CS410:Text Information Systems

Fall 2017 Final Project

Project Title - Information Extraction form MCSDS lecture material & Implementation of Search Engine.

Sujith Achuthan

12/18/2017

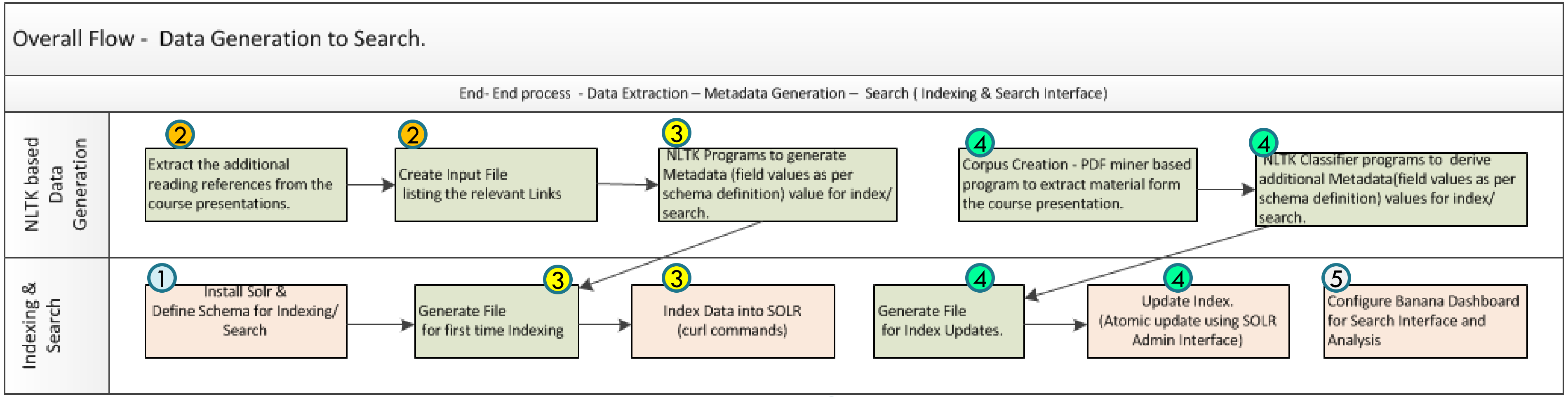
Table Of Contents.

About the Project

<u>Overview</u>

The MSDS lecture material contains a plethora of information in the form of links and references to external additional materials for a particular topic. The additional reading recommendations and references mentioned are spread across different lecture slides making it cumbersome to be accessed as the courses progress. This project aims at capturing these references from the lecture slides into an easily searchable system thus enabling easy access to information/knowledge regarding a particular topic. The end-end implementation/development of this system will be done using python NLP libraries for information extraction & SOLR search related stacks for the search engine (SOLR for search & Banana for the Search interface).Such a tool will be extremely useful for all learners pursuing this course or similar courses.

Modules and High-level Process


Overall Flow - Data Generation to Search.
End- End process - Data Extraction – Metadata Generation – Search (Indexing & Search Interface)
NLTK based Data Generation
Indexing & Search
2 Extract the additional reading references from the course presentations.
2 Create Input File listing the relevant Links
3 NLTK Programs to generate Metadata (field values as per schema definition) value for index/ search.
4 Corpus Creation - PDF miner based program to extract material form the course presentation.
4 NLTK Classifier programs to derive additional Metadata(field values as per schema definition) values for index/ search.
1 Install Solr & Define Schema for Indexing/ Search
3 Generate File for first time Indexing
3 Index Data into SOLR (curl commands)
4 Generate File for Index Updates.
4 Update Index. (Atomic update using SOLR Admin Interface)
5 Configure Banana Dashboard for Search Interface and Analysis

- The flow diagram above explains the logical breakdown of the project modules.
- The objective is to curate the additional readings and references from the coursera lecture slides so that it can be referenced easier.
- (2) The lecture slides are downloaded manually , and the links are extracted manually and fed into the respective programs for (3) entity recognition and derived values mapped to the appropriate schema element for Solr Search . The original objective was to extract the links programmatically, however had challenges in separating multiple references occurring on the same slide .

- (1) Solr with appropriate schema definition should be defined and (3) indexing is achieved with curl commands & also by using Solr (4) Admin page for atomic updates.
- PDF miner is used to extract the text from PDF for generating the (4) corpus for inputs to the classifier program .The classifier program classifies the reference(additional/suggested reading) to the respective predefined sub_category . This sub_category is also defined as a solr schema field and updated as part of second indexing run.
- (5) Banana dashboard to enable the Usecases of search & analysis.

Note - Details of the above process/steps are mentioned in the respective slides.

Technology Stacks Used.

Python & Development IDE

- WinPython3530Qt5 64bit on Windows 7 Enterprise 64bit https://winpython.github.io/#overview
- IDE – Spyder 3.1.2 supplied along with WinPython3530Qt5 64bit

Python NLTK Packages

- Stanford NER - v3.8.0 - 2017-06-09 https://nlp.stanford.edu/software/CRF-NER.shtml#Download
- Python NLTK version 3..2.5 package which is included in WinPython3530Qt5 64bit distribution.
- Additional NLTK packages – Averaged Perceptron Tagger ,Punkt Tokenizer Models,Stopwords Corpus (refer backup slides for snapshots of install.)
- pdfminer3k https://pypi.python.org/pypi/pdfminer3k/

Java

- jdk1.8.0_131is required to set JAVA_HOME and launch Spyder. This is required for the NER tagger programs (explained in the respective section.)

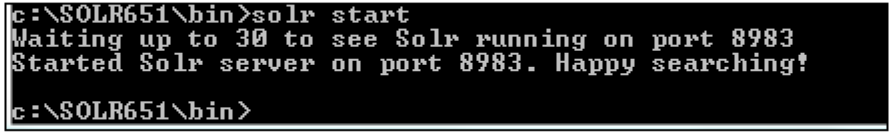
Solr Related

- Solr version 6.5.1 is used for this project. Download the Solr version from the location http://archive.apache.org/dist/lucene/solr/
- Cygwin version 2.882(64 bit) , this is only required for using the curl utility for indexing into Solr.
- Banana 1.6.17 https://github.com/lucidworks/banana deployed as a web application and is a data visualization tool for search, data analysis and display of Solr data.

SOLR – Creation of CS410 Collection

Basic Solr Install & starting the server.

1)Go to the SOLR installation e.g. c:\SOLR651\bin and run the batch file solr.cmd with the following argument >solr start



```
c:\SOLR651\bin>solr start
Waiting up to 30 to see Solr running on port 8983
Started Solr server on port 8983. Happy searching!

c:\SOLR651\bin>
```

Solr is started and listening on the default port. In case you wish to change the port one way is to run the above command with the port specified (e.g.   >solr start –p=8994).

To stop the sever use the following command

>solr stop –p=8983

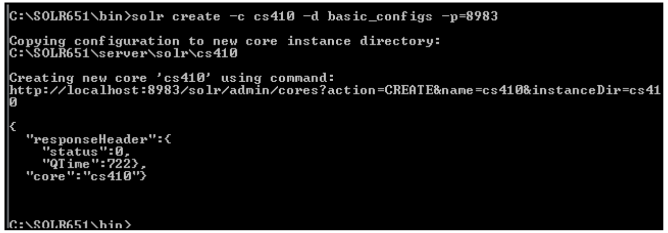
2)To verify if Solr is running pls navigate to the respective url on the localport http://localhost:8983/solr/#/

This will display the Solr admin page .

Creation of CS410 Collection.

3) Go to the SOLR installation e.g. c:\SOLR651\bin and issue the following command >solr create -c cs410 -d basic_configs -p=8983

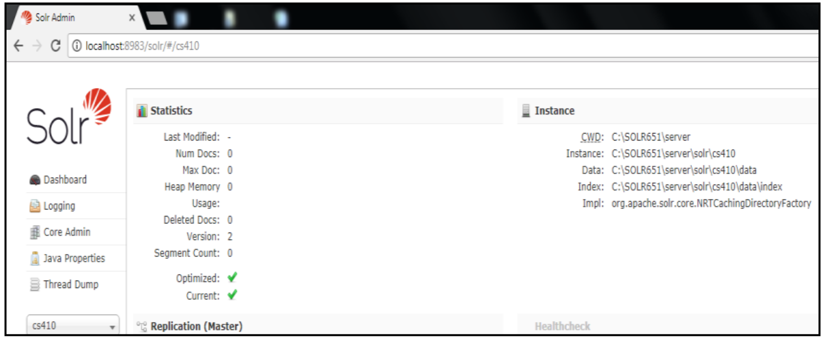
e.g. shown below in the image.



```
C:\SOLR651\bin>solr create -c cs410 -d basic_configs -p=8983

Copying configuration to new core instance directory:
C:\SOLR651\server\solr\cs410

Creating new core 'cs410' using command:
http://localhost:8983/solr/admin/cores?action=CREATE&name=cs410&instanceDir=cs41
0

{
  "responseHeader":{
    "status":0,
    "QTime":722},
  "core":"cs410"}



C:\SOLR651\bin>
```

4) Check if the collection has been created successfully using the Solr Admin Application.

http://localhost:8983/solr/#/cs410



```
localhost:8983/solr/#/cs410

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: C:\SOLR651\server
 Instance: C:\SOLR651\server\solr\cs410
     Data: C:\SOLR651\server\solr\cs410\data
    Index: C:\SOLR651\server\solr\cs410\data\index
     Impl: org.apache.solr.core.NRTCachingDirectoryFactory
```

