BAT-Framework v0.1: A Quick Reference

Marco Cornolti*- A^3 Lab - University of Pisa

January 30, 2013

1 FAQ

Before reading the rest, have a look at these answers.

What does this framework provide?
A: An easy API to run many in-depth tests to measure your topic annotator's ability to annotate the instances of a dataset. You can easily add e new annotator or a new dataset, and the framework will perform the testing. Testing can be also done to find the similarities of two annotators.

What does NOT this framework provide?
A: This framework does not let you create a new dataset (e.g. though crowd sourcing) and is not meant to modify and improving your text annotator. All it does is treating your text annotator as a black box and deep-inspect its results.

Does this framework come along with any dataset I can use?
A: Yes. Out of the box, you can use AQUAINT (news), MSNBC (news), IITB (web pages). Because of copyright issues, it's slightly more tricky to use the AIDA dataset (news) and the Meij dataset (tweets). (See Section 4.2)

Does this framework come along with any topic annotator I can use?
A: Yes. Out of the box, you can only use DBPedia Spotlight, that is the only annotator that needs no keys to access the API. The interface for Tagme, Wikipedia Miner and AIDA are also given, but you need to ask the authors a key to access them (See Section 4.3). You can even run Illinois Wikifier locally.

What is the basic knowledge I need to understand what's going on?
A: You need to be familiar with the following concepts:

- What is an *annotation* and a *tag*, and their difference.
- The definition of *true positive*, *false positive*, *false negative*, *F1*, *precision* and *recall* in their generalized version using a *match relation* M.

*{my last name}@di.unipi.it

- Problems *C2W, Sc2W, A2W, Sa2W, D2W* and their hierarchy. How can an annotator solving *A2W* be compared against a dataset giving a gold standard in *C2W* form?

All these issues are extensively exposed in chapters 2-5 of Marco Cornolti's thesis available on-line [5].

I don't agree with your implementation of <X>, can I change the code?
A: This software is completely open source (licensed under the GPLv3), and intended to become a basis for the whole scientific community working in the field of the topic annotators. You are strongly encouraged not only to edit the code, but also to interact with it, proposing modifications that may be included in the main branch and released in future versions.

I have created a new topic annotator, can I have its interface included in the main branch?
A: Yes, just develop the interface and send it to the author of this paper.

How do I contact the developers of this framework to propose modifications, request new features or report bugs?
A: Since our community, nowadays, is not that wide, we did not set up a bug tracking system. All requests should be e-mailed to the author of this paper.

2 Introduction

This is a quick reference to assist the configuration and usage of the Topic-Annotator Benchmarking Framework (BAT Framework) developed by the A^3 lab [1] at the Department of Computer Science[1], University of Pisa.

Before reading this quick reference, you should be familiar with a set of concepts underlying the framework, that include: the set of problems related to topic annotation (Sc2W, A2W, etc.); the way a system solving problem P' can be compared against a dataset solving problem P'' (problem reductions); the meaning of the extended measures based on a Match Relation (new TP, FP, F1, etc.); the meaning of each Match Relation (M_w, M_m, etc.); the way the similarity between two systems can be measured (the jaccard-like similarity measure). All this information is presented in a Master Thesis available on-line [5].

Note that all package names are in the form `it.acubelab.batframework.X`, therefore noly `X` will be used in this reference.

3 How it works

The BAT Framework basically take the set of instances from a dataset, run a topic annotator on those instances, and compare the annotator output against

[1] `http://www.di.unipi.it`

the gold standard given by the dataset, computing a set of measures to find out how close the output is to the ground truth.

The framework is easily extendible: a new annotator or a new dataset can be added implementing one single java class. This quick reference will only cover these two aspects, but the annotator can also be extended by adding new performance measures, similarity measures, problems, and match relations.

