Cite this publication as: V. Gudivada, M. Irfan, E. Fathi, and D. Rao. "Cognitive Analytics: Going Beyond Big Data Analytics and Machine Learning". In: Cognitive Computing: Theory and Applications. Ed. by V. Gudivada, V. Raghavan, V. Govindaraju, and C. R. Rao. Vol. 35. Handbook of Statistics. New York, NY: Elsevier, Sept. 2016, pp. 169-205.

Chapter 5

Cognitive Analytics: Going Beyond Big Data Analytics and Machine Learning

V.N. Gudivada[*,1], M.T. Irfan[†], E. Fathi[*] and D.L. Rao[*]

[*]*East Carolina University, Greenville, NC, United States*

[†]*Bowdoin College, Brunswick, ME, United States*

[1]*Corresponding author: e-mail: gudivadav15@ecu.edu*

ABSTRACT

This chapter defines analytics and traces its evolution from its origin in 1988 to its current stage—cognitive analytics. We discuss types of learning and describe classes of machine learning algorithms. Given this backdrop, we propose a reference architecture for cognitive analytics and indicate ways to implement the architecture. A few cognitive analytics applications are briefly described. The chapter concludes by indicating current trends and future research direction.

Keywords: Cognitive analytics, Text analytics, Learning analytics, Educational data mining, Cognitive systems, Cognitive computing, Personalized learning, Data science, Machine learning, Big data analytics, Business analytics

1 INTRODUCTION

Cognitive computing is a computational environment which is comprised of (1) a high-performance computing infrastructure powered by special processors such as multicore CPUs, GPUs, TPUs, and neuromorphic chips; (2) a software development environment with intrinsic support for parallel and distributed computing, and powered by the underlying computing infrastructure; (3) software libraries and machine learning algorithms for extracting information and knowledge from unstructured data sources; (4) a data analytics environment whose processes and algorithms mimic human cognitive processes; and (5) query languages and APIs for accessing the services of the cognitive computing environment. We have defined cognitive computing in terms of its functions, since it is not easy to define it precisely and completely by other methods. Cognitive analytics draws upon the cognitive computing environment to generate actionable insights by analyzing diverse heterogeneous data sources using cognitive models that the human brain employs.

Classical symbolic and rule-based approaches to problems such as machine translation and speech-to-speech translation are being superseded by statistical learning approaches. For example, consider the problem of recognizing handwritten digits. Rule-based approaches entail developing a number of rules which aim to explicitly capture ways to write digits by different users. This results in too many rules. Furthermore, additional rules are needed to accommodate new users who might write a digit differently from the ways that were reflected in the current rule set. In contrast, artificial neural network (ANN) approaches use several small pieces of evidence in the form of features and combine them to produce higher-level features. ANN approaches are more robust as they perform better with data which is not seen in the training phase.

The ubiquity of big data (Gudivada et al., 2015a), abundant computing power, and the resurgence of neural network algorithms are providing *scalable solutions* to several difficult problems. The performance of newer approaches to problems that have been considered difficult for computers such as finding objects in images and classifying images rival human performance. For example, in the ImageNet Large-Scale Visual Recognition Challenge (Russakovsky et al., 2015), the error rate for some algorithms for detecting objects in video and scene classification is as low as 6%, whereas the error rate for humans is 5%. In another study with deep-learning algorithms (Goodfellow et al., 2014), Google reports 99.8% accuracy in recognizing CAPTCHA images on the hardest category of reCAPTCHA dataset. In another study at Facebook on the classification of images, Taigman et al. (2014) achieved an accuracy of 97.35% on the Labeled Faces in the Wild dataset using a nine-layer deep neural network. Finally, Lake et al. (2015) describe an approach called *Bayesian Program Learning*, which is used to recognize 1623 handwritten character sets from 50 languages with only a limited training. Though the above problems are diverse, deep neural network algorithms perform exceptionally well in all these domains.

The above approaches coupled with advances in information retrieval, natural language understanding (Gudivada et al., 2015b), artificial intelligence (AI), and machine learning are helping to usher in a new paradigm for strategic decision making. The term *data analytics* when used in a generic sense refers to any actionable information that results from computational analysis of data using mathematical and statistical methods. Data analytics is an interdisciplinary domain encompassing mathematics, statistics, and computer science. Implicitly, there is a domain associated with data analytics. The domain provides the data for analysis. The primary goal of data analytics is to gain insights into a process or problem so that the latter can be improved or solved. In other words, analytics is a data-driven approach to decision making and problem solving.

Though certain types of analytics are common across various application domains, they tend to vary significantly from domain to another. This has led to the proliferation of names such as business analytics, text analytics, image analytics, video analytics, graph analytics, spatial analytics, visual analytics, and cognitive analytics. However, irrespective of the domain, data analytics is comprised of three components: data acquisition and loading, methods and algorithms, and a computational platform that implicitly embodies workflows and best practices. The data acquisition and loading components enable the preparation of input data and loading it into the computational platform. Various algorithms and approaches for data analysis are provided by the methods and algorithms component. Lastly, the computational platform brings everything together as a system and provides interfaces for users and other applications to interact with it.

From a functional perspective, there are three categories of data analytics: descriptive, prescriptive, and predictive. Descriptive analytics provides a dashboard view of the current state of a system or process. It uses descriptive statistics and machine learning algorithms to provide insight into a system. The insight often reveals, in a process, for example, various steps in the process, how the steps are sequenced, what type of resources are consumed, and how much time is spent in each process. As another example, readability of English texts is determined by text analytics such as the Fry readability formula, Automated Readability Index, Flesch-Kincaid, Gunning-Fog, Coleman-Liau Index, and SMOG Index. Software metrics and measurements are analytics used to characterize properties of software. Such metrics include the number of classes, number of methods per class, depth of inheritance tree, number of interfaces, and total lines of code.

Prescriptive analytics is a natural outcome of descriptive analytics. It suggests ways to improve a current process or system using simulation and optimization algorithms. In the case of software metrics and measurements, prescriptive analytics specifies a range of values for each measurement such as bounds for number of methods in a class. Furthermore, it specifies refactoring techniques if a measurement is not within the specified range.

Predictive analytics enables answering "what-if" questions by building predictive models using inferential statistics and forecasting techniques. It enables organizations to make data-driven strategic decisions. Predictive models are built using the operational and historical data. They extract associations and other implicit relationships in the data to build the models. Various regression models such as linear, logistic, Lasso, ridge, Cox proportional hazards, and Bayesian are widely used. Logic regression, for example, is used in clinical trials and fraud detection to associate a probability with a binary outcome.

Like cognitive computing, cognitive analytics is pursued from two complementary perspectives. The first is driven by the computer science researchers in both industry and academia. Advances in big data, cloud computing, natural language understanding, and machine learning are enabling extraction of knowledge from vast repositories of unstructured data such as natural language text, images, video, and audio. From this group's perspective, the knowledge extracted from the unstructured data coupled with statistical inference and reasoning distinguishes cognitive analytics from business analytics. The second perspective is advanced by cognitive and neuroscience researchers. They employ theories of mind, functional areas of the brain, and cognitive models and processes. For example, an approach in this class might gather analytics about a cognitive process to validate the cognitive model as well as to improve the model (Chakraborty et al., 2014).

