm": {
            "class": "org.apache.solr.ltr.norm.IdentityNormalizer"
          }
        },
        {
          "name": "before_last",
          "norm": {
            "class": "org.apache.solr.ltr.norm.IdentityNormalizer"
          }
        },
        {
          "name": "before_last_pos",
          "norm": {
            "class": "org.apache.solr.ltr.norm.IdentityNormalizer"
          }
        },
        {
          "name": "before_pos",
          "norm": {
            "class": "org.apache.solr.ltr.norm.IdentityNormalizer"
          }
        },
        {
          "name": "before_pos_bigram",
          "norm": {
            "class": "org.apache.solr.ltr.norm.IdentityNormalizer"
          }
        },
        {
          "name": "before_pos_trigram",
          "norm": {
            "class": "org.apache.solr.ltr.norm.IdentityNormalizer"
          }
        },
        {
          "name": "before_parse_tree",
          "norm": {
            "class": "org.apache.solr.ltr.norm.IdentityNormalizer"
          }
        },
        {
          "name": "after",
          "norm": {
            "class": "org.apache.solr.ltr.norm.IdentityNormalizer"
          }
        },
        {
          "name": "after_first",
          "norm": {
            "class": "org.apache.solr.ltr.norm.IdentityNormalizer"
          }
        }
      ]
    }
  ]
}
```

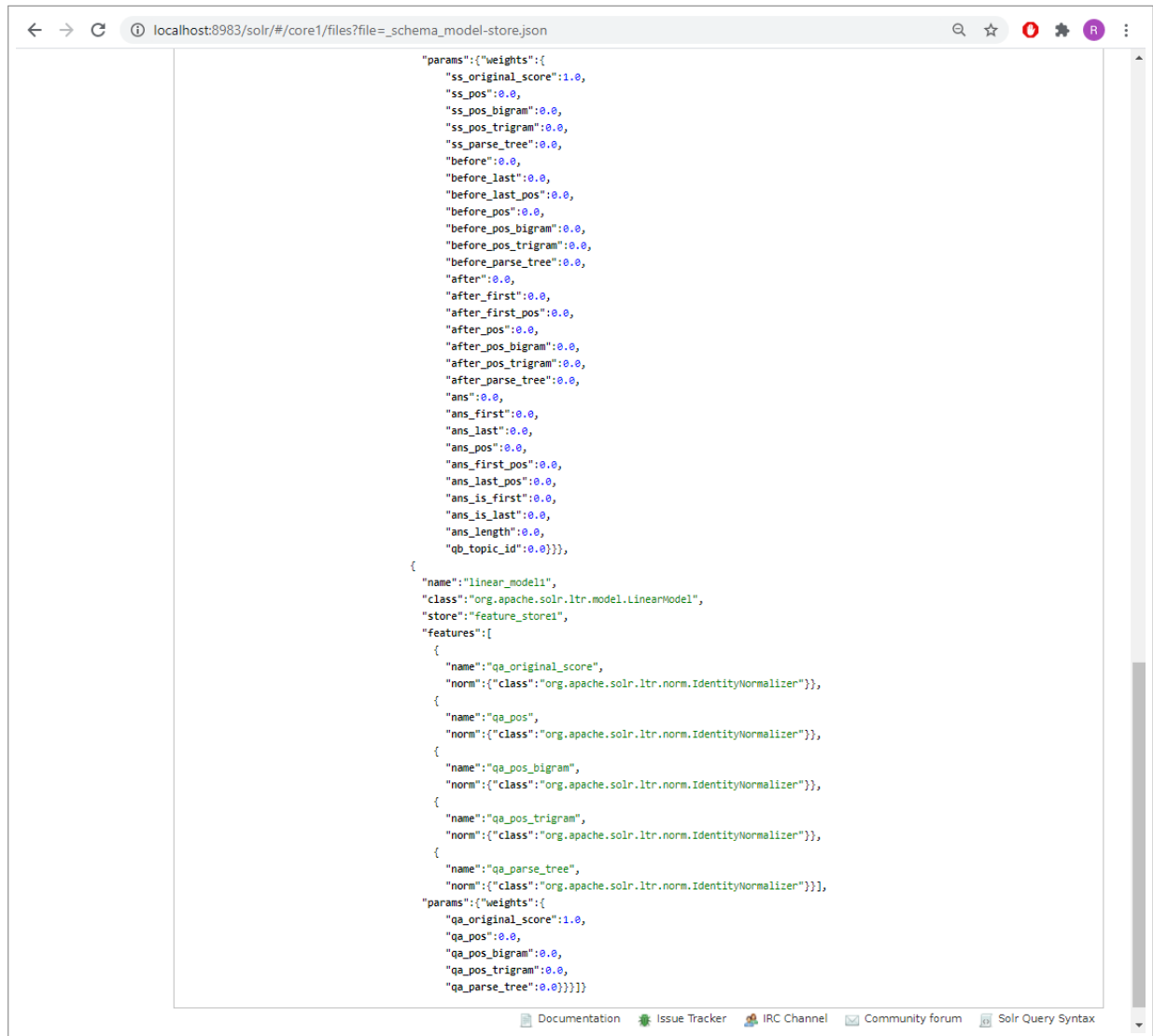


This screenshot shows the continuation of the JSON configuration from the previous image. It lists the remaining features for the 'linear_model1' model.

