

Multi-way Distributional Clustering Toolkit, version 3.x

RON BEKKERMAN

Department of Computer Science
University of Massachusetts at Amherst

ronb@cs.umass.edu

May 13, 2007

Abstract

Multi-way Distributional Clustering (MDC) toolkit provides an efficient implementation of the MDC algorithm, proposed by Bekkerman et al. (2005). In MDC, data is represented as a set of *views*, or *modalities*, each of which is then clustered simultaneously with all the others. Examples of modalities can be: a set of documents, a set of their words, a set of names of their authors, a set of document titles etc. The original algorithm proposes clustering one modality *agglomeratively* (bottom-up), while clustering all the others *divisively* (top-down). The MDC toolkit however provides means for clustering of any configuration (agglomerative, divisive or flat clustering of any modality). Also, some of the modalities may not be clustered at all (see Bekkerman and Jeon, 2007, for details). The toolkit can be also used for semi-supervised clustering and transfer learning (see Bekkerman et al., 2006). The MDC toolkit is written in C++; it is an open source software, distributed without any warranty. It can be easily compiled under Unix/Linux. This document explains how to install and use the toolkit.

Note: starting from version 3.0, the interface of MDC has changed: command line options have been relocated into the initialization file. Please contact me if you have any questions, or would like to obtain an older version of the toolkit.

1 Installation and running

MDC can be downloaded from <http://www.cs.umass.edu/~ronb/code/MDC.tar.gz>. First use `gunzip` to unzip it, then `untar` it with `tar xvf`. To compile the tool under Unix/Linux, type `gmake` in the root directory of MDC. An executable called `mdc` will be built. To compile the tool in MS Visual Studio, open a new project and add all the files from the `source` directory of the toolkit distribution; define a macro `MS` in the `main.h` header file and then compile.¹ To run MDC, simply type `mdc` in a command line environment. Please contact me at ronb@cs.umass.edu if you experience any problem with downloading / installing / running the toolkit.

¹Note that the MS Windows version is currently unsupported.

2 Input

There are two types of input that must be provided to MDC to make it work properly. These are data files (contingency tables) and an MDC initialization file. The third type is optional: initialization files for clustered modalities. Note that MDC treats the modalities as random variables, therefore in this document, we will address modalities as *variables*.

2.1 Data files: contingency tables

To be processed by MDC, your data should be represented in the form contingency tables. An example can be a document collection represented as a table, where rows are documents, columns are distinct words, and each cell (i, j) is a count of word j in document i . Multiple tables can be given. Each table should be stored in a separate file. The naming convention of such files is `var1_var2.mdc`, where `var1` is the name of the first variable to be clustered (or in other words, of rows in the contingency table) and `var2` is the name of the second variable. In our document collection example, the contingency table file will be called `docs_words.mdc`. Any file with extension `.mdc` stored in the working directory will be considered as an input file for the MDC toolkit.

The format of each line of a contingency table file should be as follows:

```
value_of_var1 value_of_var2 count
```

where `count` is a non-empty cell of row `value_of_var1` and column `value_of_var2`. Floating point counts are acceptable as well as integers. An example of such a file would look like:

```
cat1/doc1 search 1
cat1/doc1 engine 3
cat1/doc2 classifier 2
cat2/doc1 engine 2
cat2/doc1 motorcycle 1
```

2.2 Initialization file

A file named `mdc.ini` should be created in the working directory. The file should contain two mandatory sections and one optional, as described below. Lines that start with the comment mark `//` will be ignored.

- **Variables section.** This section should start with the following line:

```
# Variables
```

Each consequent line in this section defines a variable (i.e. a modality). This line's format is:

```
var type final_num_of_clusters? final_iteration_trigger?
```

where

- `var` is the name of the variable, for example `docs` or `words`.
- `type` can be either `agglomerative`, `divisive`, or `flat`.

- `final_num_of_clusters` is an optional parameter (an integer) that specifies the number of clusters in the final clustering of this variable. For agglomerative and divisive variables, the clustering algorithm stops when this number of clusters is obtained, even if the clustering schedule (see below) has not been completed yet. If this parameter is omitted, the algorithm will stop when the schedule is completed, or when any other non-flat variable reaches its `final_num_of_clusters`. For flat variables, the final number of clusters equals the initial one, such that the number of clustering iterations is solely controlled by the clustering schedule. If omitted, the default number of clusters is a natural logarithm of the number of data points.
- `final_iteration_trigger` is an optional parameter that is only relevant for agglomerative and divisive variables (it will be ignored if the variable is flat). It specifies the number of clusters reaching which the algorithm switches to the “last iteration” mode (see Bekkerman et al., 2005, for details). By default, this number equals twice the `final_num_of_clusters` for agglomerative variables, and half the `final_num_of_clusters` for divisive variables. Note that it makes no sense to set `final_iteration_trigger` without having `final_num_of_clusters` set, and therefore such an option is not provided.