1.1 Chapter Organization

The overarching goal for this chapter is to present a unified approach to the emerging area of cognitive analytics. More specifically, in Section 2, we trace the evolution of data analytics and discuss central issues. Types of learning used in cognitive analytics are described at a conceptual level in Section 3. In Section 4, we discuss the following classes of machine learning algorithms: logistic regression, decision trees, support vector machines (SVMs), Bayesian networks (BNs), neural networks, and deep learning. This section also includes a discussion on machine learning frameworks and libraries.

We propose a reference architecture called Cognalytics for cognitive analytics in Section 5. This section also indicates how this architecture can be implemented using open source tools. Section 6 presents applications of cognitive analytics including learning analytics (LA), personalized learning, cognitive businesses, brain–computer interfaces (BCIs), and assistive technologies. Cognitive analytics trends and future research directions are described in Section 7. Section 8 concludes the chapter.

2 EVOLUTION OF ANALYTICS AND CORE THEMES

AI is a subfield of computer science and machine learning is a major area within AI. The recent emergence of big data and cloud computing created AI renaissance. The attendant media coverage of machine learning is making the latter a household name. This is also creating confusion and propagation of misinformation. In blogs and other self-published forums, some authors have declared AI and Computer Science as two distinct disciplines, likewise, AI and machine learning. The scope and the meaning of the term *analytics* are being reinvented.

You cannot manage what you do not measure is an old adage from the management world that is still true today in most organizations and academic disciplines. At the core of analytics are data, mathematical and statistical models built using this data. The types of data needed and the type of processing performed, and the variety of models built varies. The models are used for a broad range of purposes under the umbrella terms descriptive analytics, prescriptive analytics, and predictive analytics. AI, machine learning, distributed computing, and high-performance computing comprise the computational infrastructure to manage data and enable model building.

2.1 Multiple Perspectives

There exists multiple perspective on analytics. The Computer Science perspective is driven by technical considerations related to storing, managing, and querying data. In the early days, there was limited support for analysis. The business perspective views analytics from an organizational level and focuses on actionable insights into data. Visual analytics is a new area of analytics whose goal is analytical reasoning through interactive visual interfaces. Even more recently, other terms such as educational data mining (EDM), LA, and cognitive analytics have emerged.

Academia has responded to this unprecedented interest in analytics by creating new *interdisciplinary* degree programs primarily at the master's level. These programs fall into three categories: (1) programs that have the term analytics somewhere in their name—business analytics, health informatics, health care informatics, and nursing informatics. Other degree programs such as econometrics also fall under this category, though they do not explicitly use the term informatics in the name. These programs are typically administered or led by noncomputer science departments; (2) programs with names such as master of science in analytics and master of science in data science. These programs are typically led by computer science departments; and (3) numerous graduate certificates, tracks, and concentrations in analytics, data mining, knowledge discovery, machine learning, and big data.

2.2 Analytics Evolution

We trace the evolution of analytics from a Computer Science perspective as shown in Fig. 1. Basic analytics functions were part of the relational database management systems (RDBMS) from their early years. RDBMS served as *operational databases* for conducting day-to-day business transactions—online transaction processing (OLTP). Basic functions for descriptive statistics were provided. In subsequent years, more sophisticated functions were introduced under the name Statistics & SQL Analytics. They included functions for ranking of results, moving and cumulative aggregating values over a range of rows, lag and lead to access data from preceding and following rows, descriptive statistics, correlations, and linear regression. In the early days of RDBMS, analytic functions were implemented outside of the RDBMS system. Each analytic function was implemented by a standalone piece of code which made code optimization across RDBMS and analytic functions difficult. Recently, there have been efforts in implementing analytic functions within the database (Feng et al., 2012).

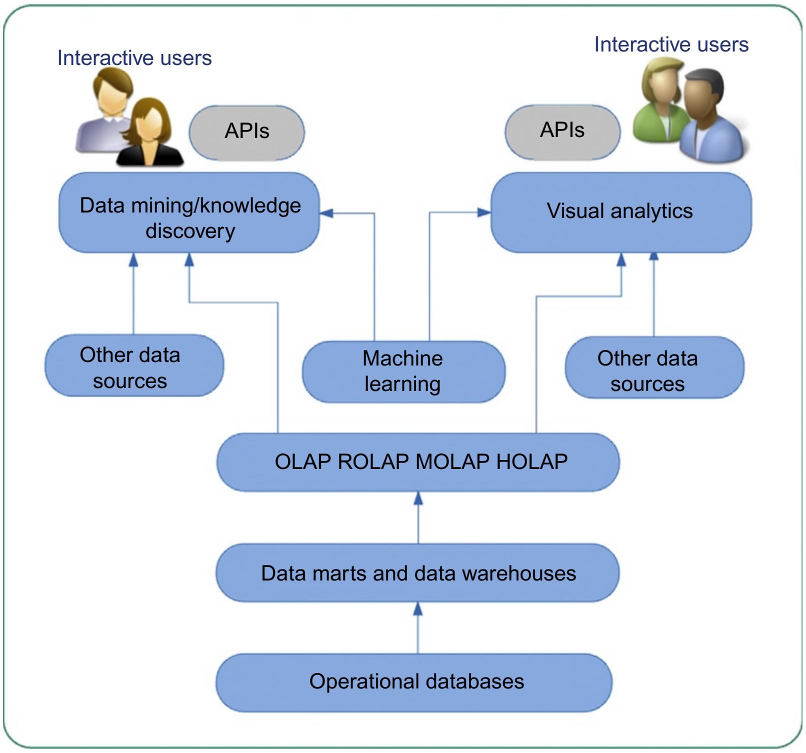

FIG. 1 Evolution of analytics.

2.3 Data Warehouses and Data Marts

The next stage in the evolution is the packaging of more advanced analytic functions into database systems for data marts and data warehouses. The latter are developed to help make data-driven strategic decisions—online analytical processing (OLAP). The terms *data mart* and *data warehouse* are sometimes used incorrectly as synonyms. A data warehouse is a consolidated and centralized repository of data extracted from various operational databases and other disparate data sources. A data mart, on the other hand, is a subset of a data warehouse, which targets the needs of a specific department of an organization. The data warehouse is like an enterprise *database schema*, whereas a data mart is akin to a *database view*. Data warehouses and data marts are used for generating customer and compliance reports, score cards, and dashboards. They are also used for planning, forecasting, and modeling. Extract, Transform, and Load (ETL) is a set of tools and processes that are used to design and implement data warehouses and data marts.