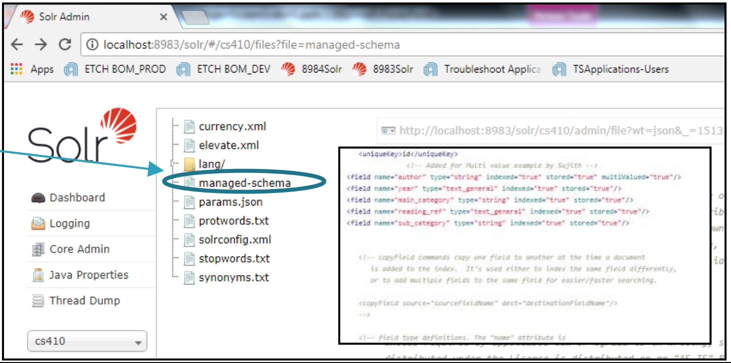
SOLR – Schema definition for CS410 Collection

Define SOLR Schema for the Indexing & Search Purposes

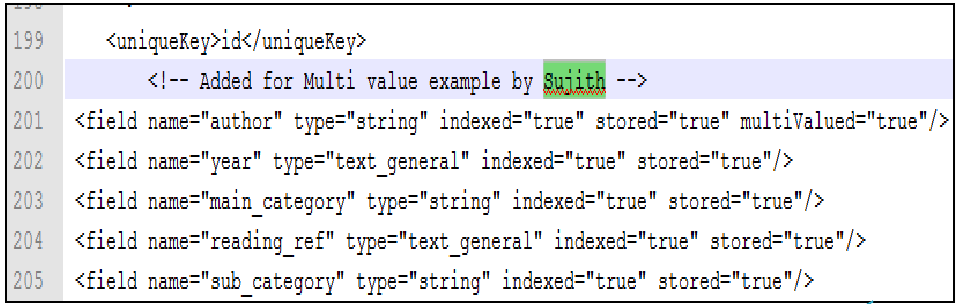
- Modify the file “managed-schema” for the CS410 collection .This file is located at C:\SOLR651\server\solr\cs410\conf.

Defining fields & field properties.

- To cater to the final search use-cases/analysis planned for this project define the following schema elements as show.
- The fields “author”,”year”,”main_category”, “reading_ref” & “sub_category” are defined.
- Note – This can also defined using the solr admin interface. The schema file “managed-schema” is uploaded as part of the software deposit into the github location.



```
<uniqueKey>id</uniqueKey>
        <!-- Added for Multi value example by Sujith -->
<field name="author" type="string" indexed="true" stored="true" multiValued="true"/>
<field name="year" type="text_general" indexed="true" stored="true"/>
<field name="main_category" type="string" indexed="true" stored="true"/>
<field name="reading_ref" type="text_general" indexed="true" stored="true"/>
<field name="sub_category" type="string" indexed="true" stored="true"/>


 <!-- copyField commands copy one field to another at the time a document
   is added to the index.  It's used either to index the same field differently,
   or to add multiple fields to the same field for easier/faster searching.

 <copyField source="sourceFieldName" dest="destinationFieldName"/>
 -->

 <!-- field type definitions. The "name" attribute is
```



```
  <uniqueKey>id</uniqueKey>
      <!-- Added for Multi value example by Sujith -->
<field name="author" type="string" indexed="true" stored="true" multiValued="true"/>
<field name="year" type="text_general" indexed="true" stored="true"/>
<field name="main_category" type="string" indexed="true" stored="true"/>
<field name="reading_ref" type="text_general" indexed="true" stored="true"/>
<field name="sub_category" type="string" indexed="true" stored="true"/>
```

File managed-schema is uploaded into Github repository

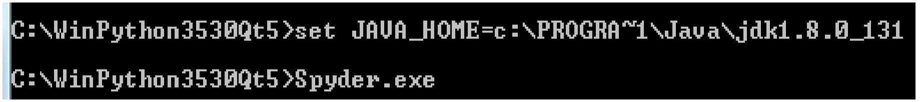
Additional Notes – For indexing data the documents are generated by the respective python programs & Indexed using solr indexing commands.

- ID – Mandatory attribute and is authored manually and is used for the first indexing and further for atomic updates of fields.
- Fields author & year are derived by the NER tagger programs.
- Field sub_category is derived by the classification program and the output is used for the second indexing or updating(atomic update) the value of sub_category.
- Details of the above fields and how the value is set/derived is covered in the respective sections where the python programs are explained.

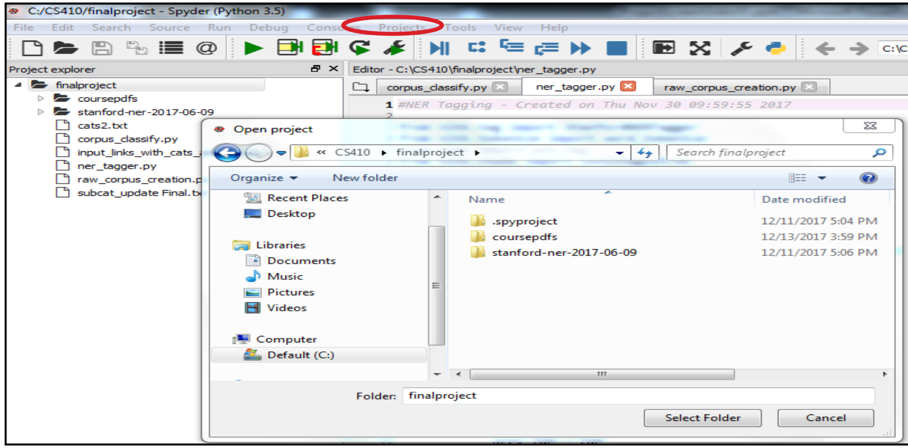
Spyder IDE - for running the Python files.

Launch Spyder IDE

- Go to the folder where WinPython3530Qt5 is installed.
- Launch the WinPython Command Prompt.exe.
- Set JAVA_HOME to the JDK version e.g.
  set JAVA_HOME=c:\PROGRA~1\Java\jdk1.8.0_131
- Launch Spyder using the Spyder.exe command.



```
C:\WinPython3530Qt5>set JAVA_HOME=c:\PROGRA~1\Java\jdk1.8.0_131

C:\WinPython3530Qt5>Spyder.exe
```



finalproject
- coursepdfs
- stanford-ner-2017-06-09
- cats2.txt
- corpus_classify.py
- input_links_with_cats_
- ner_tagger.py
- raw_corpus_creation.p
- subcat_update Final.tx

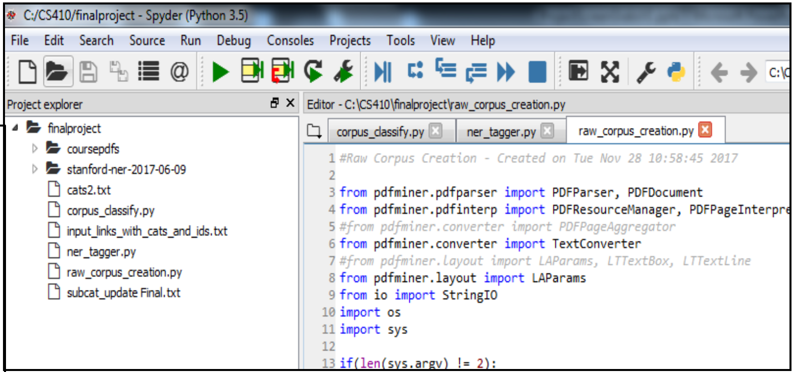
```
#NER Tagging - Created on Thu Nov 30 09:59:55 2017
```

Name	Date modified
.spyproject	12/11/2017 5:04 PM
coursepdfs	12/13/2017 3:59 PM
stanford-ner-2017-06-09	12/11/2017 5:06 PM

Folder: finalproject

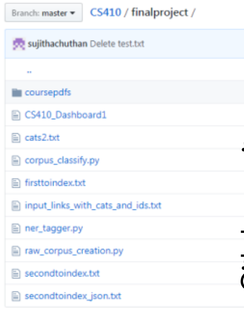
Loading the Project in Spyder.

- Use the "Projects" – "Open Project…" from the Spyder menu and navigate to the project root location and load the project into Spyder as shown in the image 1.



finalproject
- coursepdfs
- stanford-ner-2017-06-09
- cats2.txt
- corpus_classify.py
- input_links_with_cats_and_ids.txt
- ner_tagger.py
- raw_corpus_creation.py
- subcat_update Final.txt

```
#Raw Corpus Creation - Created on Tue Nov 28 10:58:45 2017

from pdfminer.pdfparser import PDFParser, PDFDocument
from pdfminer.pdfinterp import PDFResourceManager, PDFPageInterpre
#from pdfminer.converter import PDFPageAggregator
from pdfminer.converter import TextConverter
#from pdfminer.layout import LAParams, LTTextBox, LTTextLine
from pdfminer.layout import LAParams
from io import StringIO
import os
import sys

if(len(sys.argv) != 2):
```



sujithachuthan Delete test.txt

- ..
- coursepdfs
- CS410_Dashboard1
- cats2.txt
- corpus_classify.py
- firsttoindex.txt
- input_links_with_cats_and_ids.txt
- ner_tagger.py
- raw_corpus_creation.py
- secondtoindex.txt
- secondtoindex_json.txt

Github repository

Note on files Uploaded into Github

- All the source(pdf,txt) files ,python files & program output files are uploaded. Also uploaded are files that can be indexed to solr.
- Dependencies like ner tagger etc. are not uploaded.
- Also the .spyderproject is not uploaded since uploading .ini files were causing issues.
- So pls built the project as shown on the snapshots to the right.

Program 1 ner_tagger.py – Run & Explanations

About the Program – ner_tagger.py

- This program implements the ntlk modules to extract the entities "author" & "year" from the additional reading recommendations in the course material and generate the file for the first time index to solr.
- The input for this file is a "|" separated .txt file which captures the following details (refer to the below image for sample).This input file is created manually by extracting the additional readings recommendations from each topic in the cs410 lecture pdf's.

