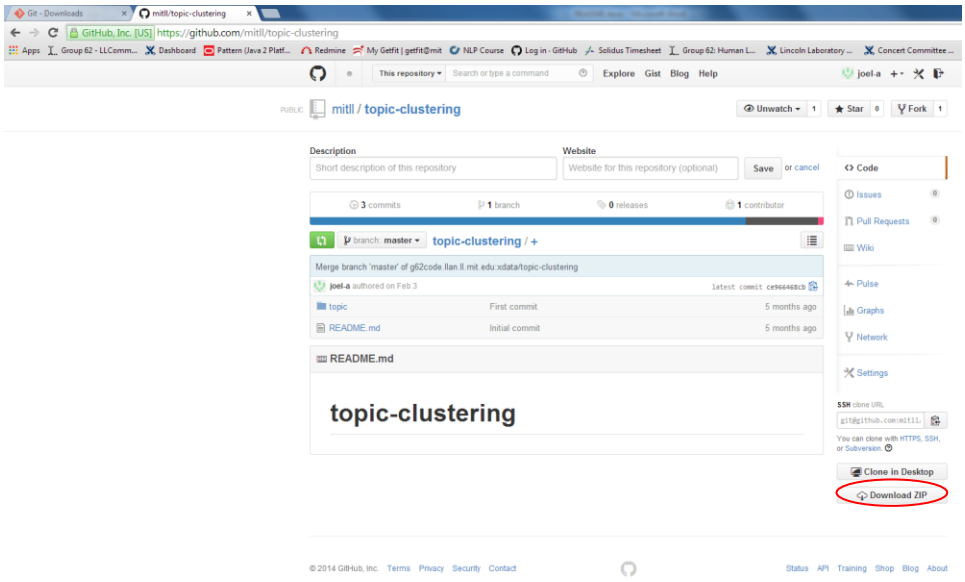
README

The Python program `run_all.py` performs state-of-the-art topic clustering on a set of text documents after filtering based on language identification. The number of topics extracted from the input documents is a parameter of the system (see section ***Options***). The results of both processing stages are saved to a set of files described in detail in the following sections.

Download / Installation steps

The code for our topic clustering application is publicly accessible through a GitHub repository on https://github.com. To download the code:

1. Navigate to https://github.com and create a user account. If you already have a github.com user account, skip to step 2.
2. You can download the code either as a zip file or a local Git repository:
    a. Zip file:
        i. Go to https://github.com/mitll/topic-clustering
        ii. Click on the ***Download ZIP*** button



    b. Git repository
        i. Install git (*http://git-scm.com/downloads*).
        ii. If this is your first time using GitHub, make sure to add your SSH key to GitHub For details see: *https://help.github.com/articles/generating-ssh-keys*
        iii. Open a git console and browse to the directory where you would like to save the code.
        iv. Run the following command:

```
git clone git@github.com:mitll/topic-clustering.git
```

Once you have downloaded the code, you can run the program by installing the dependencies listed in section ***Dependencies***, and following the steps described in section ***Running the Program***.