Both OLTP and some OLAP systems use SQL for performing analytic functions. SQL operators such as CUBE, ROLLUP, and GROUPING SET were specifically introduced for data analytics with data warehouses. An *OLAP cube* is a multidimensional data array, which is a generalization of a 2D or 3D spreadsheet. It can also be viewed as a logical structure that defines metadata. MDX (multidimensional expression) is a metadata-based query language to query OLAP cubes. Analytic operations on OLAP cube include slice (creating a new cube with fewer dimensions), dice (creating a new (smaller) cube by stating specific values for cube dimensions), drill down and drill up (navigating between the most detailed data level to the summarized data levels), roll-up (summarizing data along a specific dimension), and pivot (rotating the cube to view its various dimensions or faces).

2.4 ROLAP, MOLAP, and HOLAP

The third stage in the evolution is the emergence of ROLAP, MOLAP, and HOLAP. All three classes of cubes organize data in a way to enable efficient dimensional analysis. The first step in building a cube is to determine dimensions. For a sales department cube, for example, geographic region and industry classification are two dimensions. The next step is to determine the levels of data aggregation for each dimension. For the geographic region dimension, data aggregation levels include county, state, region, country, and continent. If the industry classification is energy utilities, data aggregation levels include natural gas, coal-powered electricity, wind, and solar.

ROLAP, MOLAP, and HOLAP are extensions of OLAP and are referred to as OLAP servers. A relational OLAP (ROLAP) server is an interface between an RDBMS warehouse and OLAP users. It implements navigational logic for the cube, sends SQL queries for execution to the underlying warehouse, and provides additional tools and services. ROLAP servers tend to suffer from performance since data needs to be fetched from the warehouse on the fly.

In contrast with ROLAP, MOLAP cubes extract data a priori from a warehouse and store the data in the cube itself. All the calculations are precomputed at the time of the cube creation. This contributes to superior performance but limits the amount of data handled by the MOLAP cube. Also, MOLAP consumes additional storage space. HOLAP is a hybrid server which combines best of both ROLAP and MOLAP. HOLAP is scalable like ROLAP and provides superior performance like MOLAP.

2.5 Data Mining/Knowledge Discovery

Independent of analytics evolution, machine learning (ML) emerged in parallel as a subdiscipline of AI. The majority of machine learning algorithms fall into the following broad categories: decision trees, associative rule learning, genetic algorithms, refinement learning, random forest, SVMs, BNs, neural networks, and deep learning.

The next stage in the analytics evolution is the emergence of data mining (aka knowledge discovery). Data mining is a synergistic confluence of databases, statistics, AI, and ML. Its goal is to find anomalies and discover hidden patterns and correlations in data to enable the generation of actionable intelligence. Such intelligence has been used to increase revenues, improve customer relationship, reduce operating costs, and make strategic decisions. A significant task in data mining is locating the relevant data and preparing the data for ingestion into ML algorithms.

2.6 Visual Analytics

Visual analytics is a relatively new field and developed independently of data mining. Like data mining, it draws data from several sources including RDBMS, OLAP cubes, and other sources such as social media. Visual analytics combines automatic and visual analysis methods with human interactive exploration. It is based on the premise that combining the quantitative capabilities of computers and the cognitive capabilities of humans leads to powerful ways to create new knowledge. Interactive exploration and visual manipulation play a central role in visual analytics. Both data mining and visual analytics systems are available as cloud services. Their functionality is accessed through APIs.

2.7 Cognitive Analytics

Cognitive analytics is the natural evolution of both data mining and visual analytics. Cognitive analytics removes humans from the loop and is completely automated. It is in a formative stage now and there is tremendous interest from both industry and academia. However, industry is primarily driving both research and development. Cognitive analytics draws upon advances in several areas and combines the computing and cognitive science approaches. Fig. 2 shows a functional view of cognitive analytics. Data for cognitive analytics comes from several sources and includes structured, semistructured, and unstructured data. Furthermore, it employs knowledge structures such as taxonomies and ontologies to enable reasoning and inference. Extraction of both low-level features and high-level information is crucial to cognitive analytics.

Shown in the wide rectangle in Fig. 2 are internal components of cognitive analytics engine. Various knowledge representation structures are needed to represent and reason with knowledge. An assortment of machine learning algorithms and inference engines are also needed. The domain cognitive models capture domain-specific cognitive processes to enable cognitive style problem solving. The learning and adaptation component improves system performance by learning from previous interactions with the users.

In contrast with all other analytics, cognitive analytics generates multiple answers for a question and assigns a degree of confidence to each answer.

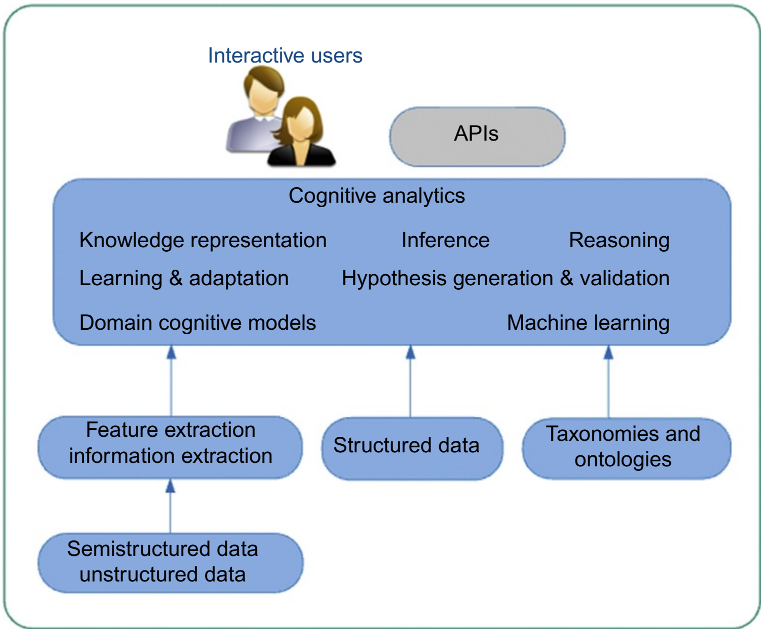

FIG. 2 Conceptual view of cognitive analytics.

In other words, cognitive analytics uses probabilistic algorithms to come up with multiple answers with varying degrees of relevance. Noncognitive analytics, in contrast, employ deterministic algorithms and compute only one answer for any question. Computing multiple answers requires another component, which is labeled Hypothesis Generation & Validation. This component is pioneered by IBM and is responsible for generating multiple hypotheses for a question, gathers evidence for each hypothesis, and using the evidence scores the relevance of a hypothesis as an answer to the question.

In summary, analytics comes in many forms with varying functional capabilities. Each form reflects the underlying technologies and the characteristics of the domain which propels the form. Regardless of these differences, we can forge a generic architecture for cognitive computing. Implementation of such an architecture requires a platform with the following characteristics: infrastructure for data cleaning, transformations, and fusion; a set of both deterministic and probabilistic algorithms for computing analytics; a learning component powered by a domain cognitive model; an array of machine learning frameworks for hypothesis generation, evidence gathering, and scoring hypotheses; and a high-performance computing system with scalability, performance, and elasticity. In Section 5, we propose a reference architecture for cognitive analytics and discuss ways to implement the architecture.