(1)ID-For the mandatory define Solr document ID. This is also used later for atomic updates further during the second index run.
(2)File name where the addition reading material was observed. Note that this field has a .txt mentioned as the file name only because this same input file is used during the next phase of the classification program (will be clear in the coming slides).
(3)main_category of this link/additional references and is classified manually.
(4)This is the main additional reference material found in the respective course lecture slides . The NER tagger uses this to derive the Author & Year attributes.

- The input file for this program is also the input file for further programs (classification program to derive the field sub_category) and hence some attributes in this file is designed in such a way to allow linking of the file name and Solr id.

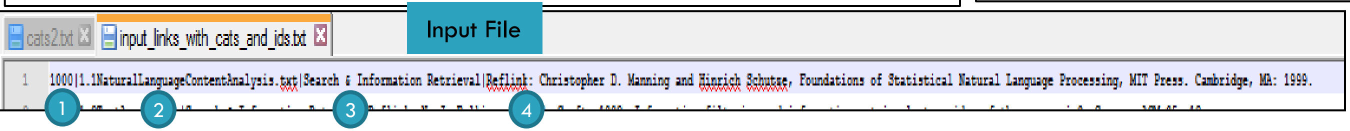
- The second input file is the root directory of stanford ner tagger libraries.
- The program uses both 4&7 class algorithms from the stanford tagger to arrive at the Person(author) & Date(Year) attributes.

Challenges Faced

- Since the links was not very organized in the input files it was difficult to mine it using PDF miner. However, PDF miner was used to extract and create corpus , which will be explained in the coming slides.
- Also for the ner tagger to perform some data like punctuations spaces etc.. Were cleaned (trail & error)

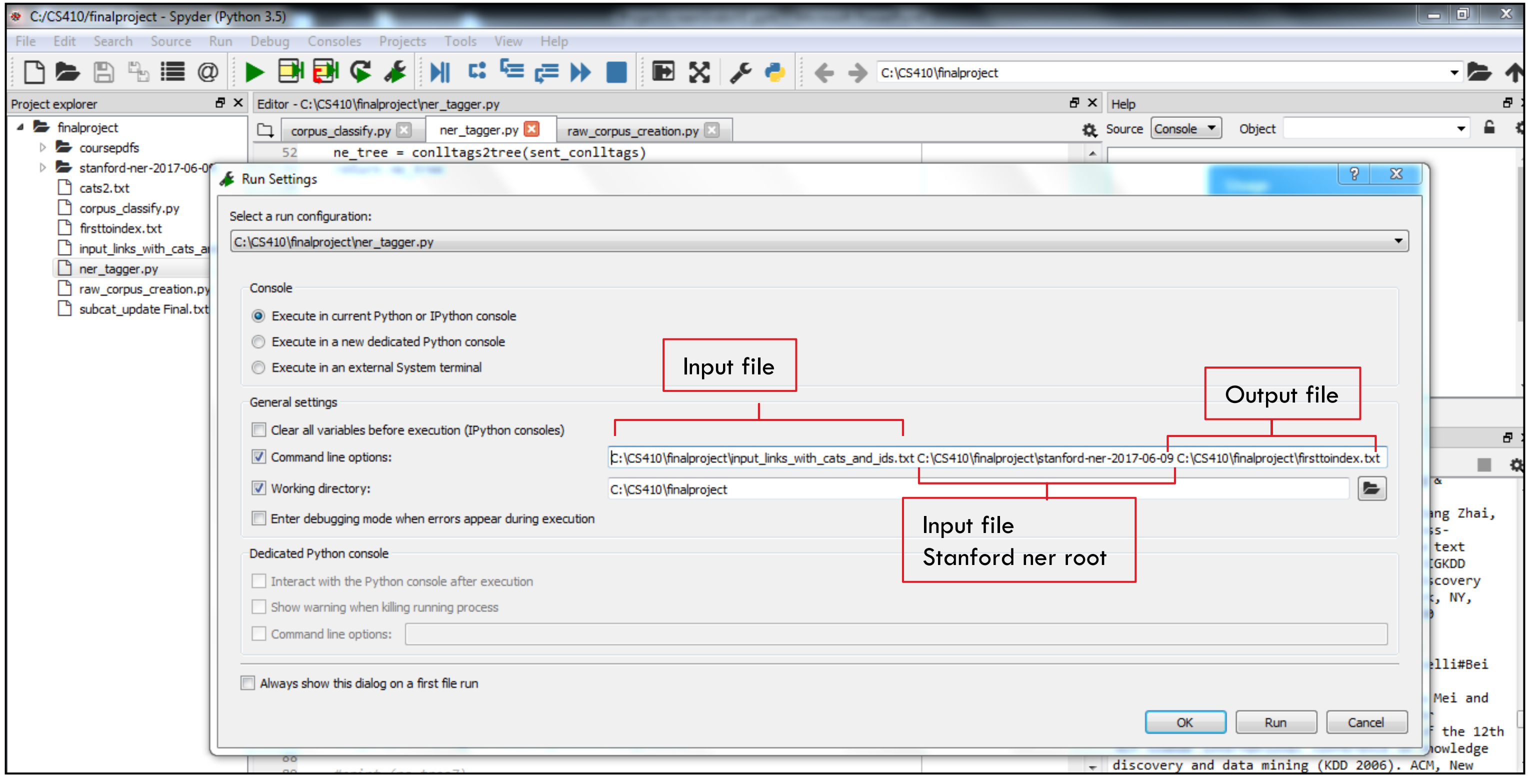
Output from this program.

- The output of the program "firsttoindex" is a "|" separator file
- This file contains the value for the following fields ID|author|year|main_category|reading_ref
- The field author has been modeled as a multivalued field and multiple values are separated by "#" . This file is used for the first time indexing to solr , header values for this file is manually edited before indexing (mentioned in future slides)



```
1000|1.1NaturalLanguageContentAnalysis.txt|Search & Information Retrieval|Reflink: Christopher D. Manning and Hinrich Schutze, Foundations of Statistical Natural Language Processing, MIT Press. Cambridge, MA: 1999.
```

Running/Providing arguments to ner_tagger.py in Spyder



```
52    ne_tree = conlltags2tree(sent_conlltags)
```

Run Settings

Select a run configuration: C:\CS410\finalproject\ner_tagger.py

Console

- [✓ Execute in current Python or IPython console]
- Execute in a new dedicated Python console
- Execute in an external System terminal

General settings

- ☐ Clear all variables before execution (IPython consoles)
- ☑ Command line options: `C:\CS410\finalproject\input_links_with_cats_and_ids.txt C:\CS410\finalproject\stanford-ner-2017-06-09 C:\CS410\finalproject\firsttoindex.txt`
- ☑ Working directory: C:\CS410\finalproject
- ☐ Enter debugging mode when errors appear during execution

Dedicated Python console

- ☐ Interact with the Python console after execution
- ☐ Show warning when killing running process
- ☐ Command line options:

☐ Always show this dialog on a first file run

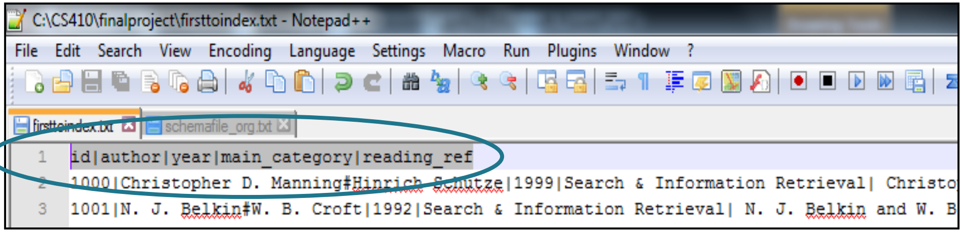
Note – The run of this program will take ~ 5 minutes with java exe window popping up for each Links being processed.

Indexing Data into Solr – First Run

Preparing Data to Index

- The output from the ner_tagger.py is indexed into solr at this step.
  e.g Output file name – firsttoindex.txt
- For indexing this file fist open this file and add the first line which represents the header (solr fields) (copy paste the below line as the first line of this file)
  id|author|year|main_category|reading_ref

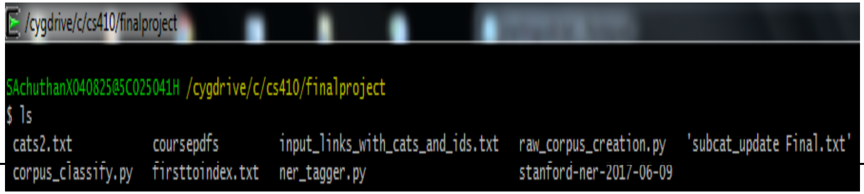
e.g. as shown below.



```
id|author|year|main_category|reading_ref
1000|Christopher D. Manning#Hinrich Schutze|1999|Search & Information Retrieval| Christo
1001|N. J. Belkin#W. B. Croft|1992|Search & Information Retrieval| N. J. Belkin and W. B
```

Indexing Data

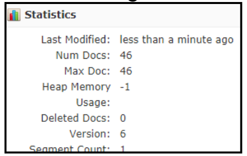
- Launch Cygwin to use the curl command for indexing and cd to the project directory



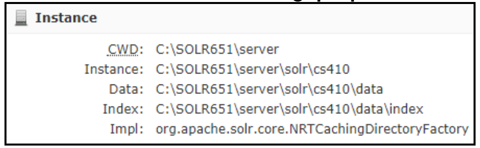
```
SAchuthanX040825@5C025041H /cygdrive/c/cs410/finalproject
$ ls
cats2.txt          coursepdfs        input_links_with_cats_and_ids.txt   raw_corpus_creation.py   'subcat_update Final.txt'
corpus_classify.py firsttoindex.txt  ner_tagger.py                       stanford-ner-2017-06-09
```