An example of the *Variables* section is:

```
# Variables
docs divisible 70 30
words agglomerative
```

If a variable definition is omitted for a particular variable, it is considered `divisive` by default. If definitions are omitted for all variables, MDC will print a warning because it is undesirable to have all divisive variables in the system (see Bekkerman et al., 2005, for discussion).

- **Schedule section.** Clustering schedule is the order of processing variables in an iterative clustering process. This section should start with the following line:

```
# Schedule
```

Each consequent line should contain a variable name. An example of this section is:

```
# Schedule
words
words
docs
words
docs
```

This means that two clustering iterations over `words` will be performed first, following by one iteration over `docs` and one more iteration over `words` and over `docs`.

- **Edge weights section.** Some interactions between variables can be considered more important than the others. MDC allows weighting interactions. This is an optional section that should start with:

```
# Edge weights
```

Each consequent line should be in the format of:

```
var1 var2 weight
```

where `weight` is a (floating point) weight of the interaction between variables `var1` and `var2` (see Bekkerman and Jeon, 2007, for an example of weighting).

If you wish a certain variable not to be clustered at all, you may mark it as `agglomerative` in the *Variables* section, and exclude this variable from the clustering *Schedule* section. Optionally, you can initiate this variable with any given clustering, using the `.ini` mechanism described below in Section 2.3.

2.3 Variable initializations (optional)

Before the clustering process starts, every variable is assigned an initial value (initial clustering). By default, for agglomerative variables, every data point is put into its own singleton cluster, while for divisive variables all the data points are put into one common cluster. For flat variables, each data point is assigned to one of the clusters uniformly at random.

However, the initialization procedure can be customized. For a variable `var1`, this is done by creating a file named `var1.ini` in the working directory, whose format is exactly as the format of the output files (see Section 3). After reading the contingency tables from `.mdc` files, for each variable in the system the MDC tool scans the working directory for the corresponding `.ini` file. Once the file is found, the variable is initialized with this file's content.

3 Output

After each clustering iteration, the resulting clusters are printed into a file. An example of the output file format is given below:

```
Cluster 0
  value1
  value2
  value3
Cluster 1
  value4
  value5
```

Note that all the values are prefixed with a tabulation.

An output file name is `var.num_of_iter`, where `var` is the name of the variable clustered at iteration `num_of_iter`. Iteration numeration starts with 0. At the last exhaustive iteration, an output is produced after each merge of two clusters (or a split of one cluster). At that iteration, the file name convention is different: it is `var_final.num_of_clust`, where `var` is again the name of the clustered variable and `num_of_clust` is the current number of clusters of variable `var`.

4 Scripts

Directory `scripts` of the MDC distribution contains a number of useful scripts:

- **build_input.pl** creates an input `.mdc` file out of a directory of labeled documents.
- **accuracy.pl** calculates the accuracy of a documents' clustering given their labels. The evaluation formula is given in Bekkerman et al. (2005).

- **run_mdc.pl** performs 10 independent runs of the MDC system.
- **average.pl** computes the accuracy of the system's performance averaged over the given number of independent runs.

5 Bug reports and other enquiries

The initial version of the MDC toolkit was written during Summer 2005. Since then, it has been constantly updated. Please let me know if you have any problem with it: ronb@cs.umass.edu.

References

- R. Bekkerman, R. El-Yaniv, and A. McCallum. Multi-way distributional clustering via pairwise interactions. In *Proceedings of ICML-05, the 22nd International Conference on Machine Learning*, pages 41–48, 2005.
- R. Bekkerman and J. Jeon. Multi-modal clustering for multimedia collections. In *Proceedings of CVPR-07, the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2007.
- R. Bekkerman, M. Sahami, and E. Learned-Miller. Combinatorial Markov Random Fields. In *Proceedings of ECML-06, the 17th European Conference on Machine Learning*, pages 30–41, 2006.