3 TYPES OF LEARNING

There are two broad classes of learning: supervised and unsupervised. Supervised learning involves learning from examples, which is a set of associations between inputs and outputs. This is akin to how children learn to read and write—a teacher presents letters of the alphabet and utters the corresponding sounds. Repeating the process with the same examples will gradually train students' biological neural networks to associate symbols with sounds.

Training data is comprised of two parts: input and expected output. Let (i, o) be an item in the training dataset, which specifies that when the system is presented with input i, it should output o. The training data is a set of n such pairs: $\{(i_1, o_1), (i_2, o_n), \ldots, (i_n, o_n)\}$. A trained model should work correctly if the examples in the training set are given to the model again. For example, given i as input to the trained model, it should produce o as the output.

A reasonable criterion should be defined to measure the error between the correct output and the outcome generated by the model. The main task of supervised learning is to minimize the error function. This is similar to how the teacher corrects students in their initial attempts to read or write, which gradually minimizes the error function of their biological neural network model. In addition to the error function, other characteristics of the model include the number of model parameters and modeling flexibility (Battiti et al., 2008). Decision trees, neural networks, regression, and Bayesian classification are examples of supervised learning algorithms.

Unsupervised learning algorithms draw inferences from datasets consisting of just input data without labeled responses. Unsupervised learning infers a function which describes the hidden structure from unlabeled data. Since the examples given to the learner are unlabeled, there is no error or reward signal to evaluate a potential solution. This distinguishes unsupervised learning from supervised learning and reinforcement learning (see Section 3.2). K-means clustering, genetic algorithms, and simulated annealing are examples of unsupervised learning algorithms.

For cognitive analytics, unsupervised algorithms have an edge over the supervised ones. In the big data context, we do not know the patterns in the data *a priori*. Furthermore, training data may not be available. Unsupervised learning algorithms are better suited for this scenario. Unsupervised learning algorithms are also used to automatically generate test data. The latter is employed to train supervised learning algorithms. For complex question-answering (Q/A) environments such as the Jeopardy! game, several hypotheses are generated as candidate answers; evidence is gathered and used to rank hypotheses. In such Q/A environments, it is advantageous to use supervised learning to generate some hypotheses and unsupervised learning to generate additional hypotheses. This approach benefits from both types of learning and the resulting system performs more robustly. There are many applications such as real-time fraud detection, continuous security vulnerability assessment, computer vision, and natural language understanding for which unsupervised learning suits well.

3.1 Active Learning

Active learning is a special case of semisupervised learning. The key hypothesis of active learning is to allow the learning algorithm to choose the data from which to learn. In other words, the learning algorithm can interactively query the user (or another information source) to obtain the desired output(s) for the input instance on hand. This is expected to result in better performance as well as less training. The advantage of active learning over supervised learning is that the former obviates the need for thousands of labeled instances for training (Settles, 2009). This is especially important for cognitive analytics where unstructured data is abundant, but has no labels.

Active learning is also called query learning and optimal experimental design. Algorithms are needed to determine which data points should be labeled with the expected result. These algorithms are also known as query strategies and include uncertainty sampling—label only those points for which the current model is least certain about the correct output; query by committee–label those points for which the *committee* disagrees the most, where the committee is comprised of a variety of models which are trained on the current labeled data; expected model change—label those points which would change the current model the most; expected error reduction—label those points which would reduce the model's generalization error the most; and many more.

3.2 Reinforcement Learning

Humans and theoretical systems have various types of learning in common. Learning by imitating a teacher is the most standard but not the only way of transferring knowledge. In real life, we notice the outstanding tendency of children to try dangerous things such as placing fingers in an electrical receptacle without a guiding teacher. Depending on the outcome or experience, a child may repeat this activity again or may never repeat it. This type of learning is called *reinforcement learning*. The latter is a form of supervised learning.

Reinforcement learning is inspired by behaviorist psychology. The latter is concerned with how to take actions by an agent in an unknown environment to maximize some notion of cumulative rewards. In the example of bicycle riding, positive rewards can be in the form of admiring friends and negative ones can be injuries to biological tissues. But after some trials with the goal of maximizing the positive rewards, learning occurs (i.e., can ride the bicycle now). Initially, the system is not explicitly trained and receives feedback on its performance when it is in production. In a sense, reinforcement learning is trial-and-error learning.

A reinforcement learning environment is formulated as a Markov decision process (MDP). Many reinforcement algorithms use dynamic programming techniques. These algorithms do not need the knowledge about the MDP. When exact methods become infeasible, these algorithms target large MDPs. The basic reinforcement learning model is comprised of (a) a set of environment states S, (b) a set of actions A, (c) stochastic rules that govern state transitions, (d) rules to compute the immediate reward of a transition, and (e) rules to describe an agent's observations. Reinforcement learning is especially well suited to problems which include a long- vs short-term reward trade-off as in elevator scheduling, and robot control.

Reinforcement learning serves as a theoretical tool for investigating the principles of autonomous agents learning to act in their environment. The agents strive to improve their behavior through their interactions and experiences with other agents in the environment. Reinforcement learning has been used as a practical computational tool for designing and constructing autonomous agents for domains such as robotics, combinatorial search problems, and industrial manufacturing.

3.3 Ensemble Learning

Ensemble learning is based on multiple learning models, which are strategically generated, and optimally combined for solving problems such as classification (Polikar, 2009). The idea behind this is that two minds are better than one. Also, to make strategic decisions, we solicit input from multiple sources and combine or rank the sources. An ensemble itself is a supervised learning algorithm. Ensemble learning systems are also called multiple classifier systems.

Ensemble algorithms yield better results if there are significant differences or diversity among the models. For example, more random decision trees lead to a stronger ensemble than entropy-reducing decision trees. However, selecting a range of strong learning algorithms, though they may significantly diverge, is the key for good performance. Common types of ensembles include (a) Bayes optimal classifier—is an ensemble of all the hypotheses in the hypothesis space, (b) Bayesian parameter averaging—approximate the Bayes Optimal Classifier by sampling hypotheses from the hypothesis space and combining them using Bayes' rule, (c) Bootstrap aggregating (bagging)—building multiple models, typically of the same kind, from different subsamples of the training dataset, (d) Boosting—building multiple models, typically of the same type, each model learns to fix the prediction errors of a prior model in the chain, (e) Stacking—building multiple models, typically of differing types, and having a supervisor model that learns how to best combine the predictions of the primary models, and (f) Bucket of models—a model selection algorithm is used to choose the best model for each problem, and the model selection is based on cross-validation.

4 MACHINE LEARNING ALGORITHMS

In this section, we review, compare, and contrast several standard machine learning (ML) algorithms that have been widely used in cognitive analytics. Before we delve into the specifics of the algorithms, we discuss considerations that are common to machine learning algorithms in general.

