Journal of Machine Learning Research 15 (2014) ??-?? Submitted 4/14; Published ??/14

MeTA: A Modern C++ Data Sciences Toolkit

Sean Massung massung1@illinois.edu

Chase Geigle geigle1@illinois.edu

Department of Computer Science
University of Illinois at Urbana-Champaign
Champaign, IL, USA

Editor: Editor Name

Abstract

MeTA is developed to unite machine learning and information retrieval algorithms in one easy-to-use toolkit. Its focus on indexing allows it to perform well on large datasets and support online classification. MeTA's liberal open source license encourages contributions, and its extensive online documentation and tutorials make this process straightforward. We perform experiments and show MeTA's performance is competitive with existing software in both domains.

Keywords: information retrieval, machine learning, tokenization, text mining

1. A Unified Framework

The existing environment of open source software solutions for machine learning is fragmented: there is rarely a single location for a wide variety of algorithms. Tools tend to specialize on one particular area, and as such there is a wide variety of tools one must sample when performing different machine learning tasks. For text-mining tasks, this is even more apparent; it is extremely difficult (if not impossible) to find tools that support both traditional information retrieval tasks (like tokenization, indexing, and search) alongside traditional machine learning tasks (like document classification, regression, and topic modeling).

This places an undue burden on researchers—not only are they required to have a detailed understanding of the research problem at hand, but they are now forced to understand this fragmented nature of the open source software community, find the appropriate software packages for their needs, and compile and configure each appropriate tool. Even when this is all done, there is the problem of data formatting—it is unlikely that the tools all have standardized upon a single input format, so a certain amount of "data munging" is now required. All of this detracts from the actual research task at hand, which has a marked impact on the speed with which new ideas are discovered.

The goal of the MeTA project is to address these issues. In particular, we provide a unifying framework for existing machine learning algorithms, allowing researchers to quickly run controlled experiments. We have modularized the feature generation, instance representation, data storage formats, and algorithm implementations; this allows for researchers to make seamless transitions along any of these dimensions with minimal effort.

We have a liberal licensing scheme for MeTA. It is dual-licensed under the University of Illinois/NCSA Open Source Licence and the MIT License. We use these permissive licenses deliberately to allow for MeTA's incorporation into commercial products: we feel that the use of a copyleft license creates unnecessary barriers between the code we develop as researchers and the applications of our techniques to areas outside of the research community.

2. Built for Big Data

Consistency across components is a key feature that allows MeTA to work well with large datasets. This is accomplished via a three-layer architecture. On the first layer, we have tokenizers, analyzers, and all the text processing that accompanies them. Once a document representation is determined, this tool chain is run on a corpus. The indexes are the second layer; they provide an efficient format for storing processed data. The third layer—the application layer—interfaces solely with indexes. This means that we may use the same index for running an SVM as we do to evaluate a ranking function, *without processing the data again*.

Since all applications use these indexes, MeTA is able to support out-of-core classification with some classifiers. We ran our large classification dataset, rcv1 (Lewis et al., 2004), with limited memory (only 95MB) using the `sgd` classifier. Where liblinear failed to run, MeTA was able to finish the classification in just under two minutes.

Besides using MeTA's rich built-in feature generation, it is possible to directly use libsvm-formatted data. This allows preprocessed datasets to be run under MeTA's algorithms. Additionally, MeTA's `forward_index` (used for classification), is stored as libsvm format. Thus the reverse is also true: you may do feature generation with MeTA, and use its index as input to any other program that supports libsvm format.

3. Usability

MeTA is hosted publicly on Github[1], which provides the project with community involvement through its bug/issue tracker and fork/pull request model. Its API is heavily documented[2] and the project website contains several tutorials that cover the major aspects of the toolkit[3] to enable users to get started as fast as possible with little friction.

A major design point in MeTA is to allow for most of the functionality to be configured via a configuration file. This enables exploratory data analysis without having to write (or recompile) any code. Designing the code in this way also encourages the components of the system to be pluggable: the entire indexing process, for example, consists of several modular layers which can be controlled by the configuration file.

A simple class hierarchy allows users to add filters, analyzers, ranking functions, and classifiers with full integration to the toolkit (*e.g.* one may specify user-defined classes in the config file). The process for adding these is detailed in the MeTA online tutorials.