4 Setting up the whole thing

4.1 Downloading the framework and preparing the environment

The BAT-Framework can be downloaded from the "Software" section of the A^3 lab web site [**?**]. It's a jar file that includes the source and compiled code. The jar is distributed in a zip archive `bat-env-0.1.zip` that also provides the suggested directory structure for the experiments and the basic configuration for the annotators. Once un-zipped, the directory structure will be:

`lib/`	jars of the BAT-Framework and the needed libraries;
`charts/`	storage of the gnuplot charts and the scripts to generate them;
`benchmark/cache/`	storage of the cache for the annotation results and the queried Wikipedia data;
`benchmark/configs/`	configuration for the access to the annotators (E.g. API keys, parameters);
`benchmark/datasets/`	raw instances and gold standard forming the datasets;
`benchmark/libs/`	libraries needed by the annotator/datset plugins;
`benchmark/systems/`	the systems that have to be downloaded instead of queried;

Let `BATframework/` be the base directory for our experiments. Extract the environment zip to this directory using these commands:

- `unzip bat-env-0.1.zip -d BATframework`
- `cd BATframework`

4.2 Configuring shipped datasets

Some dataset plugins are shipped together with the framework. Unfortunately, only some of them are redistributable due to copyright issues. The datasets that are freely distributed by their authors are shipped within `bat-env-0.1.zip`:

AQUAINT (news), MSNBC (news), IITB (web pages), Timeruns (incrasingly-long documents for time benchmarking purposes). For these four datasets, no configuration is needed and they can be used out-of-the-box.

To generate the other two, here comes a brief explaination. Note that you only have to do the following if you need to run experiments on these datasets!

AIDA/CO-NLL (news) the gold standard is released publicly, but the instances (a subset of the CoNLL 2003 dataset) are not, and you will need the original CoNLL 2003 CD to create the full dataset. Detailed instructions can be found in the AIDA site [3]. Once you have created the file `AIDA-YAGO2-dataset.tsv`, copy it to the `benchmark/datasets/aida/` folder.

Meij (tweets) same issue: the gold standard is released in E. Meij's website[**?**] but the original instances (tweets), despite being drawn from public twitter profiles, can't be redistributed. To use this dataset, you will need the three files `wsdm2012_annotations.txt`, `wsdm2012_qrels.txt` (shipped) `original_tweets.list` (not shipped) in the `benchmark/datasets/meij/` directory.

For any problem in configuring a dataset, please e-mail the author of this guide.

4.3 Configuring shipped annotators

Some of the shipped annotators can be accessed on-line or run locally with no extra configuration. Others, namely AIDA and TagMe, need some form of permission in order to be queried. These permission must be given to the annotator editing a few files under `benchmark/configs`. Again: you need to do that only if you need to use these annotators.

AIDA you need the AIDA Team to provide you with a username and password to access their Java RMI web service. Details on how to get access grants are given at their web page [2]. Once you have them, open `benchmark/configs/aida/config.xml` and replace `USER` and `PASSWD` with yours.

TagMe send an e-mail to the TagMe team[2] to ask for an access key, then edit `benchmark/configs/tagme/config.xml` replacing `KEY` with your key.

Illinois Wikifier does not provide a remote API, thus must be downloaded and run locally. You have to launch it externally of the framework. The framework and Illinois Wikifier will communicate through an IP socket, pointing to localhost. Thus, they must run on the same machine.

[2] `{name of the system}@di.unipi.it`

To prepare Illinois Wikifier, do the following steps.

Download the 4 GB software from [4], move the file `Wikifier_ACL2011_Package.tar.bz2` to `benchmark/systems/illinois/`, unzip with the following commands:

- `cd benchmark/systems/illinois/`
- `bunzip2 Wikifier_ACL2011_Package.tar.bz2`
- `tar -xf Wikifier_ACL2011_Package.tar`
- `mv Wikifier_ACL2011_Package/* .`
- `rmdir Wikifier_ACL2011_Package/`
- `ls`

Now you should see directories like `Code/`, `Config/`, `Data/`. To launch the Illinois Wikifier, do the following.