Indexing Data Continued

- Use the curl command with the following options to index the data. >curl
  http://localhost:8983/solr/cs410/update?commit=true&separator=%7C&f.author.split=true&f.author.separator=%23' --data-binary @. Firsttoindex.txt -H 'Content-type:application/csv'
- Refer to the backup slides for image capture of the above command being executed.
- Note - The file being indexed is "|" separated .The author field can contain multiple values (multivalued) and is separated by "#"
- Check using Solr Admin tool for the collection being populated



```
Statistics
   Last Modified:  less than a minute ago
       Num Docs:  46
        Max Doc:  46
    Heap Memory   -1
          Usage:
   Deleted Docs:  0
        Version:  6
  Segment Count:  1
```



```
Instance
      CWD:  C:\SOLR651\server
 Instance:  C:\SOLR651\server\solr\cs410
     Data:  C:\SOLR651\server\solr\cs410\data
    Index:  C:\SOLR651\server\solr\cs410\data\index
     Impl:  org.apache.solr.core.NRTCachingDirectoryFactory
```

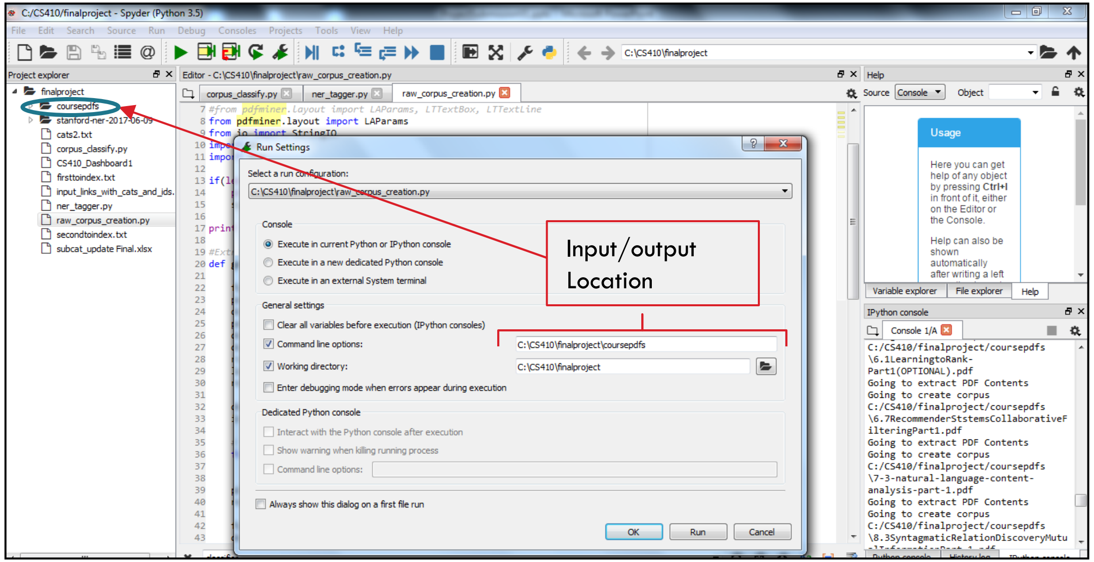
Note

- This step we are indexing only 5 fields as per the input file , the field "sub_category" will be indexed as part of index update process. At this point you can go forward to configure Banana Dashboards or continue with the classifier program to derive the field "sub_categoty".

Program 2 raw_corpus_creatio.py– Run & Explanations

Raw corpus Creation

- The idea here to create a corpus is with the intension to derive the field “sub_category” using a document classifier model (detailed in the next slide).
- This program creates the corpus for the classifier program.
- PDF Miner is used to scan for all the PDFs where the strings “Suggested Reading” or “Additional Reading "occurs and for such Lecture slides all the contents are read and written into respective text file in the output directory. E.g. location C:\CS410\finalproject\coursepdfs
- The .txt fie takes the same naming convention of the input pdf files but with an extension of .txt.
- The .txt files are created in the same location as that of the input pdf(source) files.
- The input files(pdf) the output .txt files have been uploaded into the code repository.



```
#from pdfminer.layout import LAParams, LTTextBox, LTTextLine
from pdfminer.layout import LAParams
```

Run Settings

Select a run configuration: C:\CS410\finalproject\raw_corpus_creation.py

Console
- [✓] Execute in current Python or IPython console
- Execute in a new dedicated Python console
- Execute in an external System terminal

General settings
- ☐ Clear all variables before execution (IPython consoles)
- ☑ Command line options: C:\CS410\finalproject\coursepdfs
- ☑ Working directory: C:\CS410\finalproject
- ☐ Enter debugging mode when errors appear during execution

Dedicated Python console
- ☐ Interact with the Python console after execution
- ☐ Show warning when killing running process
- ☐ Command line options:

☐ Always show this dialog on a first file run

OK Run Cancel

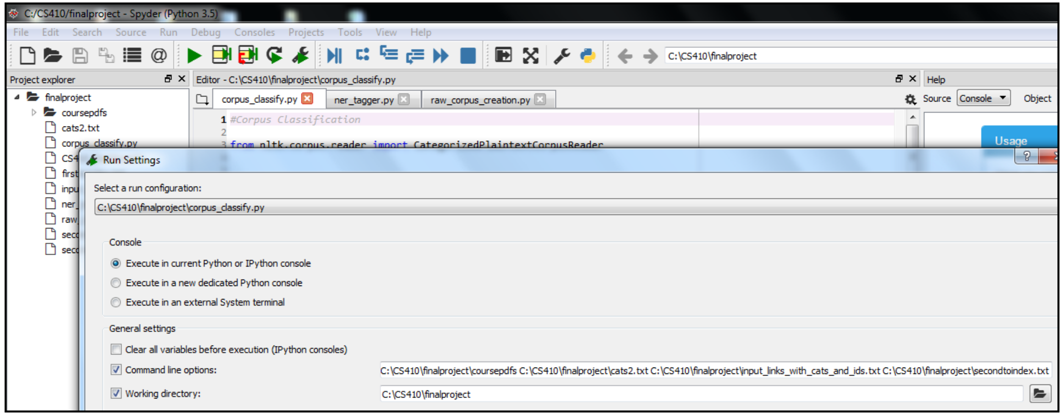
IPython console

```
C:/CS410/finalproject/coursepdfs
\6.1LearningtoRank-
Part1(OPTIONAL).pdf
Going to extract PDF Contents
Going to create corpus
C:/CS410/finalproject/coursepdfs
\6.7RecommenderStstemsCollaborativeF
ilteringPart1.pdf
Going to extract PDF Contents
Going to create corpus
C:/CS410/finalproject/coursepdfs
\7-3-natural-language-content-
analysis-part-1.pdf
Going to extract PDF Contents
Going to create corpus
C:/CS410/finalproject/coursepdfs
\8.3SyntagmaticRelationDiscoveryMutu
```

Program 3 corpus_classify.py – Run and Explanations

Classification Program Details.

- This program takes the respective inputs (arguments) and the contents extracted to classify the files & thus the links/additional reading materials into pre-defined category. Below points details the implementation.
- Classifier Used - Simple Naive Bayes for Multi Class Classification
- Corpus Created - Categorized Corpus Based on Category to File ID Mapping.
- Initial / Pre-Defined Features for Test and Train Hardcoded as a List of Tuples
- Naive Bayes Classifier used on Pre-Defined Feature Train List of Tuples
- Test and Train Data split based on the Corpus created in Point 2.
- Updated Classifier with Train data information for increasing training accuracy
- Calculated accuracy on the Pre-Defined Test and the Actual Corpus Based Test Dataset
- Looping through each dataset in the Corpus - calculated Probability of a Category based on entire Corpus Dataset
- Filtered out only those categories that had a probability of greater than 50%
- Cross-Referenced ROW ID's from the Original Links File and created a final sub-category file adding Row ID and Derived Category


C:/CS410/finalproject - Spyder (Python 3.5)

Editor - C:\CS410\finalproject\corpus_classify.py

```
#Corpus Classification

from nltk.corpus.reader import CategorizedPlaintextCorpusReader
```

Run Settings

Select a run configuration: C:\CS410\finalproject\corpus_classify.py

Console
- [✓ Execute in current Python or IPython console]
- Execute in a new dedicated Python console
- Execute in an external System terminal

General settings
- [ ] Clear all variables before execution (IPython consoles)
- [✓] Command line options: C:\CS410\finalproject\coursepdfs C:\CS410\finalproject\cats2.txt C:\CS410\finalproject\input_links_with_cats_and_ids.txt C:\CS410\finalproject\secondtoindex.txt
- [✓] Working directory: C:\CS410\finalproject

For details of Input and Output arguments pls refer to the table on the next slide.

Output Files (arguments)

- The output file e.g. "secondtoindex.txt" contains the sub_category value from the classification & the Solr document ID.
- This file is further converted/used for updates into solr using the Solr admin interface (refer next slide for details)

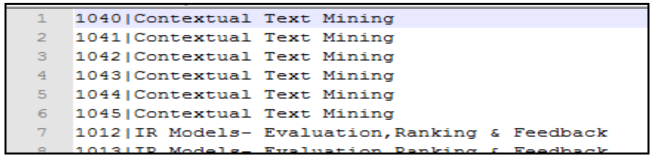
Program 3 corpus_classify.py – Arguments Explained.