Input, output, and model: The input dataset to a machine learning algorithm is usually composed of many rows, where each row is called an *example*. Each example represents one *data point* and is composed of multiple *feature values* and optionally some *target values*. All feature values of an example collectively form a *feature vector*. All the examples in the dataset usually have the same number of elements in their feature vectors and also the same number of target values. The feature vector gives a quantitative representation of the example in terms of its *features*. Finding "good" features is extremely important and is more of an art than science. The target values, if present, ascribe a *labeling* to the example. The two mainstream classes of machine learning algorithms—supervised and unsupervised (see Section 3)—differ due to the presence or absence of the target values in an input dataset. The output of a machine learning algorithm is concerned with predicting the target values for a new feature vector.

The most challenging part of a machine learning algorithm is choosing an underlying model for mapping feature vectors to target values. These models are usually predictive and very rarely are explanatory. The models also have parameters that must be instantiated using the input dataset, and this process is called learning. The difficulty in choosing a model stems from the fact that often an infinite number of candidate models exist, even if the class of models is restricted. Choosing a model from the candidate set involves a balancing act which we discuss next.

Classification vs regression: We use the terms *target values* and *labels* interchangeably. A classification problem involves assigning a class label for a given feature vector. For example, an email message is classified as *spam* or *not spam*. The terms *spam* and *not spam* are the class labels. In other words, the output is a (class) label. In other examples such as weather prediction, the target value is a scalar value—the probability of a weather event. Target values can also be vectors.

Based on the type of target values desired, a machine learning algorithm deals with either a *classification* problem or a *regression* problem. The main difference between the two lies in a discrete vs continuous range of target values. In a classification problem, there are two or more discrete classes. Each example belongs to one of these classes. In a large majority of classification problems, the class labeling of the input examples is specified. The main objective of a classification algorithm is to accurately predict class labels for previously unseen examples. Classification problems of this type where the input examples are labeled fall into the supervised learning category (see Section 3). In contrast, unsupervised classification problems start with unlabeled input examples. The job of a classification algorithm in this case is to predict which examples belong to the same class. The interpretation of the classes in the unsupervised classification scenario is provided by human domain experts. Regression problems share the same problem structure as classification problems, with the key difference that the target values are no longer discrete labels.

Measuring predictive performance: Prediction is the prime goal of machine learning algorithms in general. Therefore, the algorithms are evaluated in terms of their predictive performance. Several technical issues arise while measuring the predictive performance, which are referred to as *overfitting* and *underfitting*.

Consider a supervised classification problem, where a set of labeled examples are available in the input dataset. Following is a standard mechanism for applying a machine learning algorithm to the dataset. First, the input dataset is partitioned into three nonoverlapping subsets—training, validation, and test. The size of these three sets is a design choice. The *training set* is used during the *training phase* to help instantiate the parameters of the model so that the model predicts well. How well the model predicts is measured using the *validation set*. The latter provides a set of examples that the algorithm has not seen during the training phase and is used to select a model that may not perform the best with respect to the training set but performs very well with respect to previously unseen examples. There is an important reason for measuring the accuracy of the model using the validation set instead of the training set. Imagine that the model does a perfect job predicting target values given feature vectors of examples in the training set.

On the surface, it may seem like perfect learning. However, it could very well be the case that the model simply "memorized" the training examples. In that case, given a previously unseen example, the model will most likely perform poorly. This is known as *overfitting*. Therefore, the main goal of the validation set is to prevent overfitting by choosing a model that may not be perfect with respect to the training examples, but does the best job when faced with unseen examples. In the last phase, the *test set* is used to measure the accuracy of the algorithm, which can differ from training and validation accuracies. Note that the test set is used only for unbiased evaluation of the model.

Underfitting is the opposite of overfitting. Underfitting means that the model is not sophisticated enough to capture the richness of the data. Large error margins for both the training and validation sets indicate underfitting, while very low error margin for the training set and very high error margin for the validation set indicate overfitting. Both overfitting and underfitting are undesirable and one of the challenges of machine learning lies in finding the sweet spot between them.

The subsequent discussion in this section will primarily focus on supervised classification problems, which are most prevalent in the real world.

The reader may consult Murphy's book (2012) for a complete coverage of machine learning algorithms.

4.1 Logistic Regression

Logistic regression is essentially a classification algorithm. The word "regression" in its name comes from its close sister in the regression domain known as *linear regression*. Given that the classes are discrete in supervised classification problems, the goal for the algorithms is to find the *decision boundaries* among the classes. Decision boundaries separate examples of one class from another. Depending on the problem instance, decision boundaries may be complex and nonlinear in geometric shape. In general, different machine learning algorithms have different assumptions regarding the shape of decision boundaries. In the case of logistic regression, the assumption is that decision boundaries are linear. That is, they are hyperplanes in the high-dimensional feature space, where the dimension of the feature space is simply determined by the number of elements in the feature vector of a training example.

The logistic regression model parameters are roughly the weights for the features. Each weighted feature vector is mapped to a value between 0 and 1 via the S-shaped logistic function. This value is interpreted as the probability of an example belonging to a particular class. The learning algorithm tunes the weights in order to correctly classify the training examples. The issue of avoiding overfitting inevitably arises here. The gradient descent method and several variants of it are popular for tuning the weights. Once the weights are chosen, the logistic function is applied to any unseen example to obtain the probability of it belonging to a class.

Due to the simplistic assumption of linear decision boundaries, logistic regression is often times the *first go-to* algorithm for classification problems. Also, because of the linear, noncomplex decision boundaries, logistic regression is known to be less prone to overfitting. Intuitively, overfitting occurs when we try to correctly classify every single training example by arbitrarily wiggling the decision boundary. Additionally, gradient descent typically works very fast and thus makes the training phase of logistic regression quick. All of these advantages justify the popular application of logistic regression to a variety of classification problems. On the down side, however, the simplistic modeling assumptions may lead to underfitting for rich and complex datasets.

Logistic regression has been used in a variety of application areas. Honorio and Ortiz (2015) has used it to learn the structure and parameters of a social network model that captures the strategic behavior of individuals. The model has been used to find the most influential individuals in a network (Irfan and Ortiz, 2011, 2014). Logistic regression has also been used in GIS (Ayalew and Yamagishi, 2005; Lee, 2005), email spam filtering (Chang et al., 2008), and other problems within natural language processing

(Jurafsky and Martin, 2009; Nadkarni et al., 2011), speech recognition (Jurafsky and Martin, 2009), finance (Laitinen and Laitinen, 2001; Maher and Sen, 1997), and the broader domain of pattern recognition (Bishop, 2006).

4.2 Decision Trees

The Classification and Regression Tree (CART) method was originally presented by Breiman et al. (1984) during the 1980s. This has led to tremendous interest in decision tree learning. In the supervised classification setting, the objective of decision tree learning is to compute a special type of tree that can classify examples to classes. The notions of training, validation, and test sets as well as overfitting vs underfitting issues apply for decision trees too. The underlying model in decision tree learning is a tree in graph-theoretic sense. However, we must also recognize a stylized control flow that is superimposed on the tree structure. Each internal node of the tree, including the root, asks a decision-type question. Based on the answer for an example, we next traverse one of the children of that internal node. Once we reach a leaf node, we are certain to know the classification of the example according to the decision tree, since each leaf node is annotated with a class label.

