EXPERIMENTAL ALGORITHMICS APPLIED TO ON-LINE MACHINE LEARNING

Thomas Bartz-Beielstein
SPOTSeven Lab, TH Köln, Gummersbach, Germany
thomas.bartz-beielstein@th-koeln.de

Abstract The application of methods from experimental algorithmics to on-line or streaming data is referred to as experimental algorithmics for streaming data (EADS). This paper proposes an experimental methodology for on-line machine learning algorithms, i.e., for algorithms that work on data that are available in a sequential order. It is demonstrated how established tools from experimental algorithmics can be applied in the on-line or streaming data setting. The massive on-line analysis framework is used to perform the experiments. Benefits of a well-defined report structure are discussed.

Keywords: Experimental algorithmics, Massive on-line analysis, On-line machine learning, Streaming data.

1. Introduction: Experimental Algorithmics

This article is devoted to the question "Why is an experimental methodology necessary for the analysis of on-line algorithms?" We will mention two reasons to motivate the approach presented in this paper. First, without a sound methodology, there is the danger of generating arbitrary results, i.e., results that happened by chance; results that are not reproducible; results that depend on the seed of a random number generator; results that are statistically questionable; results that are statistically significant, but scientifically meaningless; results that are not generalizable; etc. Second, experiments are the cornerstone of the scientific method. Even the discovery of scientific highly relevant results is of no use, if they remain unpublished or if they are published in an incomprehensible manner. Discussion is the key ingredient of modern science.

Experimental algorithmics (EA) uses empirical methods to analyze and understand the behavior of algorithms. Experimental algorithmics evolved over the last three decades and provides tools for sound experimental analysis of algorithms. Main contributions, which influenced the field of EA are McGeoch's thesis "Experimental Analysis of Algorithms" [19], the experimental evaluation of simulated annealing by Johnson et al. [18], and the article about designing and reporting computational experiments with heuristic methods from Barr et al. [1]. And, Hooker's papers with the striking titles "Needed: An empirical science of algorithms" and "Testing Heuristics: We Have It All Wrong" [15, 16], which really struck a nerve. Theoreticians recognized that their methods can benefit from experimental analysis and the discipline of *algorithm engineering* was established [8]. Parameter tuning methods gained more and more attention in the *machine learning* (ML) and *computational intelligence* (CI) communities. Eiben and Jelasity's "Critical Note on Experimental Research Methodology in EC" [11] enforced the discussion. This increased awareness resulted in several tutorials, workshops, and special sessions devoted to experimental research in evolutionary computation. Results from these efforts are summarized in the collection "Experimental Methods for the Analysis of Optimization Algorithms" [2].

This overview is by far not complete, and several important publications are missing. However, it illustrates the development of an emerging field and its importance. The standard approach described so far focuses on relatively small, static data sets that can be analyzed off-line. We propose an extension of EA to the field of stream data, which will be referred to as *experimental algorithmics for streaming data* (EASD). This extension is motivated by the enormous growth of data in the last decades. Machine learning, i.e., automatically extract information from data, was considered the solution to the immense increase of data. The field of *data mining* evolved to handle data that does not fit into working memory: Data mining became popular, because it provides tools for very large, but static data sets. Models cannot be updated when new data arrives. Nowadays, data are collected in nearly every device—massive, data streams are ubiquitous. Especially, industrial production processes generate huge and dynamic data. This leads to the development of the *data stream paradigm.* Bifet et al. [5] describe core assumptions of data stream processing as follows:

(S-1) The training examples can be briefly inspected a single time only.

(S-2) The data arrive in a high speed stream.

(S-3) Because the memory is limited, data must be discarded to process the next examples.

(S-4) The order of data arrival is unknown.

(S-5) The model is updated incrementally, i.e., directly when a new data arrives.

(S-6) Anytime property: The model can be applied at any point between training examples.

(S-7) Last but not least: theory is nice, but empirical evidence of algorithm performance is necessary.

We will develop an experimental methodology for on-line machine learning, i.e., for situations in which data becomes available in a sequential order. The data is used to update the predictor for future data at each step. On-line learning differs from traditional batch learning techniques, which generate the predictor by learning on the entire training data set at once. The terms "on-line" and "data stream" will be used synonymously in the following.

