My Project

Generated by Doxygen 1.8.11

Contents

Chapter 1

Voice Enabled Natural Language Search Engine for Yelp

Dataset

This system runs on Yelp Dataset Challenge dataset (https://www.yelp.com/dataset_challenge). Download the dataset and extract the files to a folder.

Packages/tools needed to run the system:

The following packages needs to be install

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 4

Class Index

4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

Chapter 6

Namespace Documentation

6.1 distance Namespace Reference

Functions

- def toRad (degree)
- def distance (p1, p2)

6.1.1 Detailed Description

```
calculate the distance between two lat lng points in miles
point: (lat, lng)

p1 = (42.280826, -83.743038) # ann arbor
p2 = (41.878114, -87.629798) # chicago
print distance(p1, p2)
```

6.1.2 Function Documentation

6.1.2.1 def distance.distance (*p1, p2*)

6.1.2.2 def distance.toRad (*degree*)

```
convert degree to radian
```

6.2 flask_jqry Namespace Reference

Functions

- def main ()
- def view_do_something ()
- def process (q)

Variables

- app = Flask(__name__)
- searcher = None

6.2.1 Function Documentation

6.2.1.1 def flask_jqry.main ( )

```
This function is ued for display the webpage on the browser
```

6.2.1.2 def flask_jqry.process (*q*)

```
Generating parsed queries and doing search
```

6.2.1.3 def flask_jqry.view_do_something ( )

```
This function for handling the POST request sent from web page and call another python script to run for q
```

6.2.2 Variable Documentation

6.2.2.1 flask_jqry.app = Flask(__name__)

6.2.2.2 flask_jqry.searcher = None

6.3 indexer Namespace Reference

Classes

- class Indexer

6.4 query Namespace Reference

Functions

- def all_cities ()
- def extract_entity_names (t)
- def parse_location (original_setence)
- def parse_time (original_setence)
- def parse_filters (original_setence)
- def query (original)

Variables

- times = time.localtime()
- list nums
- dictionary Cities = {}
- dictionary States = {}
- dictionary States_abbr = {}
- float laititude = 36.169941
- float longitude = -115.139830

6.4.1 Detailed Description

```
@This file is used to parse the query into the multi filters and modify
the query sentense, so that the search engine is able to retrive a
list of business from dataset which meeting all requirements of query.

The filters include location limit, open time limit, happy hour, suitable
for group, suitable for kids, outdoor table availability, various price
level, take-out and delivery availability, reservation availability, and
parking availability

The words shows in the filter will be removed from query string so that
search engine is able to get a more fair and relevant ranking of different
business.

@ Input: the string of query sentense
@ Output: the imporved query sentense and a list of filters.

@ Package used: nltk, time, geonamescache
```

6.4.2 Function Documentation

6.4.2.1 def query.all_cities ( )

```
This function will set three dictionary of python.

The package geonamescache contians a list of cities in the World and
a list of states in Unite States
After scan the list of cities and the list of states, the name of cities
and the name of states of America with all lower-case will be loaded into
dictionary Cities and States separately.
```

6.4.2.2 def query.extract_entity_names (*t*)

```
This function is a helper function, which will be recursively called to
separate one sentense into various words group. And from the groups of
word, the city name and state name can be extracted as filter
```

6.4.2.3 def query.parse_filters (*original_setence*)

```
This funciton will extract specified attribute of business
listed in dataset, like happy hour, suitable for group,
suitable for kids, outdoor table availability, various price
level, take-out and delivery availability, reservation
availability, and parking availability. The filters translated
from the query will locat the specified business meeting all
requirement
```

6.4.2.4 def query.parse_location (*original_setence*)

```
This funciton will extract location information from query sentence,
including identified city name, state name, and the distance limit
from the current location. If the query try to find near business, the
default distance limit is 5 miles.
```

6.4.2.5 def query.parse_time (*original_setence*)

```
This funciton will extract time information from query sentence,
including identified weekdays and time period like morning and
afternoon. The query can alse specify exact time like 5 o'clock
and 3 pm.
```

6.4.2.6 def query.query (*original*)

```
This is main function for parsing query. The input
is the query sentence and the outputs are modified
query string and all filters
```

6.4.3 Variable Documentation

6.4.3.1 dictionary query.Cities = {}

6.4.3.2 float query.laititude = 36.169941

6.4.3.3 float query.longitude = -115.139830

6.4.3.4 list query.nums

Initial value:

```
1 = ['one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight', \
2         'nigh', 'ten', 'eleven', 'twelve', 'thirtenn', 'fourteen', 'fifteen', \
3         'sixteen', 'seventeen', 'eighteen', 'ninteen', 'twenty']
```