In addition to CART, there are many other learning algorithms for finding the "best" tree for a classification problem. Most modern algorithms like Iterative Dichotomiser 3 (ID3) (Quinlan, 1986) and its successors C4.5 (Quinlan, 2014) and C5 (Quinlan, 2016) use information theoretic measures, such as entropy, to learn a tree. Entropy can be thought of as a measure of uncertainty. Initially, the whole training set, consisting of examples of different classes, will have a very high entropy measure. ID3 and its successors repeatedly partition the training set in order to reduce the sum of the entropy measures of the partitions. Usually, a greedy strategy is employed for this purpose. The algorithm chooses a feature and partitions the training set based on that feature. The feature is chosen with the goal of minimizing the sum of the entropy measures across the resulting partitions. The same procedure is recursed on each partition, unless all the examples in that partition belong to the same class.

One big advantage of decision tree learning over other learning methods such as logistic regression is that it can capture more complex decision boundaries. Decision tree learning is suitable for datasets that are not linearly separable—there exists no hyperplane that separates out examples of two different classes. The ability of decision trees to capture complex decision boundaries is sometimes its own pitfall, since this can lead to overfitting unless certain other techniques like "pruning the tree" is applied.

A few other advantages have made decision trees popular. First, they often lead to a clear visualization of how the machine learning algorithm performs classification. Second, the training phase is usually fast and scalable to large-scale data. Lastly, decision trees have been widely used in various ensemble learning methods, such as AdaBoost (Freund and Schapire, 1995;

Freund et al., 1999) and random forests (Breiman, 2001; Ho, 1995). Random forests belong to a broader umbrella of machine learning techniques known as *bagging*. Bagging techniques are especially geared for tackling overfitting. In random forests, multiple decision trees are learned, which collectively build a graph-theoretic forest. A new feature vector is classified differently by different decision trees in the forest. These individual classifications are aggregated to output the final classification.

4.3 Support Vector Machine

SVM is one of the most widely applicable machine learning algorithms (Bell, 2014; Shalev-Shwartz and Ben-David, 2014). Since Vapnik and Chervonenkis presented SVM during the 1960s, there has been a tremendous amount of work extending it in multiple directions. We will present the key idea behind SVM and its advantages. Schölkopf's book (1999) is a comprehensive reference on this topic.

Consider a restricted classification setting where the training set consists of only examples that belong to two classes, and the examples are linearly separable. Because of linear separability assumption, there exist hyperplanes that separate out the examples of the two different classes. In fact, there exist an infinite number of such hyperplanes. The central idea in SVM is to choose that particular hyperplane which sits "right in the middle" in between the examples of the two classes. Mathematically speaking, SVM chooses the hyperplane that maximizes the minimum distance between that hyperplane and all the examples. In other words, the hyperplane is equidistant from the examples of the two classes that are closest to it. In SVM terminology, two times the distance between the hyperplane and the points closest to it is known as the *margin*. As a result, SVM is also known as a maximum margin classifier. Maximizing the margin, or equivalently selecting that particular hyperplane in between the examples of the two different classes, is extremely significant. This leads to a good generalization for classifying previously unseen examples.

One of the reasons SVM is so widely applicable is that it can be easily extended to complex instances that are not linearly separable. This is done by mapping the training examples to a higher-dimensional space where they become linearly separable in conjunction with the *kernel trick* (Aizerman et al., 1964; Boser et al., 1992) to keep computation manageable.

Another reason in favor of its applicability is a subtle issue that accounts for the name "support vector." Not all training examples are equally important. In fact, since the decision boundary only depends on the training examples closest to it, it suffices to define the underlying model of SVM in terms of only those training examples. Those examples are called support vectors. Although the original dataset may contain a very large number of examples, the number of support vectors is usually very small. This makes SVM amenable to large-scale data, including streaming data. SVM is also memory efficient for many applications.

SVM algorithms have been successfully used for classifying images in extremely large repositories such as Instagram. They have also been used to analyze natural language text and web documents (Tong and Koller, 2001). In the medical domain, SVMs have been used to classify proteins into their functional family (Cai et al., 2003).

4.4 Artificial Neural Networks and Deep Learning

ANNs, or simply neural networks, belong to the broader class of biologically inspired computation. ANNs are modeled after how the neurons in the brain "fire" and how one neuron's firing affects other neurons connected to it. One of the earliest and most influential models of a neuron is attributed to McCulloch and Pitts (1943), who combined biology and mathematics to model the firing of a neuron as a threshold function. Subsequently, Rosenblatt (1958) presented the first algorithm, known as the *perceptron* algorithm, to learn the parameters of the simplest type of neural networks that can successfully deal with linearly separable classification problems.

Advances in high-performance computing and algorithmic progress advanced ANNs to address problems where the class boundaries are not linearly separable. This led to a further explosion of growth and interest in neural networks during the 1980s. In fact, many considered ANNs as a "one size fits all" framework, which ultimately created its own downfall. In particular, Geman et al. (1992) showed that neural networks are prone to overfitting and underfitting problems. Furthermore, they demonstrated that for a neural network to be effective for various problem instances, it needs to be complex and enough data is needed to effect good learning.

There are many variants of neural networks, but we will discuss the most prevalent one known as *feed-forward neural network*. A feed-forward network consists of neurons placed in multiple layers. The first layer is known as the *input layer*, and the last layer the *output layer*. All the layers in between input and output layers are *hidden layers*. The outputs of neurons in one layer are fed as inputs to the neurons of the next layer. The parameters of the model are the weights of the connections between neurons of two consecutive layers and a threshold value for each neuron. The weights indicate connection strengths and the threshold value determines whether or not a neuron fires.

Given a training dataset, the neural network is designed in such a way that the number of neurons in the input layer is equal to the number of features. Also, the number of neurons in the output layer is equal to the number of target values. Apart from these restrictions, there is no hard and fast rule regarding the number of hidden layers and the number of neurons in a hidden layer. Often times, these are determined by experimenting with several different network architectures and choosing one based on cross-validation

(Murphy, 2012). There are many algorithms for learning the parameters of a neural network (Murphy, 2012), but the most influential one is known as the *backpropagation algorithm* (Werbos, 1974).

Neural networks have many success stories like handwriting recognition and stock market prediction. However, due to issues of network complexity and amount of training data required, enthusiasm about neural networks somewhat subsided during the 1990s. With incredible advances in high-performance parallel computing and the emergence of big data (Gudivada et al., 2015a), neural networks reemerged under a new name—deep learning (LeCun et al., 2015). The power of deep learning comes from scalability—the number of hidden layers, and not from new or complex algorithms. Deep-learning algorithms have been making one breakthrough after another in several different areas, including image classification (Krizhevsky et al., 2012), speech recognition (Hinton et al., 2012), and machine translation (Sutskever et al., 2014).

Perhaps the primary reason for the popularity of deep learning is automatic feature extraction. Traditionally, features are carefully hand-crafted by humans. However, it has been shown that for image recognition tasks, deep learning automatically extracts image features in a hierarchical fashion—beginning from the edges in the images to higher-level features progressively (LeCun et al., 2015). This automatic extraction and representation of features has hugely outperformed many standard features like the well-known *sift* features. The latter have been used by the computer vision community for years. As a result, deep learning has initiated a paradigm shift in computer vision.