Go back to the main directory:

- `cd ../../../../BATframework/`

Launch the server:

- `. benchmark/systems/illinois/start_illinois.sh`

After a lot of output, the Illinois Wikifier prints "Waiting for a connection...". At this stage, the BAT-Framework can query the Illinois Wikifier.

5 Adding a dataset

It's trivial to plug a new dataset to the BAT-Framework. All you need to do is to create a class (E.g. `MyDataset`) that implements `problems.`X`Dataset`, where X is the (computationally) hardest problem for which your dataset gives a gold standard. For example, if your dataset provides, for each document, a set of annotations, `MyDataset` will implement `A2WDataset`.

Since the class hierarchy of the datasets follows that of the problems, your dataset will also have to provide the adapted gold standard for all easier problems (an $A2W$ dataset is also a $C2W$ dataset). Standard methods to adapt a solution between problems are given in class `ProblemReduction` of package `utils`.

In the constructor, the dataset normally loads the set of instances and the gold standard, parsing them from a file. Please have a look at the example `DummyDataset` in package `examples` or the implementation of the other datasets in package `datsetPlugins`.

6 Adding an annotator

Adding an annotator is very similar to adding a dataset. All you have to do is create a class `MyAnnotator` implementing `problems.`X`System`, where X is the hardest problem that your annotator can solve.

As for the datasets, if $A \propto B$, an annotator solving B will also be able to solve A, thus an annotator implementing the interface B`System` (providing method `solve`B) will also implement A`System` (providing method `solve`A). Standard methods of class `ProblemReduction` can be used to adapt the solution.

In the implementation of the `solve`X method, `MyAnnotator` will normally query a remote annotator (through a web service, Java RMI, ...).

Please have a look at the example `LulzAnnotator` in package `examples`.

7 Testing

Before launching an experiment, you may want to test if the dataset(s) and the system(s) you are going to use works correctly.

7.1 Testing a dataset

You can test a dataset using three methods provided by the framework.

1. Print basic statistics;
2. Dump the whole dataset;
3. Export the dataset to XML and check the XML.

An example featuring all three methods for testing the A2W dataset IITB (web pages) is given in class `examples.DumpDataset`, here follows a snippet. Note that similar methods also exist for other kinds of datasets (C2W, Rc2W, ...).

```
public static void main(String[] args) throws Exception {
  //Creating the API to Wikipedia.
  WikipediaApiInterface api = new WikipediaApiInterface("benchmark/cache/
      wid.cache", "benchmark/cache/redirect.cache");

  //Creating the IITB dataset.
  A2WDataset ds = new IITBDataset("benchmark/datasets/iitb/crawledDocs",
      "benchmark/datasets/iitb/CSAW_Annotations.xml", api);

  System.out.println("Printing basic information about dataset " + ds.
      getName());
  //Basic check & Dump basic information
  TestDataset.dumpInfo(ds, api);

  //Dump of the whole dataset
  DumpData.dumpDataset(ds.getTextInstanceList(), ds.
      getA2WGoldStandardList(), api);

  //Export to XML
  Exporter.exportA2WDataset(ds, new FileOutputStream("iitb.xml"), api);
  api.flush();
}
```

7.2 Testing an annotator

Similarly, when you develop a topic annotator, you may want to test its interface to the BAT framework, to see if the output is correct. The best way to do so is using two methods:

1. Do some basic testing;
2. Manually check the output.

In the current implementation, the basic testing simply checks the size of the output (number of solutions) against the size of the input (number of instances in the dataset), that should be the same. But the real check must be done by a human: when dumping the solution, the full set of annotations for each instance should be seen, other output indicates an error in the implementation.