```

        {
          "name": "after_first_pos",
          "norm": {
            "class": "org.apache.solr.ltr.norm.IdentityNormalizer"
          }
        },
        {
          "name": "after_pos",
          "norm": {
            "class": "org.apache.solr.ltr.norm.IdentityNormalizer"
          }
        },
        {
          "name": "after_pos_bigram",
          "norm": {
            "class": "org.apache.solr.ltr.norm.IdentityNormalizer"
          }
        },
        {
          "name": "after_pos_trigram",
          "norm": {
            "class": "org.apache.solr.ltr.norm.IdentityNormalizer"
          }
        },
        {
          "name": "after_parse_tree",
          "norm": {
            "class": "org.apache.solr.ltr.norm.IdentityNormalizer"
          }
        },
        {
          "name": "ans",
          "norm": {
            "class": "org.apache.solr.ltr.norm.IdentityNormalizer"
          }
        },
        {
          "name": "ans_first",
          "norm": {
            "class": "org.apache.solr.ltr.norm.IdentityNormalizer"
          }
        },
        {
          "name": "ans_last",
          "norm": {
            "class": "org.apache.solr.ltr.norm.IdentityNormalizer"
          }
        },
        {
          "name": "ans_pos",
          "norm": {
            "class": "org.apache.solr.ltr.norm.IdentityNormalizer"
          }
        },
        {
          "name": "ans_first_pos",
          "norm": {
            "class": "org.apache.solr.ltr.norm.IdentityNormalizer"
          }
        },
        {
          "name": "ans_last_pos",
          "norm": {
            "class": "org.apache.solr.ltr.norm.IdentityNormalizer"
          }
        },
        {
          "name": "ans_is_first",
          "norm": {
            "class": "org.apache.solr.ltr.norm.IdentityNormalizer"
          }
        },
        {
          "name": "ans_is_last",
          "norm": {
            "class": "org.apache.solr.ltr.norm.IdentityNormalizer"
          }
        },
        {
          "name": "ans_length",
          "norm": {
            "class": "org.apache.solr.ltr.norm.IdentityNormalizer"
          }
        },
        {
          "name": "ob_topic_id",
          "norm": {
            "class": "org.apache.solr.ltr.norm.IdentityNormalizer"
          }
        }
      ]
    }
  ]
}
```

Linear model for Model 1:



The screenshot shows a web browser window with the address bar displaying `localhost:8983/solr/#/core1/files?file=_schema_model-store.json`. The main content area displays a JSON schema for a linear model. The schema includes parameters for weights, features, and model metadata. The weights are defined for various features, including `ss_original_score`, `ss_pos`, `ss_pos_bigram`, `ss_pos_trigram`, `ss_parse_tree`, and several positional features like `before`, `before_last`, `before_pos`, `before_pos_bigram`, `before_pos_trigram`, `before_parse_tree`, `after`, `after_first`, `after_pos`, `after_pos_bigram`, `after_pos_trigram`, `after_parse_tree`, `ans`, `ans_first`, `ans_last`, `ans_pos`, `ans_first_pos`, `ans_last_pos`, `ans_is_first`, `ans_is_last`, `ans_length`, and `qb_topic_id`. The features are defined with names and norms, all using `org.apache.solr.ltr.norm.IdentityNormalizer`. The parameters section defines the weights for the features, with `qa_original_score` set to 1.0 and other features set to 0.0.