A major drawback of deep neural networks is that they cannot explain their decisions. From a user perspective, it is an oracle and a blackbox. Designing critical systems with the blind faith that deep learning will pick the "right" features is not a sound engineering design principle.

4.5 Bayesian Networks

Probabilistic approaches to real-world problems are omnipresent today. One of the key challenges in these approaches is the representation of the joint probability of a set of random variables, whose size is exponential in the number of random variables. However, most problems show some type of probabilistic structure in the sense that not every random variable is *conditionally dependent* on every other random variable. In such cases, we can succinctly represent the probability structure. Probabilistic graphical models (Koller and Friedman, 2009) deal with problems where there is a graphical structure among the random variables in terms of the conditional dependencies. BNs are probabilistic graphical models where the graph (or network) among the random variables is a directed acyclic graph (DAG). Each node in the DAG is a random variable and each directed edge from a node A to a node B

represents A's direct influence on B. The directed edges may not necessarily encode causality—in most cases they do not.

In addition to being a data structure for a compact representation of *joint probabilities*, a BN also represents *conditional independence* among its random variables. Interestingly, these two representational aspects of BNs are equivalent. The conditional independence property states: given the nodes that have a directed edge to a node A, the node A is conditionally independent of all the nodes that cannot be reached from A via a directed path. A more technical concept known as *d-separation* deals with the issue of whether two nodes are conditionally independent given a third node, based on the graph structure and irrespective of the actual probability distribution. D-separation is algorithmically well understood, although in the worst case it takes exponential time in the size of the graph (Koller and Friedman, 2009).

The key machine learning problems in the BN setting are learning the parameters (i.e., conditional probabilities) given the graph structure and learning the structure of the graph and the parameters given a probability distribution. For the former, several well-known techniques such as Maximum Likelihood and expectation maximization are widely used. The latter problem is more involved and often times requires searching for a graph structure in the huge space of all possible graphs. Various optimization techniques are used for this task.

Today, BNs boast a wide range of practical applications in diverse fields, such as bioinformatics (Zou and Conzen, 2005), image processing (Mittal, 2007), risk analysis (Weber et al., 2012), and engineering (Heckerman et al., 1995), to name just a few.

4.6 Libraries and Frameworks

Many libraries and frameworks are available for developing cognitive analytics applications. TensorFlow is an open source software library from Google for numerical computation using data flow graphs (Abadi et al., 2016). The library is optimized for execution on clusters and GPU processors. Among many other applications, TensorFlow is a deep-learning platform for computational biologists (Rampasek and Goldenberg, 2016).

Apache Singa is a general purpose, distributed neural platform for training deep-learning models over large datasets. The supported neural models include convolutional neural networks, restricted Boltzmann machines, and recurrent neural networks.

Torch7, Theano, and Caffe are the other deep-learning frameworks which are widely used. The Torch is a GPU-based scientific computing framework with wide support for machine learning algorithms. It provides an easy to use and fast scripting language called LuaJIT, which is implemented using the C language and CUDA. It comes with a large number of community-developed packages for computer vision, signal processing, and machine learning.

Theano is a Python library which is highly suited for large-scale, computationally intensive scientific investigations. Mathematical expressions on large multidimensional arrays can be efficiently evaluated. It tightly integrates with Numpy. Access to the underlying GPU hardware is transparent. Also, it performs efficient symbolic differentiation. Lastly, extensive unit-testing and self-verification functions are integrated into Theano, which enables diagnosing several types of errors in code.

Caffe is particularly suitable for convolutional neural networks and provides options for switching between CPUs and GPUs through configuration parameters. It has been stated that Caffe can process over 60 million images per day with a single Nvidia K40 GPU.

Massive Online Analysis (MOA) is a popular framework for data stream mining. The machine learning algorithms provided by the framework are suitable for tasks such as classification, regression, clustering, outlier detection, concept drift detection, and recommender systems.

MLlib is Apache Spark's machine learning library. Tasks that can be performed using the MLlib include classification, regression, clustering, collaborative filtering, and dimensionality reduction. mlpack is a C++-based machine learning library, which can be used through command line as well as C++ classes.

Pattern is a web mining module for the Python programming language. It features tools for data mining, natural language processing, clustering, network analysis, and visualization. *Scikit-learn* is another Python framework for machine learning, which is implemented using NumPy, SciPy, and matplotlib. Using the included machine learning algorithms, tasks such as clustering, classification, and regression can be accomplished.

Shogun is one of the oldest machine learning libraries, which is written in C++. However, it provides bindings for other languages such as Java, Python, C#, Ruby, R, Lua, Octave, and Matlab. *Veles* is a C++, distributed platform for developing deep-learning applications. Trained models can be exposed through REST API. Using Vales, widely recognized neural topologies such as fully connected, convolutional, and recurrent networks can be trained. Deeplearning4J, neon, and H2O are other libraries for deep learning.

Mahout is an Apache machine learning project. Mahout library is especially suited for execution on cluster computers and GPUs. Also, it tightly integrates with Hadoop Map/Reduce distributed processing framework. Logistic regression classifier, random forest decision trees, K-means clustering, and naive Bayes classifier algorithms are available in Mahout. Apache R project is a sophisticated platform for statistical computing. It features a comprehensive set of machine learning and visualization algorithms.

Amazon Machine Learning is a cloud-hosted service for creating machine learning models without knowing the internal details of machine learning algorithms. This service provides easy access to the data stored in Amazon S3, Redshift, and RDS. Azure ML Studio is a similar service from Microsoft.

5 COGNITIVE ANALYTICS: A COVETED GOAL

The term *cognition* refers to how humans acquire and use knowledge through their senses, learn from interactions and experiences in their environment, and acquire and improve their ability to perform functions such as walking, talking, driving a car, and problem solving. It is hypothesized that cognition is enabled by the higher-level functions of the brain. A *cognitive process* refers to the specific steps the brain uses for accomplishing tasks such as perception, planning, language acquisition, and thinking. Cognitive processes are different from deterministic algorithms. They elegantly cope with data which is ambiguous, uncertain, incomplete, and inconsistent using probabilistic algorithms.

A *cognitive model* is a blueprint for a cognitive process. In other words, a cognitive model *explains* a cognitive process. A set of cognitive processes endows humans' intelligent behavior. Machine cognition is similar to human cognition. Machine cognition targets computers to perform tasks at a performance level which rivals humans. Cognitive analytics is an emerging area and is currently going through a formation stage. It is expected to evolve rapidly and make its way into many commercial software applications.

A *software architecture* defines the overall structure of a software application and specifies its components and their functional characteristics, as well as communication among the components. Some architectures are generic and are used to build a class of software applications, whereas others are specific to just one application. A *cognitive architecture* is a supposition about *fixed* structures of the mind and interactions among them to endow intelligent behavior to humans and machines. The means used to realize cognitive architecture in humans and computers are different. The underlying infrastructure for human cognition is the mind and brain, whereas algorithms and computers encompass the infrastructure for machine cognition. Some cognitive architectures are generic enough to serve as a blueprint for multiple cognitive models. In this section, we describe *Cognalytics*—a proposed architecture for cognitive analytics—and also discuss ways to implement the architecture.

