tca

Wray Buntine
Jinjing Li

June 2, 2014
Version 0.4

Abstract

tca is a C implementation of the non-parametric dynamic topic model from **?**. It allows both the topic mixtures and the word-topic distributions to change over epochs (time) through two-parameter Poisson-Dirichlet processes (PDP). The software can also be configured to run as a standard LDA model with proper parameters, though this is intended only for testing. "Document completion" testing, coherence measurements with PMI and a few other diagnostic values are supported. *tca* can run multi-core to speed up the computation.

1 Synopsis

tca [**-?**] [**-?***Arg*] *DataStem RepStem*

2 Description

tca reads the collection of files with stem *DataStem* that form the input set of data. When checkpointing, or at termination, the output is written to files with stem *RepStem*. On restart with the **-r** option, some of these are also read initially to restore the previous state. A log of the run is reported to `stderr` if the **-e** option is used. By default, the log goes to `RepStem.log`.

The programme is research software, so not all options or combinations of options work correctly. Note that in this release, all the more experimental features have been stripped, so this release contains only moderately well tested components.

The programme runs a Gibbs sampler for a model consisting of the following three parts:

theta: this is the prior on topic vector (theta) for each document. *tca* uses a Poisson-Dirichlet distribution with parameters a_θ and b_θ . The vector has dimension T (the number of topics).

phi: this is the prior on word vector (phi) for each topic. *tca* uses a Poisson-Dirichlet distribution with parameters a_ϕ and b_ϕ . The vector has dimension W (the number of words).

mu: this is the prior on topic vector (mu) for each epoch. *tca* uses a Poisson-Dirichlet distribution with parameters a_μ and b_μ . The vector has dimension T (the number of topics).

burst: this is the burstiness component which has a document specific variant of the word vector for each topic. This is not used by default.

It is possible to find-tune the model specification through the **-S** option.

The programme uses generalised second order Stirling numbers with the library extracted from *libstb* version 1.8 released at `https://github.com/wbuntine/libstb`. This is initialised with predefined bounds on the tables, and these can be modified with the **-N** option. This should be used for collections with more than 20,000 documents, but its best to run first and on error, increase the bounds.

3 Options

Options have a single letter followed by a possible single argument. Options are grouped under the following functions: *setting of hyperparameters*, *controlling sampling of hyperparameters*, *general control*, and *testing and reports*

3.1 Setting of hyperparameters

-S*var=value* Set variable `var` to float `value`, where `var` can be one of:

- **am** discount parameter for `mu`, the topic distribution.
- **at** discount parameter for `theta`, topic distribution per document.
- **ap0** discount parameter for `phi`, the word distribution per topic, in the first epoch.
- **ap1** discount parameter for `phi` for the second epoch onwards.
- **bm0** concentration parameter for `mu` in the first epoch.
- **bm1** concentration parameter for `mu` for the second epoch onwards.
- **bt** concentration parameter for `theta`
- **b0p** concentration parameter for on the prior of `phi`.
- **b0m** concentration parameter for on the prior of `mu`
- **bp0** concentration parameter for `phi` in the first epoch.
- **bp1** concentration parameter for `phi` for the second epoch onwards.
- **ab** discount parameter for burstiness
- **bb** concentration parameter for burstiness, a constant initially but subsequent sampling will allow a different value per topic.

If the discount parameter is set to 0, the distribution will become a symmetric Dirichlet, the LDA default.

3.2 Controlling sampling of hyperparameters

-F*var* Fix the variable `var` where it takes the value of a hyper-parameter to the **-S** option.

-g*var,int* Set the extra integer parameter for sampling `var` to `int`.

-G*var,cycles,start* Sample the variable `var` where it takes the value of a hyper-parameter to the **-S** option. The sampling will be after `start` cycles and then repeat every `cycles` cycles.

3.3 General control

-b*epochs* Specify the maximum number of previous epochs that table indicators can use for computation.