```
{
  "params": {
    "weights": {
      "ss_original_score": 1.0,
      "ss_pos": 0.0,
      "ss_pos_bigram": 0.0,
      "ss_pos_trigram": 0.0,
      "ss_parse_tree": 0.0,
      "before": 0.0,
      "before_last": 0.0,
      "before_last_pos": 0.0,
      "before_pos": 0.0,
      "before_pos_bigram": 0.0,
      "before_pos_trigram": 0.0,
      "before_parse_tree": 0.0,
      "after": 0.0,
      "after_first": 0.0,
      "after_first_pos": 0.0,
      "after_pos": 0.0,
      "after_pos_bigram": 0.0,
      "after_pos_trigram": 0.0,
      "after_parse_tree": 0.0,
      "ans": 0.0,
      "ans_first": 0.0,
      "ans_last": 0.0,
      "ans_pos": 0.0,
      "ans_first_pos": 0.0,
      "ans_last_pos": 0.0,
      "ans_is_first": 0.0,
      "ans_is_last": 0.0,
      "ans_length": 0.0,
      "qb_topic_id": 0.0
    }
  },
  "name": "linear_model1",
  "class": "org.apache.solr.ltr.model.LinearModel",
  "store": "feature_store1",
  "features": [
    {
      "name": "qa_original_score",
      "norm": {
        "class": "org.apache.solr.ltr.norm.IdentityNormalizer"
      }
    },
    {
      "name": "qa_pos",
      "norm": {
        "class": "org.apache.solr.ltr.norm.IdentityNormalizer"
      }
    },
    {
      "name": "qa_pos_bigram",
      "norm": {
        "class": "org.apache.solr.ltr.norm.IdentityNormalizer"
      }
    },
    {
      "name": "qa_pos_trigram",
      "norm": {
        "class": "org.apache.solr.ltr.norm.IdentityNormalizer"
      }
    },
    {
      "name": "qa_parse_tree",
      "norm": {
        "class": "org.apache.solr.ltr.norm.IdentityNormalizer"
      }
    }
  ],
  "params": {
    "weights": {
      "qa_original_score": 1.0,
      "qa_pos": 0.0,
      "qa_pos_bigram": 0.0,
      "qa_pos_trigram": 0.0,
      "qa_parse_tree": 0.0
    }
  }
}
```

At the bottom of the browser window, there is a navigation bar with links to [Documentation](#), [Issue Tracker](#), [IRC Channel](#), [Community forum](#), and [Solr Query Syntax](#).

Chapter 10

Building and Testing the Models

10.1 RankLib

After the model datasets are created, we use them to build the models. We use the LambdaMART algorithm in the LTR library **RankLib**:

<https://sourceforge.net/p/lemur/wiki/RankLib/>

Download RankLib-2.15.jar from the following website and save it in /grammar/ranklib2:

<https://sourceforge.net/projects/lemur/files/lemur/RankLib-2.15/>

To train, validate and test a model, run the following command:

```
PS D:\grammar\ranklib2> java -jar RankLib-2.15.jar
-train ../path/train.txt
-test ../path/test.txt
-validate ../path/validate.txt
-ranker 6
-metric2t NDCG@10
-metric2T NDCG@10
-save ../path/model.txt
```

`ranker 6` refers to LambdaMART.

`metric2t <metric>` refers to the metric to optimize on the training data.

`metric2T <metric>` refers to the metric to evaluate on the test data. If not specified, it will use the same metric as `metric2t <metric>`.

Supported metrics are: MAP, NDCG@k, DCG@k, P@k, RR@k, ERR@k. The default metric is ERR@10.

10.2 Model Statistics

RankLib generates model statistics. First, download the Apache Commons Mathematics library `commons-math3-3.5.jar` from:

<http://commons.apache.org/proper/commons-math/>

Save this file in the `/grammar/ranklib2` and run the following command in Windows command prompt:

```
D:\grammar\ranklib2> java -cp RankLib-2.15.jar;commons-math3-3.5.jar
ciir.umass.edu.features.FeatureManager -feature_stats model_name.txt
```

For Model 1, trained using the metric MAP, we will see the following output:

```
Algorithm : LambdaMART

Feature frequencies :
    Feature[1] :      291
    Feature[2] :      232
    Feature[3] :      226
    Feature[4] :      385
    Feature[5] :      261

Total Features Used: 5

Min frequency      :      226.00
Max frequency      :      385.00
Median frequency   :      261.00
Avg frequency      :      279.00
Variance           :      4180.50
STD                :        64.66
```

10.3 Converting Models to JSON

The tree models from RankLib are in text format. We need to convert them to JSON before uploading them to Solr. Copy the following notebook from the repository directory and run it:

`/grammar/6_convert_model_json.ipynb`

This notebook converts the models from text format to XML format and then to JSON format.

10.4 Uploading Models and Testing with Queries

The final step is to upload the full models and baseline models for Model 1 and Model 2 to Solr. Then we test the models with queries. To upload the four models, copy the following notebook from the repository directory and run it:

```
/grammar/7_solr_upload_model.ipynb
```

We select a random query and send it to Solr:

```
/grammar/8_solr_send_query.ipynb
```