Dependencies

- Python 2.7
- NumPy

Target Platform

- Linux, 64 bits

Running the Program

The program

```
run_all.py [OPTIONS] INPUT_FILE OUTPUT_DIR
```

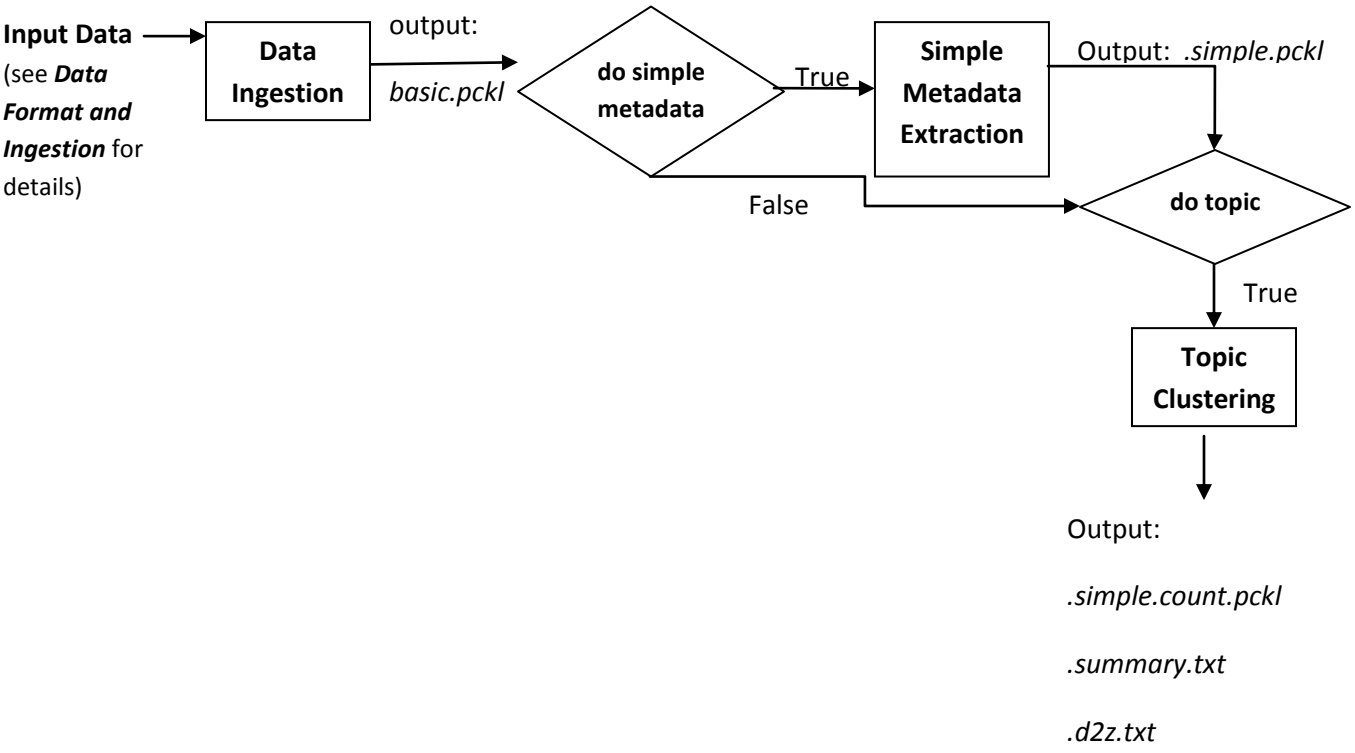
reads documents' text from INPUT_FILE, performs language id (LID) and topic clustering on the documents, and saves the results to OUTPUT_DIR. You can test the program by running the aforementioned command with the provided data file:

<*installation_dir*>/data_truncated/journalEntries.csv

Assuming you have created a directory named *output* under the installation directory, the appropriate command would be:

```
run_all.py data_truncated/journalEntries.csv output/
```

Language id and topic clustering are carried out in different but dependent steps. The following diagram shows the flow of execution, the parameters that control it, and the input/output to/of each step:



The input data consist of a text file containing a line for each document in your corpus. The section ***Data Format and Ingestion*** describes the format in which the input data should be for ingestion. The section ***Custom Data Reader*** explains how this step can be customized to fit your data format.

After reading the provided data, the program generates a serialized version of a Python dictionary object and saves it in the pickle file: *OUTPUT_DIR/<input_file_name> .basic.pckl* . This dictionary contains a key-value pair for each document in the corpus. The key is the document id provided and the corresponding value is a dictionary containing all the fields extracted during the ingestion step.

The option *–do_simple_metadata* controls whether the *Simple Metadata Extraction* step is carried out or not. The default behavior is to perform this step although this can be overridden by assigning a value of *False* to *–do_simple_metadata.*

The *Simple Metadata Extraction* step is a necessary step for *Topic Clustering* as it normalizes the text contained in every document. In addition, it performs language identification on each document for the purposes of selecting which documents will be processed by the *Topic Clustering* step (see section ***Options***, -lang_filter). Both pieces of information are stored in an augmented version of the *…basic.pckl* file. This file is named *<input_file_name>.simple.pckl* and it is saved in the user-specified output directory. In later runs, this step can be invoked by passing the *.basic.pckl* file as input as opposed to your "raw" data file; thus avoiding the overhead of performing the ingestion step.

If the *–do_topic* is unspecified or set to *True*, the *Topic Clustering* step is executed. This step requires either executing the *Simple Metadata Extraction* step in the same run or passing the *.simple.pckl* generated in a previous run as input to this program. This step generates two text files containing the results of clustering the documents based on automatically extracted topics. See section ***Output*** for details.