An example featuring these two methods is given in `examples.DumpAnnotator`, where the DBPedia Spotlight annotator is checked against a dummy dataset made only of a few instances.

```
public static void main(String[] args) throws Exception {
  //Do not store the cached results in a file
  BenchmarkCache.useCache(null);

  //Create the API to wikipedia, storing retrieved data in two files, and the interface to DBPedia.
  WikipediaApiInterface wikiApi = new WikipediaApiInterface("wid.cache", "redirect.cache");
  DBPediaApi dbpediaApi = new DBPediaApi();

  //Create the annotator and the dummy dataset.
  Sa2WSystem annotator = new SpotlightAnnotator(dbpediaApi, wikiApi);
  A2WDataset ds = new DummyDataset();

  List<Set<ScoredAnnotation>> computedAnnotations = BenchmarkCache.doSa2WAnnotations(annotator, ds, null, 0);

  //Do some basic check on the annotator's output.
  TestAnnotator.checkOutput(ds, computedAnnotations);

  //Dump of the output.
  DumpData.dumpOutput(ds.getTextInstanceList(), computedAnnotations, wikiApi);

  BenchmarkCache.flush();
  wikiApi.flush();
}
```

8 Running the experiments

After you have tested your datasets and annotators, you are ready to benchmark a system against a dataset. A main method that performs the testing usually follows these steps:

1. Load the dataset;
2. Create the interface to the annotator;

3. Run the annotator on the dataset, obtain the results;
4. Compute the metrics on the results (Optional: varying the threshold in [0,1]);
5. Print the results in latex, gnuplot or human-readable form.

8.1 The naive way

In the example file `examples.LulzMain`, the DBPedia Spotlight annotator (of type Sa2W) is checked against the IITB dataset (of type A2W). The reduction A2W $\propto$ 2W is done with the threshold $t = 0.1$. The metrics are computed with a Weak Annotation Match (considers a match iff the mention overlap and the redirected concept is the same). Before exiting, the results are printed to the screen.

```java
public static void main(String[] args) throws Exception {
  //Store the cache in a file (for future executions).
  BenchmarkCache.useCache("benchmark/cache/results.cache");
  //Create the API to wikipedia, storing retrieved data in two files, and the interface to DBPedia.
  WikipediaApiInterface wikiApi = new WikipediaApiInterface("benchmark/cache/wid.cache", "benchmark/cache/redirect.cache");
  DBPediaApi dbpediaApi = new DBPediaApi();

  //Create the interface to an annotator.
  Sa2WSystem miner = new SpotlightAnnotator(dbpediaApi, wikiApi);
  A2WDataset iitb = new IITBDataset("benchmark/datasets/iitb/crawledDocs", "benchmark/datasets/iitb/CSAW_Annotations.xml", wikiApi);

  //Run the experiment
  List<Set<ScoredAnnotation>> computedAnnotations = BenchmarkCache.doSa2WAnnotations(miner, iitb, null, 0);

  //Adapt the output to the same type of the gold standard (A2W) setting a threshold that excludes all annotations under 0.1
  List<Set<Annotation>> reducedAnnotations = ProblemReduction.Sa2WToA2WList(computedAnnotations, 0.1f);

  //Take the gold standard
  List<Set<Annotation>> goldStandard = iitb.getA2WGoldStandardList();

  //Create a Match relation and the metrics calculator.
  MatchRelation<Annotation> wam = new WeakAnnotationMatch(wikiApi);
  Metrics<Annotation> metrics = new Metrics<Annotation>();

  //Compute the results
  MetricsResultSet rs = metrics.getResult(reducedAnnotations, goldStandard, wam);

  printResults(rs);

  BenchmarkCache.flush();
  wikiApi.flush();
}

public static void printResults(MetricsResultSet rs){
  System.out.printf("Micro-Precision:%.3f%nMicro-Recall:%.3f%nMicro-F1:%.3f%n"+
      "Macro-Precision:%.3f%nMacro-Recall:%.3f%nMacro-F1:%.3f%n"+
      "TP:%d FP:%d FN:%d%n",
      rs.getMicroPrecision(), rs.getMicroRecall(), rs.getMicroF1(),
```

```
        rs.getMacroPrecision(), rs.getMacroRecall(), rs.getMacroF1(),
        rs.getGlobalTp(), rs.getGlobalFp() ,rs.getGlobalFn()
        );
}
```

Running this program for the first time, it will take a while for querying Wikipedia (this is needed to resolve redirects and will generate a lot of output) and to query DBPedia Spotlight for solving the (many and long) instances of IITB. When flushing, all this information is saved to the three files `wid.cache`, `redirect.cache` and `results.cache`. In this way, the second run of the main program will take much less and generate less output.

