README

Overview

The ***MITLL Topic Clustering System*** performs state-of-the-art topic clustering (i.e. topic-based unsupervised grouping of documents) on a set of text documents after filtering based on language identification. The number of topics extracted from the input documents is a user-defined parameter. Results are stored in human/machine readable files that developers could use for integration with other applications such as filtering documents for a particular interest group, document organization, and exploration.

Provided a collection of text documents, the *MITLL Topic Clustering System*:

1. Normalizes the input text and removes non-informative terms
2. Performs language identification on each document
3. Filters out all documents not matching the user-specified language
4. Uses a latent modeling technique called Probabilistic Latent Semantic Analysis (PLSA) to perform a soft classification of the documents into topics by:
    a. Learning topic classes in an unsupervised fashion
    b. For each document, assigning a degree of membership to each of the learned topic classes
5. Stores the results in files that can be read by a human for data exploration, or a machine for integration with other applications.

For details on PLSA please refer to the ***References*** section.

This tool is a command-line application mainly suited for developers who would like to add topic-based features to their applications.

Download

The code for the MITLL Topic Clustering System is publicly accessible through a git repository at https://github.com/mitll/topic-clustering . For help on *github* and *git*, please refer to https://github.com and http://git-scm.com/ respectively.

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

The system is a command-line application for 64-bit Linux written in Python and C++. It requires the following dependencies:

- Python 2.7
- NumPy (Python module)

Target Platform

- 64-bit Linux

Running the Program

The Python 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 following the following command with the provided data:

```
./<installation_dir>/run_all.py –lang_filter "en" data_truncated/loanMetaData.csv output/
```

The value `"en"` for the `–lang_filter` option indicates that we only want to do topic identification on English documents.

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 of each step:

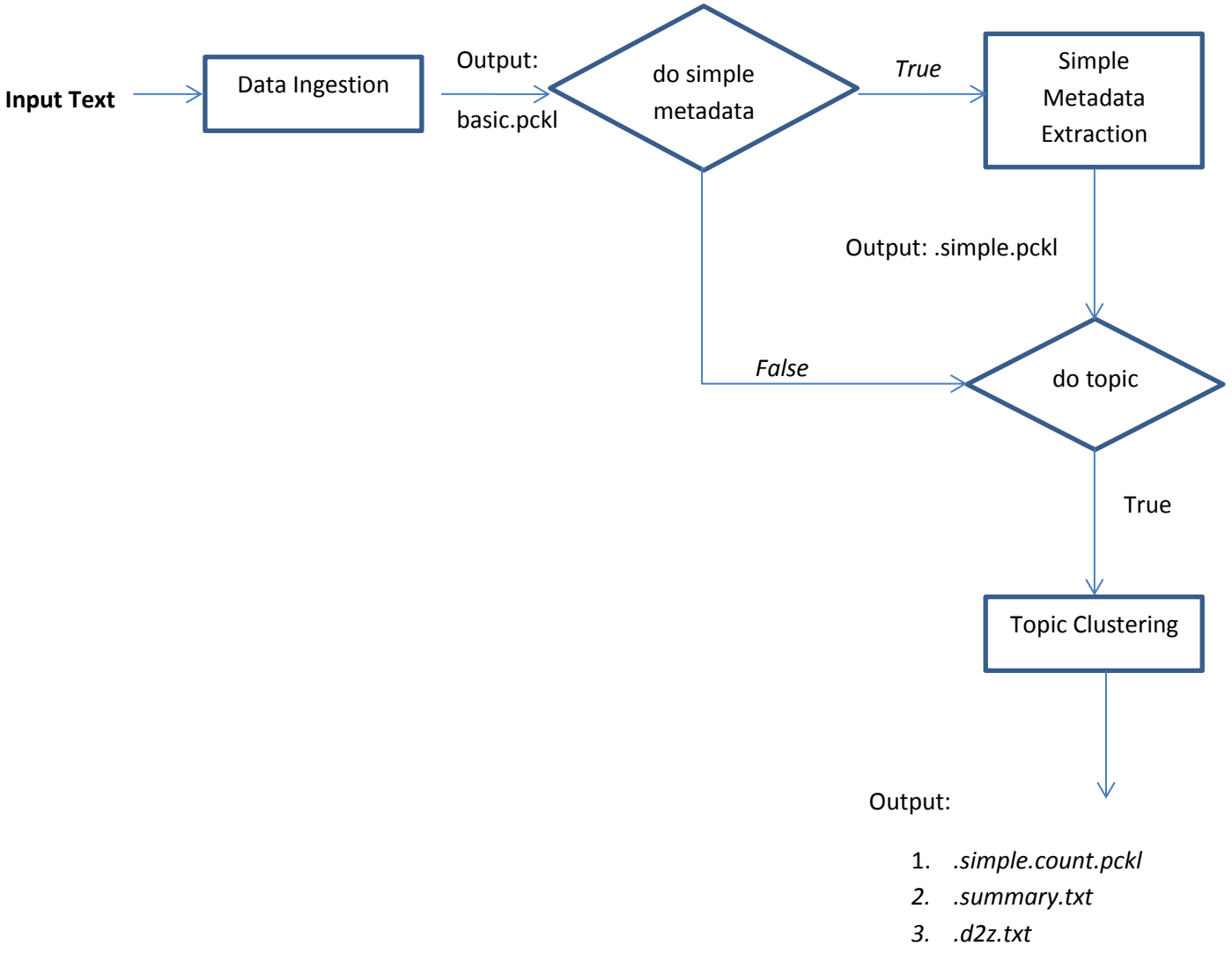


Figure 1: Processing stages and corresponding output.

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. Filtering by language is an optional feature which can be turned off by passing the string `"none"` to the –lang_filter option (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: "none")*

Process only those documents in the specified language for purposes of topic clustering. The default is "none", i.e. all documents 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. 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. Here is an excerpt of the output generating by running the provided sample data for 30 topics (note: output might differ from run to run due to the stochastic nature of the algorithm):

index	topic_score	doc_purity	percent_docs	summary0	summary1	summary2	summary3	summary4	summary5	summary6	summary7	summary8
0	2.95	0.533	5.53	months	seeking	project	active	budget	interest	rent	shs	benefit
1	2.77	0.483	5.74	group	members	lending	guarantee	repay	pressure	peer	repayment	solidarity
2	2.04	0.370	5.52	work	vegetables	fruits	life	hours	she's	grateful	ahead	day

Where:

index – is the topic identifier

topic_score - the latent topic quality score as measured in [2] (see section References).

doc_purity - purity measure (see reference [2]). Describes how semantically important a topic is. High purity describes topics that are semantically important and strongly spread among a small

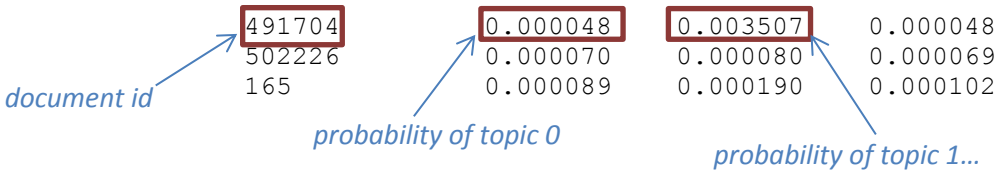
fraction of the corpus whereas low purity values indicate the topic is weakly spread across many documents.

percent_docs – the percentage of documents that belong to this topic.

summary 0….9 – highest-probability words in this topic. Used to characterize it and give an analyst an idea of what the topic is all about.

5. <input_file_name>.<num_topics>.d2z.txt

Matrix of the weights assigned to each topic per document. Each row represents a document. The first column is the document id and each additional column corresponds to a topic. The topics appeared ordered by index as described in the *.summary.txt* file . Here is an example output obtained by processing the provided sample data *(note: output differs from run to run due to the stochastic nature of the algorithm used)* :



References

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:

[1] 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

[2] Hazen, Timothy J. "Latent Topic Modeling for Audio Corpus Summarization."*INTERSPEECH*. 2011.