Options

1. *INPUT_FILE input file (see format in documentation)*
2. *OUTPUT_DIR output directory*
3. *-r (string, default: "default_ingestion")*
   Custom reader for the ingestion phase
4. *-do_simple_metadata (boolean: True|False, default: True)*
   Whether the simple metadata extraction step is performed or not
5. *-do_topic (boolean: True|False, default: True)*
   Whether the topic clustering step is performed or not
6. *-temp (string, default: temp/)*
   storage location for temp files
7. *-num_topics (int, default: 50)*
   number of topics used for clustering
8. *-stop_list (string, default:topic/data/stop_list_kiva.txt)*
   Terms to exclude from topic clustering
9. *-tf_cutoff (int, default: 3)*
   Exclude terms that occur this number of times or fewer
10. *-df_cutoff (float, default: 0.25)*
    Exclude terms that happen in greater than this fraction of vectors
11. *–lang_filter (string, default: "en")*
    Process only those documents in the specified language for purposes of topic clustering. The default is English, i.e. only English text will be used for topic clustering. Language identification is performed using the langid.py module. Here are the ISO 639-1 codes for the 97 languages it can identify and therefore the values this parameter accepts:

    af, am, an, ar, as, az, be, bg, bn, br, bs, ca, cs, cy, da, de, dz, el, en, eo, es, et, eu, fa, fi, fo, fr, ga, gl, gu, he, hi, hr, ht, hu, hy, id, is, it, ja, jv, ka, kk, km, kn, ko, ku, ky, la, lb, lo, lt, lv, mg, mk, ml, mn, mr, ms, mt, nb, ne, nl, nn, no, oc, or, pa, pl, ps, pt, qu, ro, ru, rw, se, si, sk, sl, sq, sr, sv, sw, ta, te, th, tl, tr, ug, uk, ur, vi, vo, wa, xh, zh, zu

Data Format and Ingestion

All data is assumed to be contained in a single tab-delimited UTF-8 text file. The system comes with a default data reader module: Ingest.default_ingestion. This module assumes the input file has one line per document in the following format:

<document_id>TAB<document_text>NEW_LINE

where:

document_id – is a unique identifier for the document that line represents

TAB – one tab character

document_text - the document's text; the contents to be processed

NEW_LINE - new line character

Custom Data Reader

If your data is contained in a UTF-8 text file where each line contains information about a document but does not conform to the format described in the previous section, you can provide code to extract the two necessary fields and include any additional ones in the output. To provide a custom data reader:

1. Create a module under *<installation_folder>/Ingest* . Let's call it *custom_ingestion.py*
2. Define a function called `get_fields(ln)`, where `ln` is a line in the input text file. This method should extract the two necessary fields, add them to a dictionary and return the dictionary. The key 'id' should be used for the document id and 'msg' for the document text. Any additional key-value pairs added would not be modified and will be included in the system's output. This could be useful if the output of this program will be further processed by other programs.
3. Execute the run_all.py script with the option –r and the name of your new module, i.e. :

```
run_all.py -r custom_ingestion INPUT_FILE OUTPUT_DIR
```

Output

If both the *Simple Metadata Extraction* and the *Topic Clustering* steps are executed, run_all.py generates 5 output files: 3 serialized (pickle) files and 2 human-readable text files. A description of the information contained in each file follows:

1. **<input_file_name>.basic.pckl**

   This is the serialized (pickle) representation of the python dictionary that contains the data fields extracted from the "raw" input file. Each key-value pair in the dictionary consists of the document id (key) paired with a dictionary (value) which represents all the data fields extracted from the text document. The latter dictionary contains at least two fields: 'id' (document id) and 'msg' (document text).

2. **<input_file_name>.simple.pckl**

   This pickle file is an augmented version of the dictionary stored in the *.basic.pckl* file. There are two additional fields for each document dictionary. The first one is the normalized text that will be used for topic clustering. This text is stored under the key 'msg_norm'. The second one is the result of performing automatic language id on the document's text and its key is 'lid_lui'.

3. **<input_file_name>.simple.counts.pckl**

This pickle file stores an augmented version of the dictionary stored in the *.simple.pckl* file. To each document dictionary, a field is added, keyed by the string 'counts'. This dictionary contains the term frequency vector extracted from the corresponding document; there is a key-value pair for each term and its frequency in that document, respectively.

4. **<input_file_name>.<num_topics>.summary.txt**

Human-readable file containing a description of each extracted topic.

5. **<input_file_name>.<num_topics>.d2z.txt**
   Matrix of the weights assigned to each topic per document. Each row represents a document and each column corresponds to one the topic of the same index as described in the *.summary.txt* file

Algorithm

Our topic clustering algorithm is based on a latent modeling technique called Probabilistic Latent Semantic Analysis (PLSA). For details please refer to the following book chapter:

Hazen, T. J., **Topic Identification**, Chapter 12 in Tur, G., De Mori, R. (Editors),*Spoken Language Understanding: Systems for Extracting Semantic Information from Speech*, John Wiley & Sons, West Sussex, United Kingdom, 2011.

To download a pdf version / Additional references:

http://www.ll.mit.edu/mission/cybersec/publications/publication-topics/topic-modeling-recognition.html