When running the framework, you may need to include some libraries. To run the bat-framework, a command like this should be ok:

- `java -classpath bin:lib/batframework-0.1.jar: lib/*:benchmark/libs/*/* it.acubelab.batframework. examples.LulzMain`

This assumes your compiled `.class` files are in the `bin/` directory. Remember to replace the path to `LulzMain` with your main method. Also note that you don't actually need to include the jars in `benchmark/libs/*` in the classpath unless you are using those systems.

8.2 The pro way

In the case of annotators solving problems Sa2W and Sc2W (in which tags/annotations have a score), we are interested in finding the threshold $t^* \in [0, 1]$ which globally maximize the micro-F1. This is done by measuring the micro-F1 ranging the threshold t in 128 discrete incrementing fractions of 1. The `utils.RunExperiments` wrapper class provides a set of methods to do such measurements. Moreover, `RunExperiments` provides some methods to run in batch the same experiments using a set of match relations, a set of datasets, and a set of annotators, performing an experiment for each combination. Please refer to the in-line documentation or javadoc for more details.

The methods of `RunExperiments` store the metrics results (F1, recall, precision, ...) in an hashtable. This hashtable can be passed to the methods of `utils.DumpResults` to generate latex tables, gnuplot data, or print the results to the screen. Please refer to the inline documentation of that class for more details.

A less trivial example for a main method, featuring the usage of the wrapper methods of `RunExperiments` and the methods to output results of `DumpResults` is given in `examples.BatchMain` (not listed here). It features almost everything the BAT-Framework can do:

1. Load the cache; print some information about the cached documents;
2. Load all the shipped A2W and C2W datasets;
3. Perform measures of the ability of the Sa2W annotators to solve A2W (using the A2W datasets) for the threshold ranging in $t \in [0, 1]$; print the results in many forms;
4. Perform measures of the ability of the Sa2W annotators to solve C2W (using an A2W datasets and a C2W dataset); print the results;
5. Print the timings in many forms;
6. Print the similarity between pairs of annotators as a latex table.

9 Gnuplotting

Methods `gnuplot*` of class `DumpResults` generate many `.dat` files compatible with `Gnuplot`. Of course, you can do anything you want with them, but the BAT-Framework comes with some templates for generating some standard charts. The three scripts involved in this are:

- `make_charts_performance.p` gnuplot script that generates F1, precision and recall charts;
- `make_charts_time.p` gnuplot script that generates time/F1 charts;
- `make_charts.sh` shell script to launch the other two.

The `make_charts_performance.p` scripts depend on two variables: `DATASET` and `METRICS`. The way they should be set is given in `make_charts.sh`. To clear the environment, it is strongly suggested to delete all `.dat` and `.png` files in che `charts/` folder before running the BAT-Framework.

After properly editing the scripts in `charts/`, you can run them with:

- `. cache/make_charts.sh`

This will generate the `.png` charts.

References

[1] A-cube lab. http://acube.di.unipi.it.

[2] Aida: Accurate online disambiguation of named entities in text and tables. http://www.mpi-inf.mpg.de/yago-naga/aida/.

[3] Aida dataset download page. http://www.mpi-inf.mpg.de/yago-naga/aida/downloads.html.

[4] Illinois wikifier - download page. http://cogcomp.cs.illinois.edu/page/download_view/Wikifier.

[5] Marco Cornolti. A framework to compare text annotators and its applications. Master's thesis, University of Pisa - Computer Science department, 2012. Available at http://etd.adm.unipi.it/theses/available/etd-09262012-113229/unrestricted/tesi.pdf.