This paper is structured as follows. Section 2 compares the traditional batch setting with the stream data setting. How to assess model performance is described in Section 3. A simple experiment, which exemplifies the EASD approach, is presented in Sec. 4. This article concludes with a summary in Section 5.

2. Batch Versus Stream Classification

By comparing the traditional batch and the stream classification, the following observations can be made: Both classification procedures partition the data into test and training set. In contrast to batch classification, stream classification is a cyclic process, which uses nonpermanent data. The elementary steps used in both settings are illustrated in Fig. 1. The figure is based on the data stream classification cycle in Bifet et al. [6].

The batch classification cycle processes data as follows:

(CB-1) Input, i.e., the algorithm receives the data.

(CB-2) Learn, i.e., the algorithm processes the data and generates its own data structures (builds a model).

(CB-3) Predict, i.e., the algorithm predicts the class of unseen data using the test set.

Data availability differs in the stream classification cycle. Additional restrictions have to be considered [6]. Freely adapted from Bifet et al. [6], the data stream processing can be described as follows:

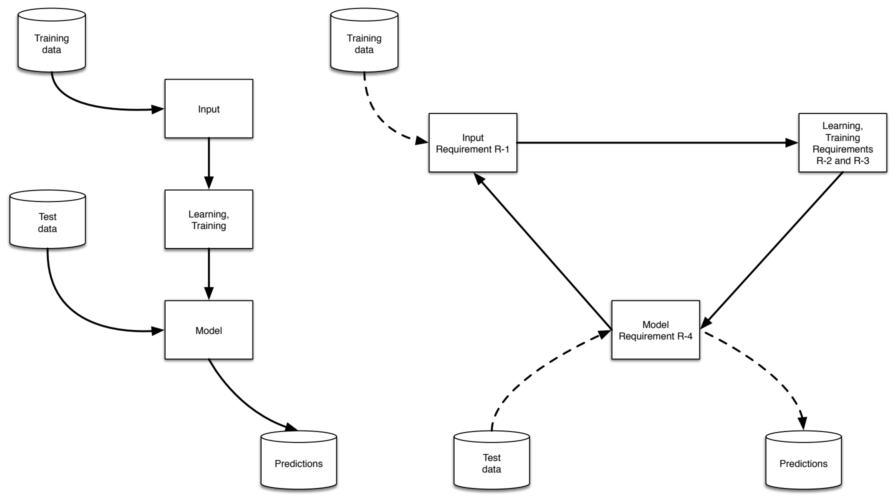

Figure 1: Left: The batch classification cycle. *Right:* The stream classification cycle. Dotted lines represent nonpermanent data. Both classification cycles partition the data into test and training set. To keep the illustration simple, this split is not shown.

(CS-1) Input, i.e., the algorithm receives the next data from the stream. At his stage of the process, the *process only once* (R-1) requirement has to be considered: Data stream data is accepted as they arrive. After inspection, the data is not available any more. However, the algorithm itself is allowed to set up an archive (memory).

(CS-2) Learn, i.e., the algorithm processes the data and updates its own data structures (updates the model). The limited memory and limited time requirements (R-2) and (R-3), respectively, have to be considered. Data stream algorithms allow processing data that are several times bigger than the working memory and real-time processing requires that the algorithm process the data quickly (or even faster) than they arrive.

(CS-3) Predict, i.e., the algorithm is able to receive the next data. It is also able to predict the class of unseen data. The *predict at any point* requirement has to be considered. The best model should be generated as efficiently as possible.

3. Assessing Model Performance

Elementary performance criteria for data stream algorithms are based on

(P-1) Time (speed): We consider the amount of time needed (i) to learn and (ii) to predict. If the time limit is reached, continuing the data processing will take longer or results will loose precision. This consequence is not so hard as the space limit, because overriding the space limit will force the algorithm to stop.

(P-2) Space (memory): A simple strategy for the handling space budget is to stop once the limit is reached. To continue processing if the the space limit is reached is to force the algorithm to discard parts of its data.

(P-3) Error rates (statistical measures): The prediction error is considered. Several error measures are available [17, 26].