1. `https://github.com/meta-toolkit/meta/`
2. `http://meta-toolkit.github.io/meta/doxygen/`
3. `http://meta-toolkit.github.io/meta/`

Multi-language support is hard to do correctly. Many toolkits sidestep this issue by only supporting ASCII text or the OS language; MeTA supports multiple (non-romance) languages by default, using the industry standard ICU library (ICU Project, 2014). This allows MeTA to tokenize arbitrarily-encoded text in nearly any language.

Unit tests ensure that contributors are confident that their modifications do not break the toolkit. Unit tests are automatically run after each commit and pull request, so developers immediately know if there is an issue (of course, unit tests may be run manually before committing).

4. Experiments

MeTA's IR performance is compared with two well-known search engine toolkits: Lucene (Apache, 2014), a top-level Apache venture; and Lemur (Croft et al., 2013), a joint collaboration between the University of Massachusetts Amherst and Carnegie Mellon University.

MeTA's ML performance is compared with liblinear (Fan et al., 2008), a well-known SVM library; scikit-learn (Pedregosa et al., 2012) a Python ML library; and SVMMulticlass (Joachims, 2008), a competitor to liblinear. Statistics for datasets used in both parts can be found in Fig 1.

4.1 Information Retrieval Performance

First, we compare the speed at which the search engines can index each corpus (Fig 3). We index unigram words with the identical tokenization[1] across toolkits. Furthermore, all experiments were run on a system with an Intel quad core i7-2760QM (2.40GHz) CPU, eight gigabytes of memory, and a 7200 RPM disk. We used the datasets 20newsgroups (Lang, 2013), the blog authorship corpus (Schler et al., 2006), Reddit comments (Anonymous, 2014), TREC 2006 Homepages (TREC, 2014), and English Wikipedia (Wikimedia Foundation, 2013).

We also compare how large the indexes are for each search engine (Fig 4). Ideally, since we are not storing the full text, the indexes should be smaller than the original data, though this is not always the case.

To investigate retrieval speed, we created 500 queries for each dataset by randomly selecting 500 documents, then randomly selecting one sentence from each of the 500 documents. Results are shown in Fig 5. Relevance of returned queries is not an interesting comparison since all indexes should be storing identical information; that is, all tokenize the documents the same way and a given retrieval function will perform the same operations on each indexed document.

We see from the experimental results that MeTA is a competitive information retrieval framework. In indexing speed, it is between Lemur and Lucene. Its index sizes are almost the smallest, except against Lucene's Homepages index. In terms of query speed, again MeTA falls between Lemur and Lucene.

1. All terms are lower-cased, stop words are removed from a common list of 431 stop words, Porter2 (MeTA) or Porter (Lemur, Lucene) stemming is performed, a maximum word length of 35 characters is set, and the original documents are not stored in the index. In addition, each corpus is converted into a single file with one document per line to reduce the effects of many file operations.

4.2 Machine Learning Performance

For the machine learning evaluation, we focus on linear classification across toolkits. Many applications in MeTA are in a textual domain, and linear classification lends itself to the high-dimensional space that comes with text documents. All experiments are performed on a system with a dual core Intel Core i5 CPU (M460) clocked at 2.53GHz and eight gigabytes of RAM.

Four datasets (20news, Blog, Newegg, and the Yelp Academic Dataset (Yelp, 2014)) are textual datasets. The Newegg (crawled ourselves) and Yelp datasets are review datasets, and we consider only a partial list of the whole dataset where we have removed all three-star reviews. We used MeTA for tokenization and feature generation, applying the same constraints as we did for the indexing tests. Randomized training (two thirds) and test splits (one third) were generated for each of these datasets. For rcv1, we used the existing tokenization and training/test splits available on the LIBSVM data website (Chang and Lin, 2013).

In Fig 2, we can see that MeTA performs well both in terms of speed and accuracy, and presents itself as a viable option in the machine learning domain.

	Docs	**Size**	$\lvert D\rvert_{avg}$	$\lvert V\rvert$
20news	18,828	39 MB	223.4	97,667
Blog	18,713	654 MB	3432.1	462,425
Homes	324,553	5.0 GB	1742.3	3,609,811
Newegg	106,998	57 MB	55.6	38,533
rcv1	697,641	1.2 GB	*N/A*	47,236
Reddit	15,497,530	2.6 GB	18.6	1,942,065
Yelp	194,543	130 MB	74.9	75,703
Wiki	8,284,146	25 GB	272.3	15,248,289

Figure 1: Datasets for IR and ML experiments.