-c*cycles* Do a checkpoint every this many `cycles`. This saves the output statistics and the parameter file adequate to do a restart with **-r** option.

-C*cycles* Stop after this many `cycles`. Default is 100.

-d*dots* For really big batches of data, print a "." every `dots` documents within a single cycle.

-e Reroute logging to the `stderr`.

-f*format* Read input data from data formatted according to the type `format`. Data is expected to come from an input file with name `DataStem.Suff` where `Suff` is an appropriate suffix. These are given with Input Files below. Allowed formats are: `ldac`, `witdit`, `docword`, `bag` and `lst`.

-K*topics* Set T the maximum number of topics. Default is 10.

-N*maxN,maxM* Set maximum for the Stirling number tables to count `maxM` and maximum table counts for a_μ and a_ϕ to `maxM`. Default is 1000,10000. Higher numbers are needed when running with large number of documents.

-q*threads* If compiled with threading, enables this many threads. Default is 1.

-r Restart from a previous run, reloading data and parameters.

-R Restart from the output file from *hca* and the beta side is held constant and not sampled. Can significantly speed up testing or querying sometimes.

-s*seed* Initialise the random number seed.

-v Up verbosity by one increment. Starts at zero and currently understands 0-3.

-V load the dictionary from the `DataStem.tokens` file for use in reporting. It has one token per line.

-W*max* Set to the maximum number of unique words to `max`

3.4 Testing and reports

-h*Hold,arg* Do document completion testing on the test set. There are three styles of document completion implemented. When `Hold` is the string "doc", then every `arg`-th word is held out in estimating the latent variables (like theta) for the document and used instead for testing of perplexity. That is, words at document positions `arg-1`, `2*arg-1`, *etc.* When `Hold` is the string "dict", then every `arg`-th word in the dictionary is held out in estimating and used for testing. So if a word has dictionary index `arg-1`, `2*arg-1`, *etc.*, it is held out. When `Hold` is the string "fract", then the `fract` proportion at the tail of the document is held out. The initial proportion is used in estimating.

-l*Diag,cycles,start* Do a run-time estimation of the diagnostic `Diag` starting after the `start` cycle and then taking the estimate every `cycles` cycle. Diagnostics are:

sp Estimate topic sparsity in the theta matrix for the words given in `DataStem.smap`. Results placed in `RepStem.smap`. The report gives "topic/weight" for topics including the word.

prog How often to do the standard diagnostic reports (default is every 5-th cycle).

phi Estimate the word probability vector for each topic. Stored in the `RepStem.phi` file.

testprob Estimate the topic probability vector for each test document. Stored in the `RepStem.testprob` file.

theta Estimate the topic probability vector for each training document. Stored in the `RepStem.theta` file.

-L*Diag,cycles,start* Do a diagnostic estimate `Diag` after all Gibbs sampling is complete. Sampling of the estimate starts after the `start` cycle and goes for a total of `cycles` cycles (including the starting ones). Diagnostics are:

class Estimate class probabilities with "true" classes given in `DataStem.class` and then produce confusion matrix for the test data. Output to files `DataStem.cnfs` and `DataStem.pcnfs`.

like Estimate likelihood/perplexity on the test set using the standard (biased) document likelihood.

-o*score* Scoring rule to pick top words for printing. Methods are 'count', 'idf', and 'cost'.

-p Report topic coherency in the log file, and save results in the parameter file. This requires a `DataStem.pmi` or `DataStem.pmi.gz` file exist in the right format. This can be created with the *mkmat.pl* and *cooc2pmi.pl* scripts in the scripts directory of the release. The format is a simple sparse matrix form with lines of the form "N M PMI" for word indices (offset by 0) N and M and PMI value.

-t*size* Specify size of training set. It takes the first `size` entries in the data set. Default is all the set minus the test data.

-T*size* Specify size of test set. It takes the last `size` entries as the test set. Default is zero.

-V load the dictionary from the `DataStem.tokens` file for use in reporting. It has one token per line.