File Argument / Description	Example
Location / Path to Directory where Input Coursera PDFs are stored – First Argument	“C:\\CS410\\finalproject\\coursepdfs”
Path to File (along with File Name) containing Category and Input File ID (Text File from Raw Corpus) Mapping delimited using “PIPE” – Second Argument **Example Contents Of File:** 1.1NaturalLanguageContentAnalysis.txt|General 1.2TextAccess.txt|General 9.9LatentDirichletAllocation(LDA)Part2.txt|Topic Models, Clustering & Categorization	“C:\\CS410\\finalproject\\cats2.txt”
Path to File (along with File Name) containing details of Reference Links along with Primary Categories and Record ID delimited using “PIPE”– Third Argument **Example Contents Of File: Record ID|File Name/FileID|Primary Category|Reference Link** 1000|1.1NaturalLanguageContentAnalysis.txt|Search & Information Retrieval|Reflink: Christopher D. Manning and Hinrich Schutze, Foundations of Statistical Natural Language Processing, MIT Press. Cambridge, MA: 1999. 1001|1.2TextAccess.txt|Search & Information Retrieval|Reflink: N. J. Belkin and W. B. Croft. 1992. Information filtering and information retrieval: two sides of the same coin?. Commun. ACM 35, 12 1002|1.3TextRetreivalProblem.txt|Search & Information Retrieval|Reflink: S.E Robertson, The probability ranking principle in IR. Journal of Documentation 33, 294-304, 1977 1003|1.3TextRetreivalProblem.txt|Search & Information Retrieval|Reflink: C. J. van Rijsbergen, Information Retrieval, 2nd Edition, Butterworth-Heinemann, Newton, MA, USA, 1979	"C:\\CS410\\finalproject\\input_links_with_cats_and_ids.txt"
Path To Output file where final Derived Sub-Category will be written along with Record ID – Fourth Argument **Example Contents Of File:** 1032|Topic Models, Clustering & Categorization 1033|Topic Models, Clustering & Categorization 1015|IR Models & Implementations	"C:\\CS410\\finalproject\\secondtoindex.txt"

Solr Indexing – Atomic Update of sub_category field

Updating the sub_category field.

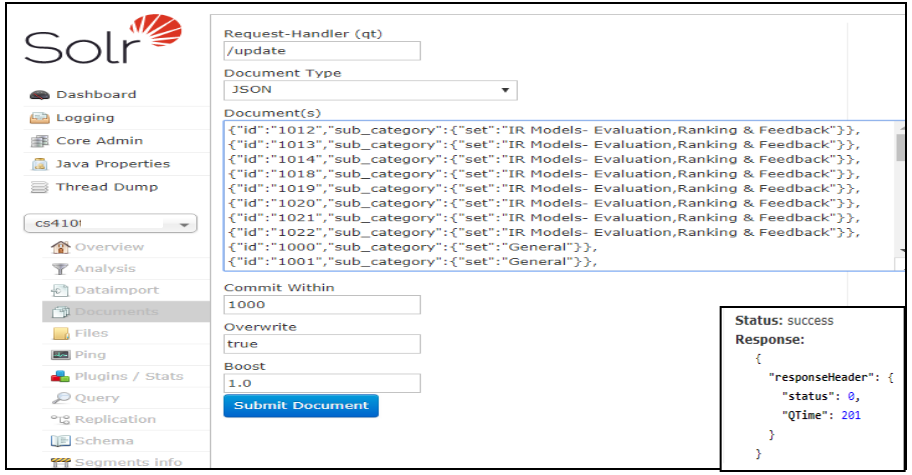
- The output of the program "corpus_classify.py" contains the ID and the corresponding sub_category value for update into Solr



```
1040|Contextual Text Mining
1041|Contextual Text Mining
1042|Contextual Text Mining
1043|Contextual Text Mining
1044|Contextual Text Mining
1045|Contextual Text Mining
1012|IR Models- Evaluation,Ranking & Feedback
1013|IR Models- Evaluation,Ranking & Feedback
```

- Knowing the ID of the document and the "sub_category" from the file , we can now proceed for Atomic updates into Solr i.e. update only the relevant field value for a given document rather than re-indexing the whole collection.

- This can be achieved by using the Solr Admin page as shown here. The json file has been manually authored from the above text file .Uploaded into the code repository is the formatted file which you can cut and paste into solr admin for atomic updates.



Request-Handler (qt): /update

Document Type: JSON

Document(s):

```
{"id":"1012","sub_category":{"set":"IR Models- Evaluation,Ranking & Feedback"}},
{"id":"1013","sub_category":{"set":"IR Models- Evaluation,Ranking & Feedback"}},
{"id":"1014","sub_category":{"set":"IR Models- Evaluation,Ranking & Feedback"}},
{"id":"1018","sub_category":{"set":"IR Models- Evaluation,Ranking & Feedback"}},
{"id":"1019","sub_category":{"set":"IR Models- Evaluation,Ranking & Feedback"}},
{"id":"1020","sub_category":{"set":"IR Models- Evaluation,Ranking & Feedback"}},
{"id":"1021","sub_category":{"set":"IR Models- Evaluation,Ranking & Feedback"}},
{"id":"1022","sub_category":{"set":"IR Models- Evaluation,Ranking & Feedback"}},
{"id":"1000","sub_category":{"set":"General"}},
{"id":"1001","sub_category":{"set":"General"}},
```

Commit Within: 1000

Overwrite: true

Boost: 1.0

```
Status: success
Response:
  {
    "responseHeader": {
      "status": 0,
      "QTime": 201
    }
  }
```

- Pls note the above image and the arguments used for the update via the Submit Document button.
- secondtoindex_json.txt has been uploaded into github repository can be used for this.

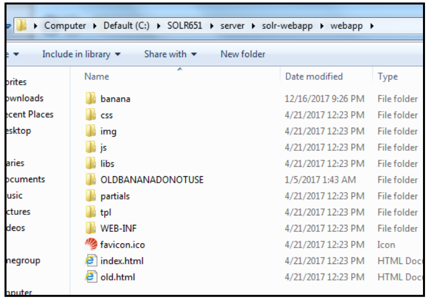
By now we have all the fields populated for further search/data analysis using Banana Dashboards.

Banana Dashboard- Install and Configure for CS410

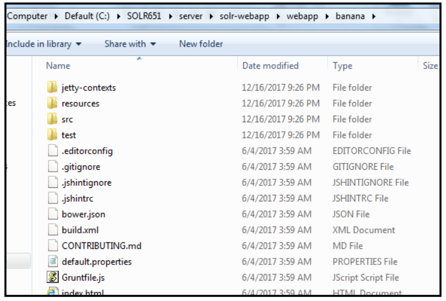
Installing & Configuring Banana

- Banana 1.6.17 https://github.com/lucidworks/banana deployed as a web application and is a data visualization tool for search, data analysis and display of Solr data.

- Unzip the package and copy it into the location c:\solr651\server\solr-webapp\webapp\banana



Name	Date modified	Type
banana	12/16/2017 9:26 PM	File folder
css	4/21/2017 12:23 PM	File folder
img	4/21/2017 12:23 PM	File folder
js	4/21/2017 12:23 PM	File folder
libs	4/21/2017 12:23 PM	File folder
OLDBANANADONOTUSE	1/5/2017 1:43 AM	File folder
partials	4/21/2017 12:23 PM	File folder
tpl	4/21/2017 12:23 PM	File folder
WEB-INF	4/21/2017 12:23 PM	File folder
favicon.ico	4/21/2017 12:23 PM	Icon
index.html	4/21/2017 12:23 PM	HTML Doc
old.html	4/21/2017 12:23 PM	HTML Doc


Computer ▸ Default (C:) ▸ SOLR651 ▸ server ▸ solr-webapp ▸ webapp ▸ banana ▸
Include in library
Share with
New folder

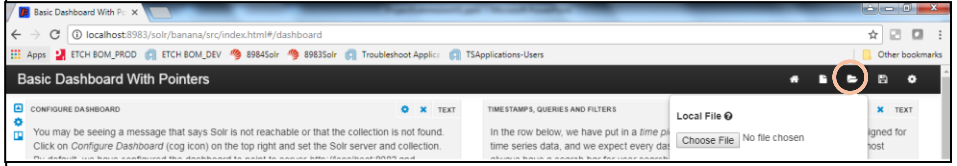
Name	Date modified	Type	Size
jetty-contexts	12/16/2017 9:26 PM	File folder	
resources	12/16/2017 9:26 PM	File folder	
src	12/16/2017 9:26 PM	File folder	
test	12/16/2017 9:26 PM	File folder	
.editorconfig	6/4/2017 3:59 AM	EDITORCONFIG File	
.gitignore	6/4/2017 3:59 AM	GITIGNORE File	
.jshintignore	6/4/2017 3:59 AM	JSHINTIGNORE File	
.jshintrc	6/4/2017 3:59 AM	JSHINTRC File	
bower.json	6/4/2017 3:59 AM	JSON File	
build.xml	6/4/2017 3:59 AM	XML Document	
CONTRIBUTING.md	6/4/2017 3:59 AM	MD File	
default.properties	6/4/2017 3:59 AM	PROPERTIES File	
Gruntfile.js	6/4/2017 3:59 AM	JScript Script File	
index.html	6/4/2017 3:59 AM	HTML Document	

More Details On Configuring Banana Dashboard

- Pls refer to the technology review submission for more details on Banana dashboards and basic concepts.
- Here I am not showing the steps of creation of the dashboard but just highlighting the dashboard features I have used and loading the already configured dashboard.

Loading the Dashboard for this project.

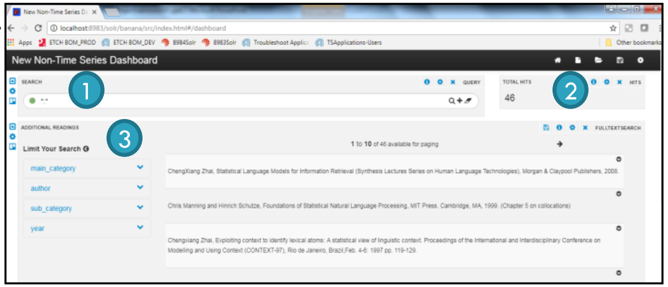
- Navigate to the Banana home in a browser with the below url http://localhost:8983/solr/banana/src/index.html#/dashboard



You may be seeing a message that says Solr is not reachable or that the collection is not found. Click on *Configure Dashboard* (cog icon) on the top right and set the Solr server and collection.