A contingency table or *confusion matrix* is a standard methods to summarize results. Based on the values from the confusion matrix, the *accuracy* can be determined as the percentage of correct classifications, i.e., it is defined as the sum of the number of true positives and the number of true negatives divided by the total number of examples (total population). Since accuracy can be misleading (consider the so-called accuracy paradox), further measures are commonly used [27]: For example, the *precision* is defined as the number of true positives divided by the number of true positives and false positives. Precision is also referred to as the *positive predictive value* (PPV). Or, the *negative predictive value* (NPV) is defined as the number of true negatives divided by the number of true negatives and false negatives. The *specifity* (or true negative rate, TNR) is defined as the number of true negatives divided by the number of true negatives and false positives. And, the *sensitivity* (or true positive rate (TPR) or *recall*) is defined as the number of true positives divided by the number of true positives and the number of false negatives. Using training data to measure these statistical measures can lead to overfitting and result in poorly generalizable models. Therefore, testing data, i.e., using unseen data, should be used [13].

Generating test data appears to be trivial at the first sight. However, simply splitting the data into two sets might cause unwanted effects, e.g., introduce bias, or result in an inefficient usage of the available information. The test data generation process needs careful considerations in order to avoid these fallacies. In the dynamic data stream setting, plenty of data is available. The simple holdout strategy can be used without causing the problems mentioned in the batch setting. In contrast to the batch settings, large data sets for exact accuracy estimations can be used for testing without problems. The simplest approach is just holding out one (large) single reference data set during the whole learning (training) phase. Using this holdout data set, the model can be evaluated periodically. A graphical plot (accuracy versus number of training samples) is the most common way presenting results. Very often, the comparison of two algorithms is based on graphical comparisons by visualizing trends (e.g., accuracy) over time [6]. To obtain reliable results, statistical measures such as the standard error of results, are recommended. The statistical analysis should be accompanied by a comprehensive reporting scheme, which includes the relevant details for understanding and possible replication of the findings [23]. Only a few publications that perform an extensive statistical analysis are available [10]. Fortunately, the open source framework for data stream mining MOA is available and provides tools for an extensive experimental analysis [14].

Simulators for Data Stream Analysis. Random data stream simulators are a valuable tool for the experimental analysis of data stream algorithms. For our experiments in Section 4 a static data set, which contained pre-assigned class labels, was used to simulate a real-world data stream environment. The open source framework for data stream mining MOA [14] is able to generate a few thousand examples up to several hundred thousand examples per second. Additional noise can slow down the speed, because it requires the generation of random numbers.

4. A Simple Experiment in the EASD Framework

The Scientific Question. Before experimental runs are performed, the scientific question, which motivates the experiments, should be clearly stated. To exemplify the EASD approach, the following task is considered: Machine learning methods, which combine multiple models to improve prediction accuracy, are called *ensemble data mining algorithms*. Diversity of the different models is necessary to reach this performance gain compared to individual models. Each individual ML algorithm requires the specification of some parameters. Building ensembles requires the specification of additional algorithm parameters, e.g., the number of ensemble members. The scientific question can be formulated as follows: "How does the number of models to boost affect the performance of on-line learning algorithms?" To set up experiments, a specific algorithm (or a set of algorithms) has to be selected. Oza et al. presented a simple on-line bagging and boosting algorithm, OzaBoost [20]. The effect of the number of models to boost on the algorithm performance is an important research question, which will be analyzed in the following experiments.

Therefore, the experimental setup as well as the implementation details have to be specified further.

Implementation Details. Our observations are based on the *Massive On-line Analysis* (MOA) framework for data stream mining. The MOA framework provides programs for evaluation of ML algorithms [14, 6]. We will consider a typical classification setting, which can transferred into a regression setting without any fundamental changes. The model is trained on data with known classes. In the MOA classification setting, the following assumptions are made by Bifet et al. [6]:

(i) Small and fixed number of variables,

(ii) large number of examples,

(iii) limited number of possible class labels, typically less than ten,

(iv) the size of the training data will not fit into working memory,

(v) the available time for training is restricted,

(vi) the available time for classification is restricted, and

(vii) drift can occur.