4 Input Files

The following files provide details about the dataset. The filenames are constructed by adding a suffix to the data stem. The data (document+word) format itself can be one of four different formats and is specified with the **-f** option.

`DataStem.class` Class index for each document, one per line. Optional file used with some reports instigated by **-X** or **-L***class* options.

`DataStem.cnf` This format appears in some UCI data sets

`DataStem.docs` A list of documents including their titles

`DataStem.epoch` epoch information file. The first number indicates the total number of epochs in the dataset, followed by the number of entries in each epoch in subsequent lines.

`DataStem.ldac` Standard LdaC format. Word indices to the dictionary are offset by 1.

`DataStem.srcpar` summary information on the total number of documents and features of the dataset.

`DataStem.stops` List of stop words

`DataStem.titles` A list of titles for all entries

`DataStem.tokens` tokens/words in the dictionary, one per line.

`DataStem.tpc` Meta data of the dataset structure

`DataStem.txtbag` default bag or list format for *linkBags*(1) command of `DCABags`. Word indices offset by 0.

`DataStem.words` tokens/words in the dictionary, one per line.

5 Output Files

The following files are output when the system checkpoints or at the end of the run. These are built by adding a suffix to the report stem, `RepStem`. The first set of files are:

`RepStem.cnfs`+`RepStem.pcnfs` Best prediction and probability vector confusion matrices built on the test data with the **-L***class* command.

`RepStem.log` Log file created if **-e** option not used.

`RepStem.par` Parameter and dimensions file in simple "var = value" format. These are detailed in the next section.

`RepStem.phi` The Phi matrix written as a binary file: first W (dictionary size), T (topics), C (sample size) are written as 32 bit integers and then the full Phi matrix as native floats with W as the minor index. Only generated with appropriate use of the **-l***phi* option.

`RepStem.smap` Optional sparsity report on the word indices listed in `DataStem.smap`. The report is instigated by the **-l***sp* option.

`RepStem.top` A simple text report giving the top word indices for each topic. If a hierarchical model in use, then the "-1" topic is for the base distribution of words. Word indices are offset from 0.

`RepStem.theta` Estimated topic probabilities for each training document written in a simple sparse form. The class index ("-1" or "+1" for binary classes, otherwise just the index) is also added if it exists. Topic indices are offset by 0. Only generated with appropriate use of the **-l***theta* option.

`RepStem.tpk` Topic-word information per epoch.

`RepStem.testprob` Like the `-ltheta` option but for the test documents. Only generated with appropriate use of the **-l***testprob* option.

The second set of files gives the actual runtime statistics. Output matrices are in a simple readable sparse vector format the same as the `DataStem.docword` format.

`RepStem.ndt` Document by topic counts.

`RepStem.nwt` Word by topic counts.

`RepStem.tdt` Document by topic table counts if the Alpha side of the model is non-parametric.

`RepStem.twt` Word by topic table counts if the Beta side of the model is non-parametric.

`RepStem.zt` With no burstiness, gives topic index (offset by 0), one per line. With burstiness, gives one "z,r" per line where "z" is the topic index (offset by 0) and "r" is the burst table indicator, which is 1 if the word contributes to standard LDA statistics, and 0 if burstiness modelling says the word is a burst so does not contribute to LDA statistics.

These files along with `RepStem.par` are input on a restart using **-r**.

6 The Parameter File

The parameter file has the following *dimensions*:

N – number of words in the full collection, summed over all documents.

NT – number of words in the training set, summed over all training documents.

W – number of words in the dictionary.

D – number of documents in total.

TRAIN – number of documents to train on, is always the the first ones in the file.

TEST – number of documents to test on, is always the the last ones in the file.

T – maximum number of topics.

ITER – number of major cycles made last.

In addition, the float parameters allowed to be specified with the **-F** and **-G** options are also given. This includes all the hyper-parameters for distribution μ, ϕ and θ (`am`, `bm0`, `bm1`, `b0m`, `at`, `bt`, `ab` , `bb`, `ap0`, `ap1`, `b0p`, `bp1`).