In the row below, we have put in a *time pi* time series data, and we expect every das

- Use the icon "Load" from the menu bar and click choose file to load the preconfigured dashboard "CS410_Dashboard1" from your local machine. Make sure you have downloaded the "CS410_Dashboard" from the code repository.



ChengXiang Zhai, Statistical Language Models for Information Retrieval (Synthesis Lectures Series on Human Language Technologies), Morgan & Claypool Publishers, 2008.

Chris Manning and Hinrich Schutze, Foundations of Statistical Natural Language Processing, MIT Press. Cambridge, MA, 1999. (Chapter 5 on collocations)

Chengxiang Zhai, Exploiting context to identify lexical atoms: A statistical view of linguistic context. Proceedings of the International and Interdisciplinary Conference on Modeling and Using Context (CONTEXT-97), Rio de Janeiro, Brazil,Feb. 4-6: 1997 pp. 119-129.

- This is a time Non-Time series dashboard created by pointing to the CS410 collection . It has the following panels (1)query panel (2)Hits (3)fulltextsearch panel. The use-cases slides will use these features for data analysis and search.

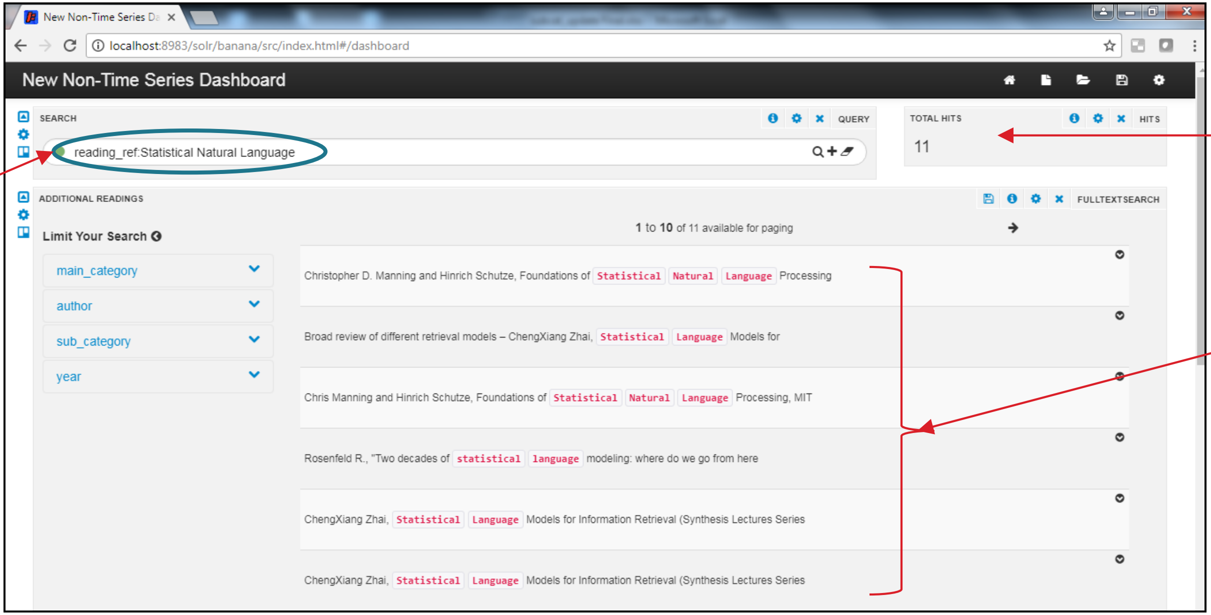
Usecase1- Keyword Search

Keyword Search

- The query widget can be used to do Keyword Search within the collection.
- Example shown is search for the term "Statistical Natural Language" and the hits are highlighted.

- This easily helps to maintain and retrieve the additional reading references mentioned in the lecture slides .

- There are duplicate references of the same material /inconsistent naming of authors etc. but these are attributed to the source data in the lecture slides. This can be cleaned as desired.



New Non-Time Series D
localhost:8983/solr/banana/src/index.html#/dashboard
New Non-Time Series Dashboard
SEARCH
QUERY
TOTAL HITS
HITS
reading_ref:Statistical Natural Language
11
ADDITIONAL READINGS
FULLTEXTSEARCH
Limit Your Search
main_category
author
sub_category
year
1 to 10 of 11 available for paging

Christopher D. Manning and Hinrich Schutze, Foundations of Statistical Natural Language Processing

Broad review of different retrieval models – ChengXiang Zhai, Statistical Language Models for

Chris Manning and Hinrich Schutze, Foundations of Statistical Natural Language Processing, MIT

Rosenfeld R., "Two decades of statistical language modeling: where do we go from here

ChengXiang Zhai, Statistical Language Models for Information Retrieval (Synthesis Lectures Series

ChengXiang Zhai, Statistical Language Models for Information Retrieval (Synthesis Lectures Series

Total Hits

Keyword Search

Return Results for the field Reading references.

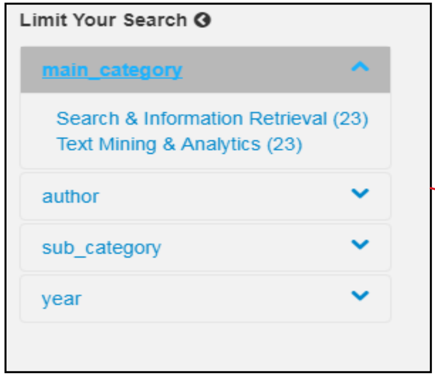
Usecase2- Applying facets for various Analysis

Faceted/Filtered Search Navigation.

- The filters implemented enable to analyze various use-cases like
  - List of authors/references being called out in the lecture slides
  - Year which the references were published.
  - Subcategory of topics covered.

- There are duplicate references of the same material /inconsistent naming of authors etc. but these are attributed to the source data in the lecture slides. This can be cleaned as desired.

Facets & Facet counts available for drill down search.
Click the respective value to narrow down search.



Limit Your Search

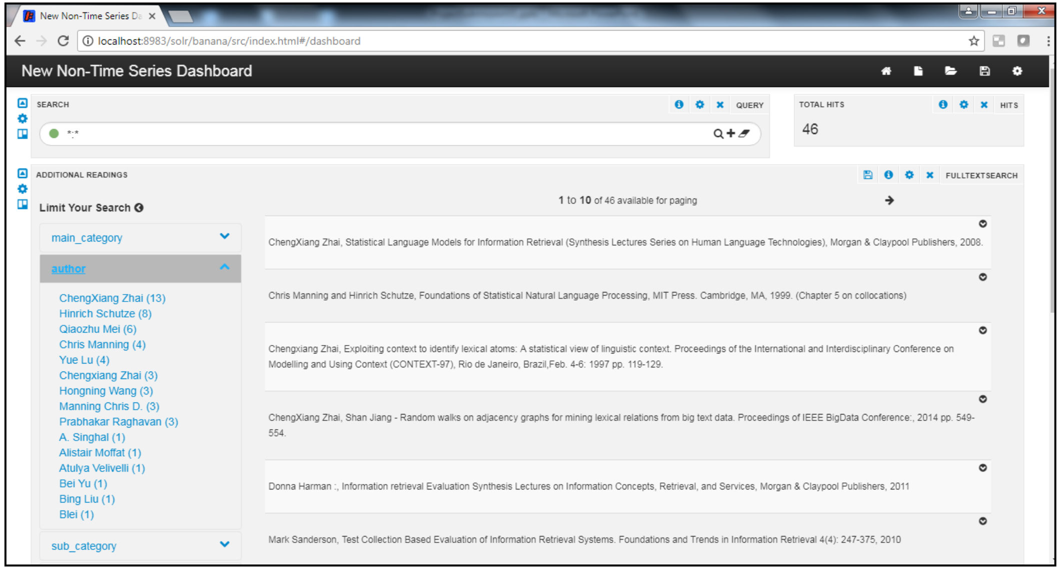
main_category

Search & Information Retrieval (23)
Text Mining & Analytics (23)

author

sub_category

year



New Non-Time Series Da

localhost:8983/solr/banana/src/index.html#/dashboard

New Non-Time Series Dashboard

SEARCH — QUERY

:

TOTAL HITS — HITS

46

ADDITIONAL READINGS — FULLTEXTSEARCH

Limit Your Search

1 to 10 of 46 available for paging

main_category

author

ChengXiang Zhai (13)
Hinrich Schutze (8)
Qiaozhu Mei (6)
Chris Manning (4)
Yue Lu (4)
Chengxiang Zhai (3)
Hongning Wang (3)
Manning Chris D. (3)
Prabhakar Raghavan (3)
A. Singhal (1)
Alistair Moffat (1)
Atulya Velivelli (1)
Bei Yu (1)
Bing Liu (1)
Blei (1)

sub_category

ChengXiang Zhai, Statistical Language Models for Information Retrieval (Synthesis Lectures Series on Human Language Technologies), Morgan & Claypool Publishers, 2008.

Chris Manning and Hinrich Schutze, Foundations of Statistical Natural Language Processing, MIT Press. Cambridge, MA, 1999. (Chapter 5 on collocations)

Chengxiang Zhai, Exploiting context to identify lexical atoms: A statistical view of linguistic context. Proceedings of the International and Interdisciplinary Conference on Modelling and Using Context (CONTEXT-97), Rio de Janeiro, Brazil,Feb. 4-6: 1997 pp. 119-129.