We used *OzaBoost*, the incremental on-line boosting of Oza and Russel [21], which was implemented in Version 12.03 of the MOA software environment [14]. OzaBoost uses the following parameters: The classifier to train, l, the number of models to boost. s, and the option to boost with weights only, p. Experiments were performed in the statistical programming environment R [24]. The *sequential parameter optimization toolbox* (SPOT) was used for the experimental setup [3]. SPOT is implemented as an R package [4]. An additional R package, RMOA, was written to make the classification algorithms of MOA easily available to R users. The RMOA package is available on github (`https://github.com/jwijffels/RMOA`).

Empirical Analysis. The number of models to boost will be referred to as s in the following. In addition to s, the classifier to train will be modified as well. It will be referred to as l. Therefore, two algorithm parameters will be analyzed. The accuracy was used as a performance indicator. Using this setting, the scientific question can be concretized as the following research question: "How does the number of models to boost, s, affect the performance of the OzaBoost algorithm?"

Optimization Problem. After the algorithm was specified, a test function, e.g., an optimization problem, or a classification task, has to be defined. MOA provides tools to generate data streams. To keep the setup simple, we used the iris data set [12], which is a available as an R dataset [24].

Pre-experimental Planning. Before experimental runs are started, it is important to calibrate the problem difficulty to avoid floor- and ceiling effects. We will perform a comparison with a simple algorithm. If the simple algorithm is able to find the optimal solution with a limited computational budget, then the experimental setup is not adequate (too easy). If the simple algorithm is not able to find any solution, this may indicate that the problem is too hard. The naive Bayes classifier, which was used as the simple algorithm, obtained an accuracy of 91 percent. Because no floor- or ceiling effects were observed, we continue our experimentation with the OzaBoost algorithm.

Task and Experimental Setup. Two parameters of the OzaBoost algorithm were analyzed:

(i) the base learner, l, and

(ii) the ensemble size, s.

Hoeffding trees were used as base learners in our experiments [22]. In addition to the standard Hoeffding tree [10], a random Hoeffding tree was used in out experiments as a base learner. To be more specific, the categorical variable l was selected from the set {`HoeffdingTree`, `RandomHoeffdingTree`}, see Bifet et al. [7] for details. Values between one and one hundred were used for the ensemble size s.

Results and Visualizations. A comparison of the mean values from the two learners shows a significant difference: the first learner, i.e., `RandomHoeffdingTree`, obtained a mean accuracy of 0.66 (standard deviation (s.d.) = 0.06), whereas the mean accuracy of the second learner, i.e., `HoeffdingTree`, is 0.81 (s.d. = 0.18). The distributions of the accuracies are plotted in Fig. 2 and provide a detailed presentation of the results. Although the mean values of the two learners are different, the standard deviation of the `HoeffdingTree` learner approximately three times higher than the standard deviation of the `RandomHoeffdingTree`. This is reflected in the plots: the `HoeffdingTree` algorithm is not able to find an acceptable classification in some experimental runs.

Therefore, an additional analysis of the relationship between ensemble size and accuracy for the `HoeffdingTree` learner is of interest. We plot

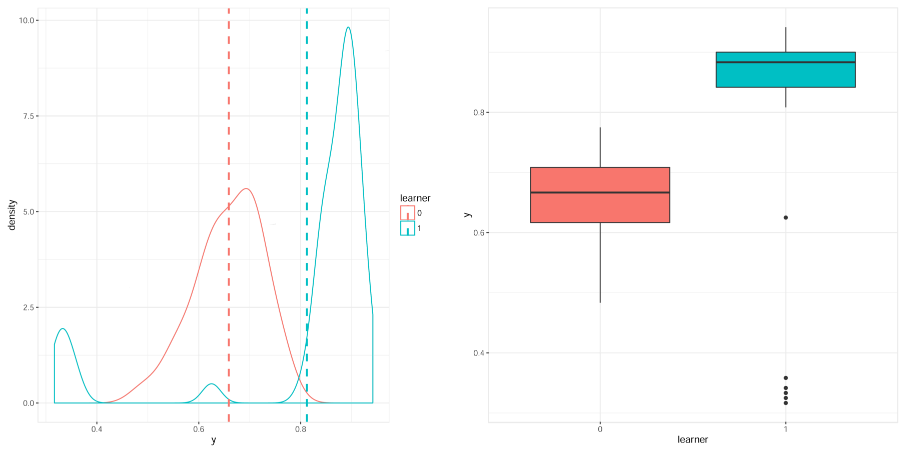

Figure 2: Comparison of the two learners. 0 = `RandomHoeffdingTree`, 1 = `HoeffdingTree`. *Left:* Density plots (accuracy, y). The dotted lines represent the mean values of the corresponding learners. *Right:* Boxplots (accuracy). Same data as in the panel on the left were used in the boxplots. The comparison of these two plots nicely illustrates strength and weakness of the plotting methods.