6.4.3.5 dictionary query.States = {}

6.4.3.6 dictionary query.States_abbr = {}

6.4.3.7 query.times = time.localtime()

6.5 query_speechrecognition Namespace Reference

Functions

- def all_cities ()
- def get_audio_query ()
- def extract_entity_names (t)
- def parse_location (original_setence)
- def parse_time (original_setence)
- def parse_filters (original_setence)
- def query_speech ()

Variables

- times = time.localtime()
- list nums = ['one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight', 'nigh', 'ten', 'eleven', 'twelve', 'thirtenn', 'fourteen', 'fifteen', 'sixteen', 'seventeen', 'eighteen', 'ninteen', 'twenty']
- dictionary Cities = {}
- dictionary States = {}
- dictionary States_abbr = {}
- float laititude = 36.169941
- float longitude = -115.139830

6.5.1 Detailed Description

```
This file is very similar to query.py. The only difference is that
text input of the query sentense becomes audio input
This file is used to parse the query into the multi filters and modify
the query sentense, so that the search engine is able to retrive a
list of business from dataset which meeting all requirements of query.

The filters include location limit, open time limit, happy hour, suitable
for group, suitable for kids, outdoor table availability, various price
level, take-out and delivery availability, reservation availability, and
parking availability

The words shows in the filter will be removed from query string so that
search engine is able to get a more fair and relevant ranking of different
business.

@ Input: the input would be audio
@ Output: the imporved query sentense and a list of filters.

@ Package used: nltk, time, geonamescache, speech_recognition
```

6.5.2 Function Documentation

6.5.2.1 def query_speechrecognition.all_cities ( )

```
This function will set three dictionary of python.

The package geonamescache contians a list of cities in the World and
a list of states in Unite States
After scan the list of cities and the list of states, the name of cities
and the name of states of America with all lower-case will be loaded into
dictionary Cities and States separately.
```

6.5.2.2 def query_speechrecognition.extract_entity_names (*t*)

```
This function is a helper function, which will be recursively called to
separate one sentense into various words group. And from the groups of
word, the city name and state name can be extracted as filter
```

6.5.2.3 def query_speechrecognition.get_audio_query ( )

```
This function wiil open the microphone and record the speech. Then
the speech will be translated into text by google speedch recognizition.
```

6.5.2.4 def query_speechrecognition.parse_filters (*original_setence*)

```
This funciton will extract specified attribute of business
listed in dataset, like happy hour, suitable for group,
suitable for kids, outdoor table availability, various price
level, take-out and delivery availability, reservation
availability, and parking availability. The filters translated
from the query will locat the specified business meeting all
requirement
```

6.5.2.5 def query_speechrecognition.parse_location (*original_setence*)

```
This funciton will extract location information from query sentence,
including identified city name, state name, and the distance limit
from the current location. If the query try to find near business, the
default distance limit is 5 miles.
```

6.5.2.6 def query_speechrecognition.parse_time (*original_setence*)

```
This funciton will extract time information from query sentence,
including identified weekdays and time period like morning and
afternoon. The query can alse specify exact time like 5 o'clock
and 3 pm.
```

6.5.2.7 def query_speechrecognition.query_speech ( )

```
This is main function for parsing query. The input
is the query sentence and the outputs are modified
query string and all filters
```

6.5.3 Variable Documentation

6.5.3.1 dictionary query_speechrecognition.Cities = {}

6.5.3.2 float query_speechrecognition.laititude = 36.169941

6.5.3.3 float query_speechrecognition.longitude = -115.139830

6.5.3.4 list query_speechrecognition.nums = ['one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight', 'nigh', 'ten', 'eleven', 'twelve', 'thirtenn', 'fourteen', 'fifteen', 'sixteen', 'seventeen', 'eighteen', 'ninteen', 'twenty']

6.5.3.5 dictionary query_speechrecognition.States = {}

6.5.3.6 dictionary query_speechrecognition.States_abbr = {}

6.5.3.7 query_speechrecognition.times = time.localtime()

6.6 searcher Namespace Reference

Classes

- class Searcher

6.6.1 Detailed Description

```
This class is used to perform query searches.
When a Searcher object is created, it loads the preprocessed data from disk to memory.
A query can be performed by calling function perform_query. The function will return list of search
results ranked by ranking score.
```

6.7 test_indexer Namespace Reference

Functions

- def main ()