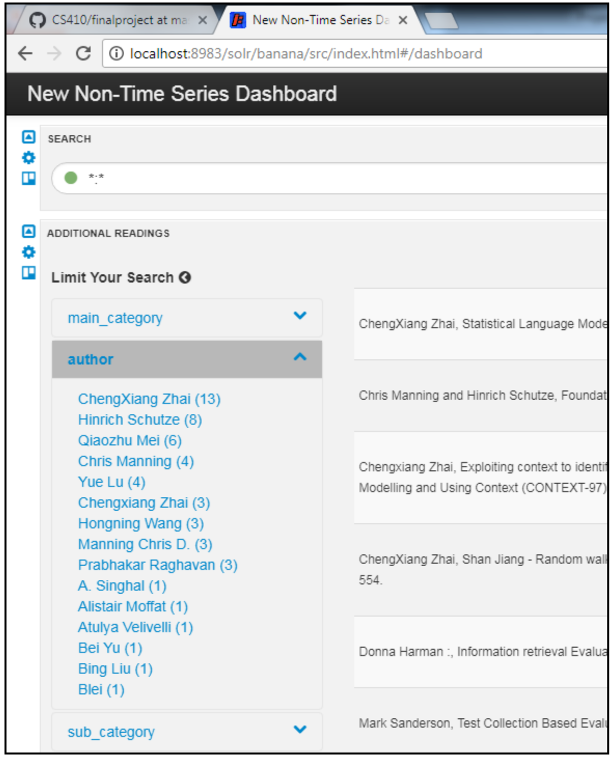
ChengXiang Zhai, Shan Jiang - Random walks on adjacency graphs for mining lexical relations from big text data. Proceedings of IEEE BigData Conference:, 2014 pp. 549-554.

Donna Harman :, Information retrieval Evaluation Synthesis Lectures on Information Concepts, Retrieval, and Services, Morgan & Claypool Publishers, 2011

Mark Sanderson, Test Collection Based Evaluation of Information Retrieval Systems. Foundations and Trends in Information Retrieval 4(4): 247-375, 2010

Usecase2- Facets Available.

Filter/Count by field Authors.


CS410/finalproject at ma × | New Non-Time Series Da ×

localhost:8983/solr/banana/src/index.html#/dashboard

New Non-Time Series Dashboard

SEARCH

:

ADDITIONAL READINGS

Limit Your Search

main_category

author

ChengXiang Zhai (13)
Hinrich Schutze (8)
Qiaozhu Mei (6)
Chris Manning (4)
Yue Lu (4)
Chengxiang Zhai (3)
Hongning Wang (3)
Manning Chris D. (3)
Prabhakar Raghavan (3)
A. Singhal (1)
Alistair Moffat (1)
Atulya Velivelli (1)
Bei Yu (1)
Bing Liu (1)
Blei (1)

sub_category

ChengXiang Zhai, Statistical Language Mode

Chris Manning and Hinrich Schutze, Foundat

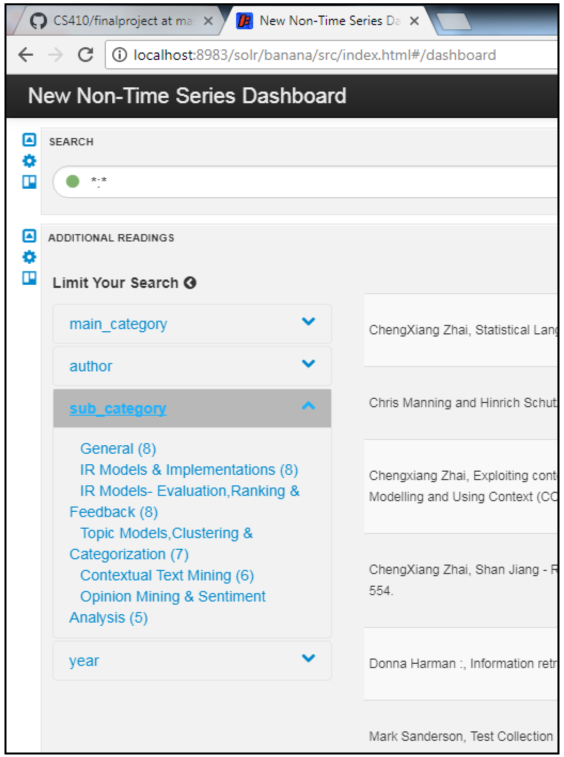
Chengxiang Zhai, Exploiting context to identi
Modelling and Using Context (CONTEXT-97)

ChengXiang Zhai, Shan Jiang - Random wal
554.

Donna Harman :, Information retrieval Evalua

Mark Sanderson, Test Collection Based Evalu

Filter/Count by sub_category


CS410/finalproject at ma × | New Non-Time Series Da ×

localhost:8983/solr/banana/src/index.html#/dashboard

New Non-Time Series Dashboard

SEARCH

:

ADDITIONAL READINGS

Limit Your Search

main_category

author

sub_category

General (8)
IR Models & Implementations (8)
IR Models- Evaluation,Ranking & Feedback (8)
Topic Models,Clustering & Categorization (7)
Contextual Text Mining (6)
Opinion Mining & Sentiment Analysis (5)

year

ChengXiang Zhai, Statistical Lang

Chris Manning and Hinrich Schut

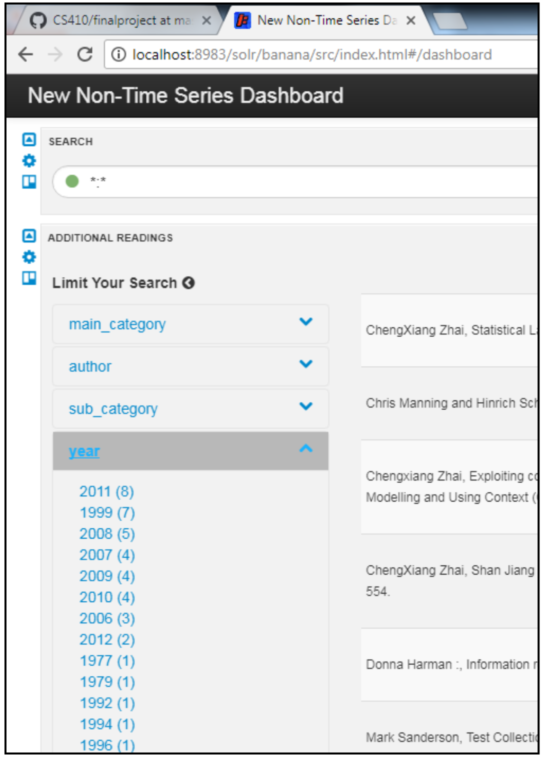
Chengxiang Zhai, Exploiting cont
Modelling and Using Context (CC

ChengXiang Zhai, Shan Jiang - F
554.

Donna Harman :, Information retr

Mark Sanderson, Test Collection

Filter/Count by year


CS410/finalproject at ma × | New Non-Time Series Da ×

localhost:8983/solr/banana/src/index.html#/dashboard

New Non-Time Series Dashboard

SEARCH

:

ADDITIONAL READINGS

Limit Your Search

main_category

author

sub_category

year

2011 (8)
1999 (7)
2008 (5)
2007 (4)
2009 (4)
2010 (4)
2006 (3)
2012 (2)
1977 (1)
1979 (1)
1992 (1)
1994 (1)
1996 (1)

ChengXiang Zhai, Statistical L

Chris Manning and Hinrich Sch

Chengxiang Zhai, Exploiting c
Modelling and Using Context (

ChengXiang Zhai, Shan Jiang
554.

Donna Harman :, Information r

Mark Sanderson, Test Collecti

References

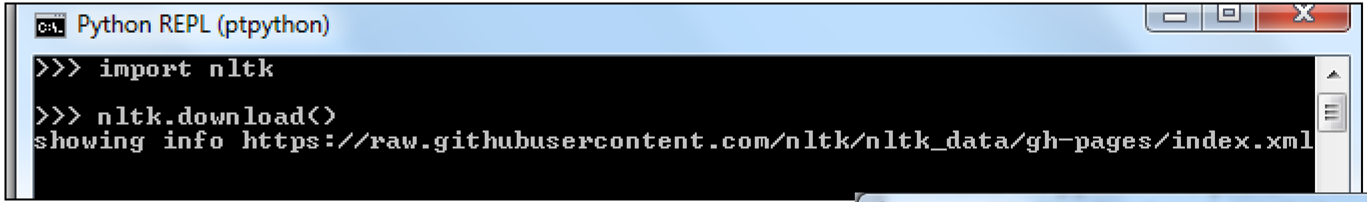
- NER tagger -https://nlp.stanford.edu/software/CRF-NER.shtml
- NLTK - http://www.nltk.org/book/
- NLTK http://www.nltk.org/howto/corpus.html
- PDFMiner related -https://stackoverflow.com/questions/26413216/pdfminer3k-has-no-method-named-create-pages-in-pdfpage
- For Data cleaning (trail and error) fine tuning the data for the NER tagger http://nlp.stanford.edu:8080/ner/
- Solr Configuration guides and https://lucene.apache.org/solr/guide/6_6/uploading-data-with-index-handlers.html for data indexing.
- http://yonik.com/ Solr 'n stuff for atomic updates formats
- https://doc.lucidworks.com/lucidworks-hdpsearch/2.5/Guide-Banana.html