the results from the `HoeffdingTree` learner and add a smooth curve computed by `loess` (LOcal regrESSion) to a scatter plot [9]. `loess` fitting is done locally, i.e., the fit at a point x is based on points from a neighborhood of x, weighted by their distance from x. The result is shown in Fig. 3. This plot indicates that outliers occur if the sample size is small, i.e., $s < 60$.

Observations. The results reveal that the `HoeffdingTree` learner performs better (on average) than the `RandomHoeffdingTree` learner, but appears to be more sensitive to the settings of the ensemble size.

Discussion. The selection of a suitable base learner is important. Results indicate that too small ensemble sizes worsen the algorithm's performance. This statement has to be investigated further, e.g., by finding improved parameter settings for the OzaBoost learner. The sequential parameter optimization framework can be used for tuning the learner. A typical result from this tuning procedure is shown in the right panel of Fig. 3. In this plot, the accuracy, which was obtained by OzaBoost, is plotted against the number of iterations of the SPO tuner.

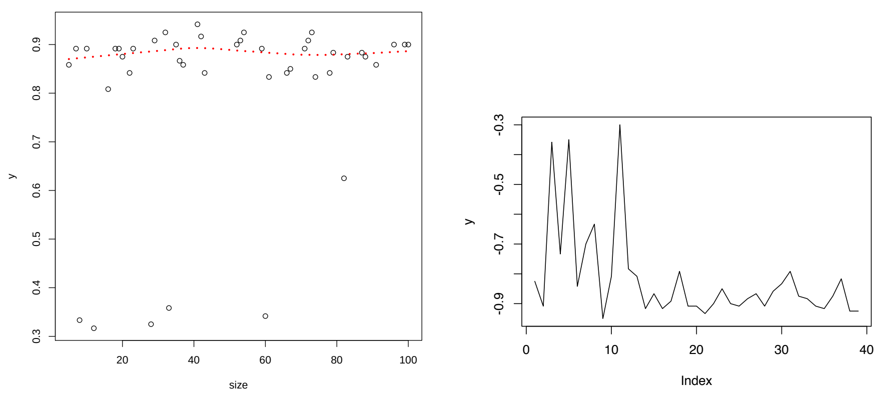

Figure 3: Left: Results, i.e., accuracy (y), obtained with the `HoeffdingTree` learner ($l = 1$) plotted against ensemble size (s). *Right:* A typical result from the parameter tuning procedure. Accuracy (y) is plotted against the number of algorithm runs (Index). The negative accuracy is shown, because the tuner requires minimization problems.

5. Summary and Outlook

An experimental methodology for the analysis of on-line data was presented. Differences between the traditional batch setting and the on-line setting were emphasized. Although useful, the actual practice of comparing run-time plots of on-line algorithms, e.g., accuracy versus time, should be complemented by more advanced tools from *exploratory data analysis* [25] and statistical tools, which were developed for the analysis of traditional algorithms. It was demonstrated, that statistical methods from experimental algorithmics can be successfully applied in the on-line setting. A combination of MOA, RMOA and the SPO toolbox was used to demonstrate the applicability and usefulness of standard tools from experimental algorithmics. The design and analysis of the algorithm were performed in the EASD framework. This report methodology, which is also described and exemplified by Preuss [23], is an integral part of the EASD framework.

References

[1] R. Barr, B. Golden, J. Kelly, M. Rescende, and W. Stewart. Designing and Reporting on Computational Experiments with Heuristic Methods. *Journal of Heuristics*, 1(1):9–32, 1995.