7 Examples

These examples work as is on late model Linux, Macs and Windows. However, you need to replace the executable, `tca`, by the system dependent one, from the install directory where the `data/` directory is. For instance, on Windows that might be `tca/tca.exe`.

Run the topic model from ? with fixed hyper-parameters on the full dataset and no testing, sending logging to *stderr*.

```
tca -v -e -V -K20 -C100 data/ch c1
```

The parameters as they are sampled will be printed on a line beginning wth "Par:". At the end, the top 20 words will be printed and the final training perplexity printed. This is based on the posterior probability, not word probability estimates. Restart and build a topic probability vector for each document, as well as sparsity mappings for the words in `data/ch.smap` file.

```
tca -v -r -e -lsp,2,1 -ltheta,2,1 -C20 data/ch c1
```

Now view the sparsity report at `c1.smap` and the topic probabilities at `c1.theta`, and the values saved in the parameter file `c1.par`. Rerun the model fitting on the full dataset and testing on the final 100 documents. Logging now to `c1.log`. Checkpoint every 20 cycles (note, we usually only do this for cycles taking over 10 minutes each).

```
tca -v -V -K20 -C100 -c20 -T100 data/ch c1
```

View the end of the log file to get the test perplexity, which is printed after "log_2(test perpML)" Now rerun but use document completion (every 4th word), not the default likelihood calculation.

```
tca -v -V -r -hdoc,4   data/ch c1
```

View the end of the log file to get the test perplexity, which is printed after "log_2(test perpHold)". Note it is also recorded in the parameter file. Restart, no sampling this time using **-C***0*, just record the PMI and the classification details on test data.

```
tca -v -V -r -C0 -Llike,0,0 -Lclass,10,1 -p  data/ch c1
```

Note the **-L***like,0,0* option is used to prevent it doing test likelihood calculations, which are potentially slow on larger data sets. The PMI data has a value printed for each topic as well as a final average. It bases its calcuations on the matrix `data/ch/pmi.gz` created explicitly for this test set. For other datasets, you will need to down load prepared PMI matrices from the project homepage.

To relax the assumptions on the fixed hyperparameters, one can run with the optimisation options. For instance, if we allow the word-topic distribution (`phi`) to evolve at different speed for different topics, we can sample the concentration parameter `bp0` and `bp1` as the following example,

```
tca -v -v -K20 -C200 -Gbp0,2,2 -Gbp1,2,2 data/ch c1
```

The hyperparameter sampling slows it down quite a bit but seems to make a significant difference. Unused topics sometimes get a very low concentration.

8 Errors

There is some error reporting on failure.

If the software quits during a run on larger data with an error message like:

```
  S_V(N,M,A) tagged 'XXX' hit bounds (BN,BM)
```

for integers `N,M` and label `XXX` then you need to increase the bounds. If the tag `XXX` is "SX, docXtopic PYP" then increase the bound `BM` using the option **-N***BM* with an increased `BM`. If the tag `XXX` is "SX, docXtopic PYP" then increase both bounds `BN,BM` using the option **-N***BM,BN* (note the order of the bounds).

For other errors, please report to the maintainer.

9 See Also

The command *linkBags*(1) is available from `http://mloss.org` in the software *DCABags*. The extended library *libstb*, parts of which are included, is available individually from `http://mloss.org`.

Other supporting software and data sets are available at the project home page at `https://github.com/wbuntine/topic-models`

10 Version

This programme is version 0.4 of June 2, 2014. This incorporates parts of the library *libstb* version 1.8 also of June 2, 2014.

11 License and Copyright

12 Author

Prof. Wray Buntine
Email: `Wray.Buntine@monash.edu`

Some parts also done by Dr. Jinjing Li and Mr. Swapnil Mishra.

References

Buntine, W. and Li, J. (2014). Dynamic Topic Models using Hierarchical Two-parameter Poisson-Dirichlet Process