	liblinear	**scikit**	**SVM$^{\text{mult}}$**	**MeTA**
20news	88.18%	78.03%	70.27%	**89.10%**
($k = 20$)	15.64s	5.28s	35.36s	7.55s
Blog	45.53%	35.91%	**46.22%**	41.14%
($k = 7$)	310.6s	46.94s	164.6s	28.34s
Newegg	90.39%	90.71%	78.71%	**91.96%**
($k = 2$)	23.90s	10.93s	3.806s	4.182s
Yelp	91.65%	92.02%	80.22%	**92.98%**
($k = 2$)	51.89s	27.48s	9.700s	8.854s
rcv1	**88.43%**	87.68%	55.21%	79.06%
($k = 2$)	21.04s	99.07s	3.37s	37.42s

Figure 2: Accuracy and speed classification results.

	Lemur	**Lucene**	**MeTA**
20news	4s	4s	5s
Blog	1m 22s	43s	1m 2s
Reddit	15m 41s	4m 2s	10m 35s
Homes	15m 31s	7m 11s	13m 20s
Wiki	1h 39m 7s	33m 0s	1h 21m 14s

Figure 3: Indexing speed.

Lemur	**Lucene**	**MeTA**
62 MB	8.5 MB	7.1 MB
775 MB	119 MB	40 MB
6.5 GB	636 MB	1.1 GB
7.1 GB	1.3 GB	479 MB
30 GB	5.6 GB	2.5 GB

Figure 4: Size of each index.

Lemur	**Lucene**	**MeTA**
37s	2s	2s
1m 36s	4s	7s
1h 50m 55s	29s	18m 28s
11m 12s	39s	1m 54s
1h 10m 4s	2m 20s	14m 21s

Figure 5: Speed for 500 queries.

5. Conclusions

We have shown MeTA is a valuable resource for text mining applications; it is both a viable alternative to existing toolkits and unifies algorithms from information retrieval and machine learning. This allows one extensible, consistent framework to replace the many disparate alternatives.

References

Anonymous. Reddit Comments over 15 Days, April 2014. http://www.reddit.com/r/datasets/comments/1mbsa2/155m_reddit_comments_over_15_days/.

Apache. Apache Lucene, March 2014. http://lucene.apache.org/.

Chih-Chung Chang and Chih-Jen Lin. Libsvm data: Classification, regression, and multi-label, October 2013. http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/.

Bruce Croft, Jamie Callan, David Fisher, Sam Huston, Marc Cartright, David Pane, John Lafferty, James Allan, Howard Turtle, ChengXiang Zhai, Thi Truong Avrahami, Matt Bilotti, Jon Brown, Kevyn Collins-Thompson, Fangfang Feng, Mark J. Hoy, Leah Larkey, Don Metzler, Luo Si, Trevor Strohman, Yangbo Zhu, Paul Ogilvie, Nancy Steadle, and Le Zhao. The Lemur Project, October 2013. www.lemurproject.org.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *Journal of Machine Learning Research*, pages 1871–1874, 2008.

ICU Project. International Components for Unicode, April 2014. http://site.icu-project.org/.

Thorsten Joachims. SVMmulticlass: Multi-Class Support Vector Machine, August 2008. http://www.cs.cornell.edu/people/tj/svm_light/svm_multiclass.html.

Ken Lang. The 20 newsgroups data set, October 2013. http://qwone.com/~jason/20Newsgroups/.

David D. Lewis, Yiming Yang, Tony G. Rose, and Fan Li. Rcv1: A new benchmark collection for text categorization research. *J. Mach. Learn. Res.*, 5:361–397, December 2004. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=1005332.1005345.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake VanderPlas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Edouard Duchesnay. Scikit-learn: Machine learning in python. *CoRR*, abs/1201.0490, 2012.

Jonathan Schler, Moshe Koppel, Shlomo Argamon, and James W. Pennebaker. Effects of Age and Gender on Blogging. In *AAAI Spring Symposium: Computational Approaches to Analyzing Weblogs*, pages 199–205, 2006. http://u.cs.biu.ac.il/~koppel/BlogCorpus.htm.

TREC. The BLOGS06 Test Collection, April 2014. http://ir.dcs.gla.ac.uk/test_collections/blogs06info.html.

Wikimedia Foundation. Wikipedia database download, October 2013. http://en.wikipedia.org/wiki/Wikipedia:Database_download.

Yelp. Yelp Academic Dataset, April 2014. https://www.yelp.com/academic_dataset.