Thankyou

Using nltk.download() to download packages

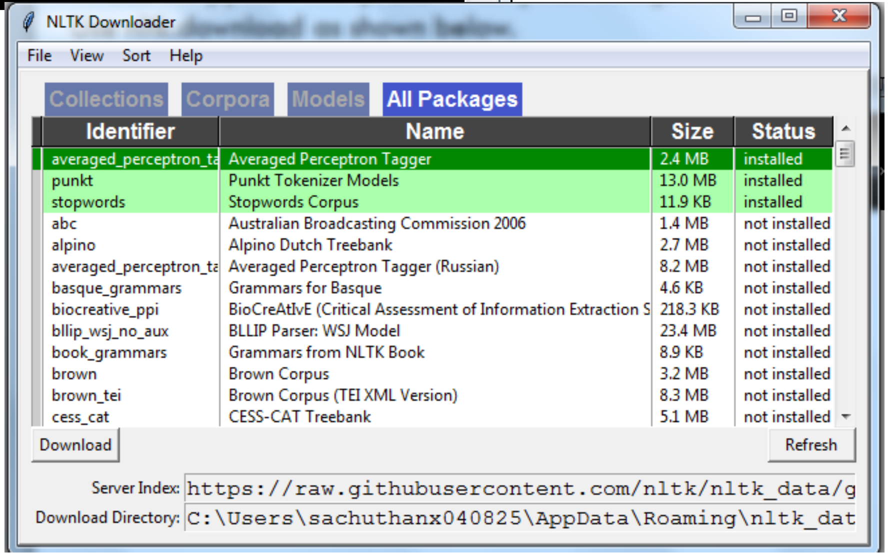
Invoke the python interpreter WinPython Interpreter.exe at C:\WinPython3530Qt5

Use nltk.download as shown below.



```
>>> import nltk

>>> nltk.download()
showing info https://raw.githubusercontent.com/nltk/nltk_data/gh-pages/index.xml
```

Download the packages highlighted as green.



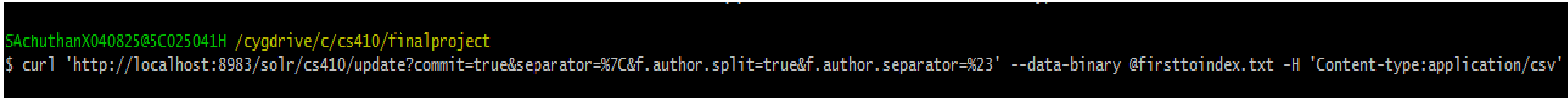
Identifier	Name	Size	Status
averaged_perceptron_ta	Averaged Perceptron Tagger	2.4 MB	installed
punkt	Punkt Tokenizer Models	13.0 MB	installed
stopwords	Stopwords Corpus	11.9 KB	installed
abc	Australian Broadcasting Commission 2006	1.4 MB	not installed
alpino	Alpino Dutch Treebank	2.7 MB	not installed
averaged_perceptron_ta	Averaged Perceptron Tagger (Russian)	8.2 MB	not installed
basque_grammars	Grammars for Basque	4.6 KB	not installed
biocreative_ppi	BioCreAtIvE (Critical Assessment of Information Extraction S	218.3 KB	not installed
bllip_wsj_no_aux	BLLIP Parser: WSJ Model	23.4 MB	not installed
book_grammars	Grammars from NLTK Book	8.9 KB	not installed
brown	Brown Corpus	3.2 MB	not installed
brown_tei	Brown Corpus (TEI XML Version)	8.3 MB	not installed
cess_cat	CESS-CAT Treebank	5.1 MB	not installed

Server Index: https://raw.githubusercontent.com/nltk/nltk_data/g

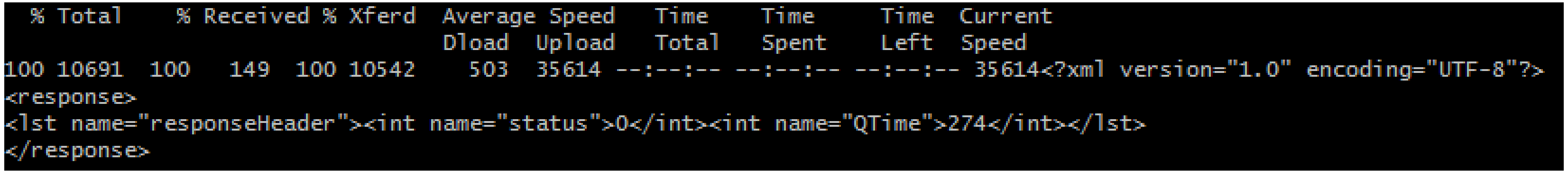
Download Directory: C:\Users\sachuthanx040825\AppData\Roaming\nltk_dat

Index Commands snapshots.

First time index into Solr – curl commands snapshots.

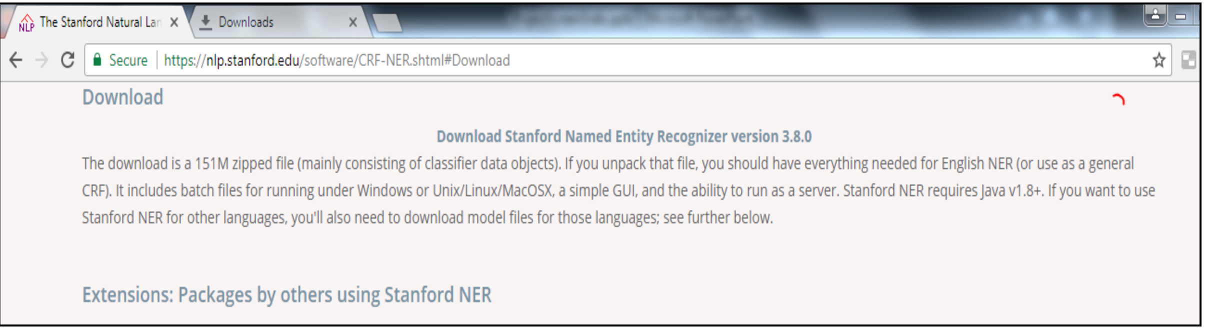


```
SAchuthanX040825@5C025041H /cygdrive/c/cs410/finalproject
$ curl 'http://localhost:8983/solr/cs410/update?commit=true&separator=%7C&f.author.split=true&f.author.separator=%23' --data-binary @firsttoindex.txt -H 'Content-type:application/csv'
```



```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 10691  100   149  100 10542    503  35614 --:--:-- --:--:-- --:--:-- 35614<?xml version="1.0" encoding="UTF-8"?>
<response>
<lst name="responseHeader"><int name="status">0</int><int name="QTime">274</int></lst>
</response>
```

Stanford NER Tagger & nltk.download.



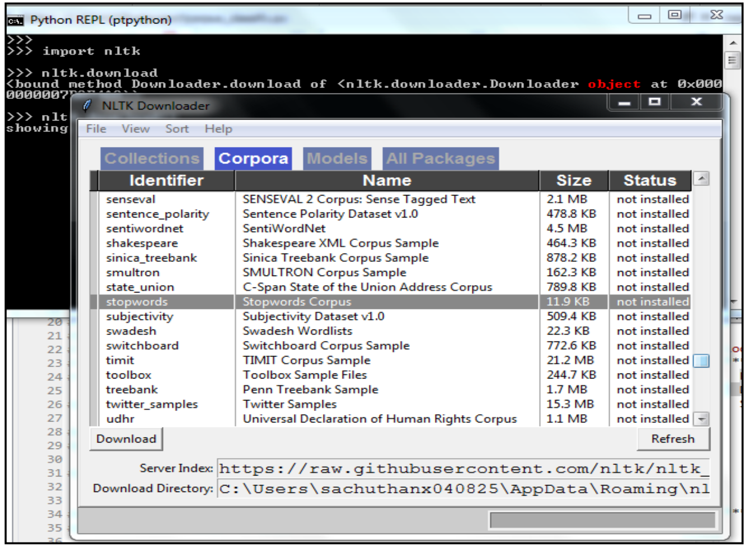
https://nlp.stanford.edu/software/CRF-NER.shtml#Download

Download

Download Stanford Named Entity Recognizer version 3.8.0

The download is a 151M zipped file (mainly consisting of classifier data objects). If you unpack that file, you should have everything needed for English NER (or use as a general CRF). It includes batch files for running under Windows or Unix/Linux/MacOSX, a simple GUI, and the ability to run as a server. Stanford NER requires Java v1.8+. If you want to use Stanford NER for other languages, you'll also need to download model files for those languages; see further below.

Extensions: Packages by others using Stanford NER



```
>>>
>>> import nltk

>>> nltk.download
<bound method Downloader.download of <nltk.downloader.Downloader object at 0x000
0000007

>>> nlt
showing
```

NLTK Downloader

Identifier	Name	Size	Status
senseval	SENSEVAL 2 Corpus: Sense Tagged Text	2.1 MB	not installed
sentence_polarity	Sentence Polarity Dataset v1.0	478.8 KB	not installed
sentiwordnet	SentiWordNet	4.5 MB	not installed
shakespeare	Shakespeare XML Corpus Sample	464.3 KB	not installed
sinica_treebank	Sinica Treebank Corpus Sample	878.2 KB	not installed
smultron	SMULTRON Corpus Sample	162.3 KB	not installed
state_union	C-Span State of the Union Address Corpus	789.8 KB	not installed
stopwords	Stopwords Corpus	11.9 KB	not installed
subjectivity	Subjectivity Dataset v1.0	509.4 KB	not installed
swadesh	Swadesh Wordlists	22.3 KB	not installed
switchboard	Switchboard Corpus Sample	772.6 KB	not installed
timit	TIMIT Corpus Sample	21.2 MB	not installed
toolbox	Toolbox Sample Files	244.7 KB	not installed
treebank	Penn Treebank Sample	1.7 MB	not installed
twitter_samples	Twitter Samples	15.3 MB	not installed
udhr	Universal Declaration of Human Rights Corpus	1.1 MB	not installed

Server Index: https://raw.githubusercontent.com/nltk/nltk_

Download Directory: C:\Users\sachuthanx040825\AppData\Roaming\nl