[2] T. Bartz-Beielstein, M. Chiarandini, L. Paquete, and M. Preuss (Eds.) *Experimental Methods for the Analysis of Optimization Algorithms.* Springer, Berlin,

Heidelberg, New York, 2010.

[3] T. Bartz-Beielstein, C. Lasarczyk, and M. Preuss. The Sequential Parameter Optimization Toolbox. In T. Bartz-Beielstein, M. Chiarandini, L. Paquete, and M. Preuss (Eds.) *Experimental Methods for the Analysis of Optimization Algorithms*, pages 337–360. Springer, Berlin, Heidelberg, New York, 2010.

[4] T. Bartz-Beielstein and M. Zaefferer. SPOT Package Vignette. Technical report, 2011.

[5] A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer. MOA: Massive Online Analysis. *The Journal of Machine Learning Research*, 11:1601–1604, 2010.

[6] A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer. Data Stream Mining. pages 1–185, May 2011.

[7] A. Bifet, R. Kirkby, P. Kranen, and P. Reutemann. *Massive online analysis - Manual*, Mar. 2012.

[8] G. Cattaneo and G. Italiano. Algorithm engineering. *ACM Computing Surveys*, 31(3):3, 1999.

[9] J. M. Chambers and T. J. Hastie (Eds.). *Statistical Models in S.* Statistical Models in S. Wadsworth and Brooks/Cole, Pacific Grove, CA, 1992.

[10] P. Domingos and G. Hulten. Mining high-speed data streams. *Proceedings of the the Sixth ACM SIGKDD International Conference*, pages 71–80, 2000.

[11] A. E. Eiben and M. Jelasity. A Critical Note on Experimental Research Methodology in EC. *Proceedings of the Congress on Evolutionary Computation (CEC)*, pages 582–587, 2002.

[12] R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(2):179–188, 1936.

[13] T. Hastie. *The elements of statistical learning : data mining, inference, and prediction.* Springer, New York, 2nd ed., 2009.

[14] G. Holmes, B. Pfahringer, P. Kranen, T. Jansen, T. Seidl, and A. Bifet. MOA: Massive Online Analysis, a Framework for Stream Classification and Clustering. *Proceedings of the International Workshop on Handling Concept Drift in Adaptive Information Systems in conjunction with European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD)*, pages 3–16, 2010.

[15] J. N. Hooker. Needed: An empirical science of algorithms. *Operations Research*, 42(2):201–212, 1994.

[16] J. N. Hooker. Testing Heuristics: We Have It All Wrong. *Journal of Heuristics*, 1(1):33–42, 1996.

[17] R. J. Hyndman and A. B. Koehler. Another look at measures of forecast accuracy. *International Journal of Forecasting*, 22(4):679–688, 2006.

[18] D. S. Johnson, C. R. Aragon, L. A. McGeoch, and C. Schevon. Optimization by Simulated Annealing: an Experimental Evaluation. Part I, Graph Partitioning. *Operations research*, 37(6):865–892, 1989.

[19] C. C. McGeoch. Experimental Analysis of Algorithms. PhD thesis, Carnegie Mellon University, Pittsburgh PA, 1986.

[20] N. C. Oza and S. Russell. Experimental comparisons of online and batch versions of bagging and boosting. *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 359–364, 2001.

[21] N. C. Oza and S. Russell. Online bagging and boosting. *Proceedings of the 8th International Workshop on Artificial Intelligence and Statistics*, pages 105–112, 2001.

[22] B. Pfahringer, G. Holmes, and R. Kirkby. New Options for Hoeffding Trees. In *AI 2007: Advances in Artificial Intelligence*, pages 90–99. Springer, Berlin Heidelberg, 2007.

[23] M. Preuss. *Multimodal Optimization by Means of Evolutionary Algorithms*. Natural Computing Series. Springer International Publishing, Cham, 2015.

[24] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2015.

[25] J. W. Tukey. *Explorative data analysis.* Addison-Wesley, 1977.

[26] C. J. Willmott and K. Matsuura. Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance. *Climate Research*, 30(7982):1–4, 2005.

[27] X. Zhu. *Knowledge Discovery and Data Mining: Challenges and Realities: Challenges and Realities.* Gale virtual reference library. Information Science Reference, 2007.