6.7.1 Detailed Description

```
This script is used to index the yelp dataset.
Please use the following command to start this script:
    python test_indexer.py <path_to_yelp_dataset_folder> data.json
The inverted index will be saved to ./index/ folder, refer to indexer.py for more information.
The preprocessed data will be saved to file ./data.json.
```

6.7.2 Function Documentation

6.7.2.1 def test_indexer.main ( )

6.8 test_speech Namespace Reference

Functions

- def main ()

6.8.1 Detailed Description

```
This is to test the speech recognition functionality.
```

6.8.2 Function Documentation

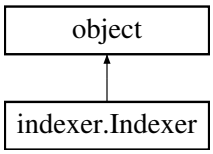
6.8.2.1 def test_speech.main ( )

Chapter 7

Class Documentation

7.1 indexer.Indexer Class Reference

Inheritance diagram for indexer.Indexer:



Public Member Functions

- def __init__ (self, source_path, destination=None)
- def preprocess (self)
- def index (self)

Public Attributes

- source_path
- destination
- business
- review
- tip
- data

7.1.1 Detailed Description

```
This class is used to preprocess the yelp dataset and index the preprocessed data.
The inverted index will be saved on disk.
```

7.1.2 Constructor & Destructor Documentation

7.1.2.1 def indexer.Indexer.__init__ (*self, source_path, destination* = None)

```
source_path: the path to the yelp dataset folder
destination: the destination file where to save the preprocessed data
The business, review and tip data from the Yelp dataset are processed and indexed.
The inverted index will be saved to folder ./index/
The preprocessed data will be saved to "destination" which can be read by searcher later on.
```

7.1.3 Member Function Documentation

7.1.3.1 def indexer.Indexer.index (*self*)

```
This function is used to index the preprocessed data.
The inverted index will be saved to ./index/ folder
business_id, name, address, categories, review and tip data are indexed.
```

7.1.3.2 def indexer.Indexer.preprocess (*self*)

```
This function is used to preprocess the yelp dataset.
Data from the business, review and tip file will be combined,
to form a single entity (a dict with business_id as key) that represents all the businesses.
```

7.1.4 Member Data Documentation

7.1.4.1 indexer.Indexer.business

7.1.4.2 indexer.Indexer.data

7.1.4.3 indexer.Indexer.destination

7.1.4.4 indexer.Indexer.review

7.1.4.5 indexer.Indexer.source_path

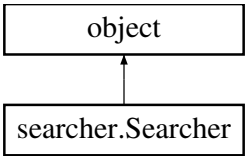
7.1.4.6 indexer.Indexer.tip

The documentation for this class was generated from the following file:

- /Users/Federer-HYJ/Documents/EECS549/final project/YelpNLSearch/indexer.py

7.2 searcher.Searcher Class Reference

Inheritance diagram for searcher.Searcher:



Public Member Functions

- def __init__ (self, path)
- def process_query (self, query, filters)
- def searching (self, query)
- def filtering (self, search_results, filters)
- def check_distance (self, id, value)
- def check_parking (self, id)
- def check_hours (self, id, time)
- def hour_to_number (self, hours)

Public Attributes

- data
- analyzer
- reader
- searcher

7.2.1 Constructor & Destructor Documentation

7.2.1.1 def searcher.Searcher.__init__ (*self, path*)

7.2.2 Member Function Documentation

7.2.2.1 def searcher.Searcher.check_distance (*self, id, value*)

```
check if a business resides in the requested distance radius
```

7.2.2.2 def searcher.Searcher.check_hours (*self, id, time*)

```
function to check if a business opens at a given time
```

7.2.2.3 def searcher.Searcher.check_parking (*self, id*)

```
function to check if a business has parking
```

7.2.2.4 def searcher.Searcher.filtering (*self, search_results, filters*)

```
Function to perform filtering.
```

7.2.2.5 def searcher.Searcher.hour_to_number (*self, hours*)

7.2.2.6 def searcher.Searcher.process_query (*self, query, filters*)

```
Function to process a query.
There are two steps: the first step uses the query to find relevant results ranked;
the second step use the filter to eliminate items that do not match the filter.
```

7.2.2.7 def searcher.Searcher.searching (*self, query*)

```
Function to perform the search. Results will be returned based on relevance.
```

7.2.3 Member Data Documentation

7.2.3.1 searcher.Searcher.analyzer

7.2.3.2 searcher.Searcher.data

7.2.3.3 searcher.Searcher.reader

7.2.3.4 searcher.Searcher.searcher

The documentation for this class was generated from the following file:

- /Users/Federer-HYJ/Documents/EECS549/final project/YelpNLSearch/searcher.py

Chapter 8

File Documentation

8.1 /Users/Federer-HYJ/Documents/EECS549/final project/YelpNLSearch/distance.py File Reference

Namespaces

- distance

Functions

- def distance.toRad (degree)
- def distance.distance (p1, p2)

8.2 /Users/Federer-HYJ/Documents/EECS549/final project/YelpNLSearch/flask_jqry.py File Reference

Namespaces

- flask_jqry

Functions

- def flask_jqry.main ()
- def flask_jqry.view_do_something ()
- def flask_jqry.process (q)

Variables

- flask_jqry.app = Flask(__name__)
- flask_jqry.searcher = None

8.3 /Users/Federer-HYJ/Documents/EECS549/final project/YelpNLSearch/indexer.py File Reference

Classes

- class indexer.Indexer

Namespaces

- indexer

8.4 /Users/Federer-HYJ/Documents/EECS549/final project/YelpNLSearch/query.py File Reference

Namespaces

- query

Functions

- def query.all_cities ()
- def query.extract_entity_names (t)
- def query.parse_location (original_setence)
- def query.parse_time (original_setence)
- def query.parse_filters (original_setence)
- def query.query (original)

Variables

- query.times = time.localtime()
- list query.nums
- dictionary query.Cities = {}
- dictionary query.States = {}
- dictionary query.States_abbr = {}
- float query.laititude = 36.169941
- float query.longitude = -115.139830

8.5 /Users/Federer-HYJ/Documents/EECS549/final project/YelpNLSearch/query_speechrecognition.py File Reference

Namespaces

- query_speechrecognition

Functions

- def query_speechrecognition.all_cities ()
- def query_speechrecognition.get_audio_query ()
- def query_speechrecognition.extract_entity_names (t)
- def query_speechrecognition.parse_location (original_setence)
- def query_speechrecognition.parse_time (original_setence)
- def query_speechrecognition.parse_filters (original_setence)
- def query_speechrecognition.query_speech ()

Variables

- query_speechrecognition.times = time.localtime()
- list query_speechrecognition.nums = ['one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight', 'nigh', 'ten', 'eleven', 'twelve', 'thirtenn', 'fourteen', 'fifteen', 'sixteen', 'seventeen', 'eighteen', 'ninteen', 'twenty']
- dictionary query_speechrecognition.Cities = {}
- dictionary query_speechrecognition.States = {}
- dictionary query_speechrecognition.States_abbr = {}
- float query_speechrecognition.laititude = 36.169941
- float query_speechrecognition.longitude = -115.139830

8.6 /Users/Federer-HYJ/Documents/EECS549/final project/YelpNLSearch/README.md File Reference

8.7 /Users/Federer-HYJ/Documents/EECS549/final project/YelpNLSearch/searcher.py File Reference

Classes

- class searcher.Searcher

Namespaces

- searcher

8.8 /Users/Federer-HYJ/Documents/EECS549/final project/YelpNLSearch/static/js/bootstrap.min.↩ js File Reference

Functions

- if ("undefined"==typeof jQuery) throw new Error("Bootstrap requires jQuery")
- function b ()
- a fn a (function(){a.support.transition=b()})}(jQuery)
- a fn a fn alert a fn alert c prototype close (jQuery)
- c hasClass ("btn")||(c

Variables

- function a {"use strict"
- a fn emulateTransitionEnd =function(b){var c=!1,d=this;a(this).one(a.support.transition.end,function(){c=!0});var e=function(){c||a(d).trigger(a.support.transition.end)};return setTimeout(e,b),this}
- var b ='[data-dismiss="alert"]'
- var c =function(c){a(c).on("click",b,this.close)}
- c prototype close =function(b){function c(){f.trigger("closed.bs.alert").remove()}var d=a(this),e=d.attr("data-target");e||(e=d.attr("href"),e=e&&e.replace(/.*(?=#[^\s]*$)/,""));var f=a(e);b&&b.preventDefault(),f.length||(f=d.hasClass("alert")?d:d.parent()),f.trigger(b=a.Event("close.bs.alert")),b.isDefaultPrevented()||(f.removeClass("in"),a.support.transition&&f.hasClass("fade")?f.one(a.support.transition.end,c).emulateTransitionEnd(150):c())}
- var d =a.fn.alert
- a fn alert =function(b){return this.each(function(){var d=a(this),e=d.data("bs.alert");e||d.data("bs.alert",e=new c(this)),"string"==typeof b&&e[b].call(d)})}
- a fn a fn alert Constructor =c
- a fn a fn alert a fn alert noConflict =function(){return a.fn.alert=d,this}
- b DEFAULTS ={loadingText:"loading..."}
- b b prototype setState =function(a){var b="disabled",c=this.$element,d=c.is("input")?"val":"html",e=c.data();a+="Text",e.resetText||c.data("resetText",c[d]()),c[d](e[a]||this.options[a]),setTimeout(function(){"loadingText"==a?c.addClass(b).attr(b,b):c.removeClass(b).removeAttr(b)},0)}
- b b prototype b prototype toggle =function(){var a=this.$element.closest('[data-toggle="buttons"]'),b=!0;if(a.length){var c=this.$element.find("input");"radio"===c.prop("type")&&(c.prop("checked")&&this.$element.hasClass("active")?b=!1:a.find(".active").removeClass("active")),b&&c.prop("checked",!this.$element.hasClass("active")).trigger("change")}b&&this.$element.toggleClass("active")}
- a fn button =function(c){return this.each(function(){var d=a(this),e=d.data("bs.button"),f="object"==typeof c&&c;e||d.data("bs.button",e=new b(this,f)),"toggle"==c?e.toggle():c&&e.setState(c)})}

8.8.1 Function Documentation

8.8.1.1 a fn a (*function(){a.support.transition=b()}*)

8.8.1.2 function b ( )

8.8.1.3 a fn a fn alert a fn alert c prototype close (*jQuery*)

8.8.1.4 c hasClass ("btn")

8.8.1.5 if ("undefined" ==typeof jQuery) [new]

8.8.2 Variable Documentation

8.8.2.1 a fn a fn button a fn button a {"use strict"

8.8.2.2 a fn alert =function(b){return this.each(function(){var d=a(this),e=d.data("bs.alert");e||d.data("bs.alert",e=new c(this)),"string"==typeof b&&e[b].call(d)})}

8.8.2.3 function b ='[data-dismiss="alert"]'

8.8.2.4 a fn button =function(c){return this.each(function(){var d=a(this),e=d.data("bs.button"),f="object"==typeof c&&c;e||d.data("bs.button",e=new b(this,f)),"toggle"==c?e.toggle():c&&e.setState(c)})}

8.8.2.5 var c =function(c){a(c).on("click",b,this.close)}

8.8.2.6 c prototype close =function(b){function c(){f.trigger("closed.bs.alert").remove()}var d=a(this),e=d.attr("data-target");e||(e=d.attr("href"),e=e&&e.replace(/.*(?=#[^\s]*$)/,""));var f=a(e);b&&b.preventDefault(),f.length||(f=d.hasClass("alert")?d:d.parent()),f.trigger(b=a.Event("close.bs.alert")),b.isDefaultPrevented()||(f.removeClass("in"),a.support.transition&&f.hasClass("fade")?f.one(a.support.transition.end,c).emulateTransitionEnd(150):c())}

8.8.2.7 a fn a fn button Constructor =c

8.8.2.8 var d =a.fn.alert

8.8.2.9 b DEFAULTS ={loadingText:"loading..."}

8.8.2.10 a fn emulateTransitionEnd =function(b){var c=!1,d=this;a(this).one(a.support.transition.end,function(){c=!0});var e=function(){c||a(d).trigger(a.support.transition.end)};return setTimeout(e,b),this}

8.8.2.11 a fn a fn button a fn button noConflict =function(){return a.fn.alert=d,this}

8.8.2.12 b b prototype setState =function(a){var b="disabled",c=this.$element,d=c.is("input")?"val":"html",e=c.data();a+="Text",e.resetText||c.data("resetText",c[d]()),c[d](e[a]||this.options[a]),setTimeout(function(){"loadingText"==a?c.addClass(b).attr(b,b):c.removeClass(b).removeAttr(b)},0)}

8.8.2.13 b b prototype b prototype toggle =function(){var a=this.$element.closest('[data-toggle="buttons"]'),b=!0;if(a.length){var c=this.$element.find("input");"radio"===c.prop("type")&&(c.prop("checked")&&this.$element.hasClass("active")?b=!1:a.find(".active").removeClass("active")),b&&c.prop("checked",!this.$element.hasClass("active")).trigger("change")}b&&this.$element.toggleClass("active")}

8.9 /Users/Federer-HYJ/Documents/EECS549/final project/YelpNLSearch/test_indexer.py File Reference

Namespaces

- test_indexer

Functions

- def test_indexer.main ()

8.10 /Users/Federer-HYJ/Documents/EECS549/final project/YelpNLSearch/test_speech.py File Reference

Namespaces

- test_speech

Functions

- def test_speech.main ()

Index