5.1 Cognalytics: A Reference Architecture for Cognitive Analytics

Shown in Fig. 3 is Cognalytics, a high-level reference architecture for implementing cognitive analytics. It is a layered architecture and the circled numbers on the right denote layer numbers. We use the terms system and architecture synonymously, and the context should help to elucidate the intended meaning.

Layer 1 is the *physical data layer* which stores unstructured, semistructured, and structured data. It also stores open source taxonomies and ontologies such as DBpedia and WordNet. Some data in the physical data layer is static or changes rarely, and other data is dynamic and changes with time.

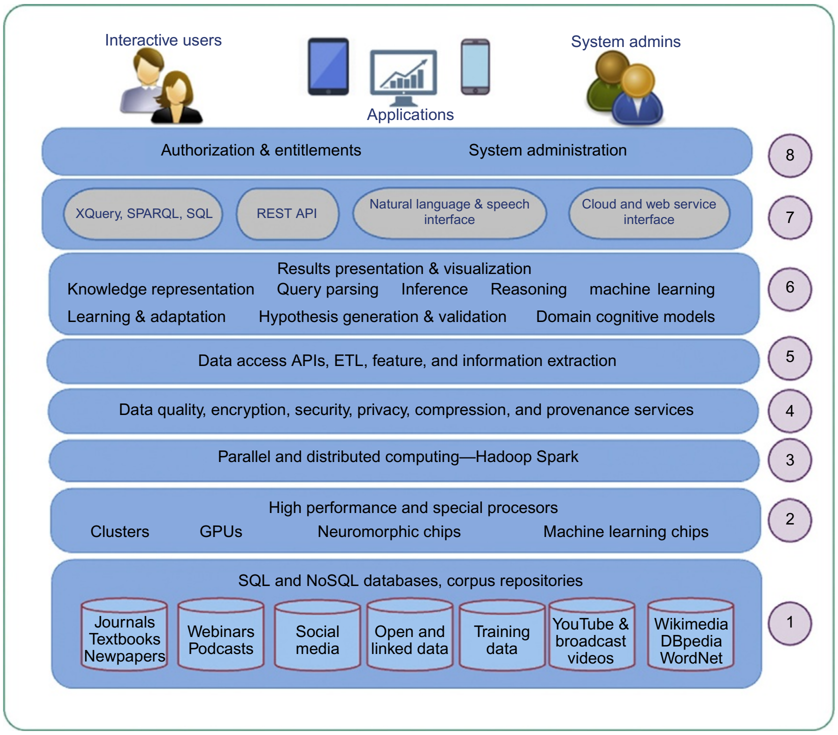
Interactive users
Applications
System admins
Authorization & entitlements
System administration
8
XQuery, SPARQL, SQL
REST API
Natural language & speech interface
Cloud and web service interface
7
Results presentation & visualization
Knowledge representation
Query parsing
Inference
Reasoning
machine learning
Learning & adaptation
Hypothesis generation & validation
Domain cognitive models
6
Data access APIs, ETL, feature, and information extraction
5
Data quality, encryption, security, privacy, compression, and provenance services
4
Parallel and distributed computing—Hadoop Spark
3
High performance and special procesors
Clusters
GPUs
Neuromorphic chips
Machine learning chips
2
SQL and NoSQL databases, corpus repositories
Journals Textbooks Newpapers
Webinars Podcasts
Social media
Open and linked data
Training data
YouTube & broadcast videos
Wikimedia DBpedia WordNet
1

FIG. 3 Cognalytics: a reference architecture for cognitive analytics.

This suggests that static data should be physically stored, whereas the dynamic data should be stored logically. In the latter case, the system knows the requisite information about how to fetch this data as needed from the sources. Even the physically stored static data needs to be kept in sync with their sources. These are optimization issues and are not part of the high-level architecture. Since the data is large and heterogeneous, suitable database management systems (DBMS) should be used. They include both relational and NoSQL databases (Gudivada et al., 2016). Natural language texts are stored as text corpora.

Extremely large data volumes and attendant compute-intensive processing require high-performance computing and distributed processing techniques to meet stringent query latency requirements. Layer 2 addresses this need and is referred to as the *physical hardware layer*. Layer 3 provides a virtual machine and abstractions over layer 2 so that cognitive analytics applications can effectively leverage the computing power of layer 2. Layer 3 is referred to as the *hardware abstractions layer*.

Layer 4 provides data services which are implemented using the abstractions provided by layer 3. Functions featured by the data services layer range a broad spectrum from data cleaning, data quality assessment, compression, and encryption to ensuring privacy and maintaining data provenance. Not every cognitive analytics application may need all these data services. However, the services are generic, relatively low level, and are relevant across a broad range of cognitive analytics applications. Layer 4 is referred to as the *low-level data services layer*.

Layer 5 provides high-level data services. Application developers can specify workflows using the low-level data services of layer 4 and execute them. This layer also provides ETL tools for data integration and creates data warehouses and data marts. Lastly, it provides software tools and libraries for extracting both features and information from semistructured and unstructured data. This layer is referred to as the *high-level data services layer*.

Layer 6 is the core of the Cognalytics reference architecture. It features an assortment of machine learning algorithms, domain cognitive models, and inference and reasoning mechanisms including spatial and temporal reasoning. To facilitate inference and reasoning, several knowledge representation schemes are provided. The Learning & Adaptation subsystem is responsible for storing episodic and other types of knowledge and enables learning, adaptation, and evolution. The Query Parsing subsystem is responsible for parsing queries and identifying subqueries of a query. The Hypothesis Generation & Validation subsystem is responsible for providing several answers to a problem and assigning a degree of confidence to each answer. The Results Presentation & Visualization subsystem provides multimodal interfaces for presenting results. It also features functionality for interactive exploration of results through visualization. This layer is referred to as the *cognitive analytics layer*.

Layer 7 provides access to both interactive users and external systems through declarative query languages and APIs. Queries can be specified using natural language text as well as spoken language. This layer also exposes Cognalytics functions as cloud and web services. These services enable developing cognitive analytics applications without having to deal with the internal complexities of the Cognalytics architecture and its implementation. This layer is referred to as the *API layer*.

Layer 8 provides two major functions. The System Administration subsystem provides functions for creating users and associating them with roles. A role specifies a set of predetermined system functions that the role bearer can execute. The Authorization & Entitlement subsystem is responsible for authenticating users and ensuring that the users execute only functions for which they have authorizations. This layer is referred to as the *administration layer*.

5.2 Implementing Cognalytics

Implementing Cognalytics architecture requires substantial effort. Numerous open source libraries and tools are available to ease the effort. Furthermore, one can select the best library or framework from among the choices for each subsystem. We describe implementation layer-wise starting with the bottom layer. The tools and frameworks we indicate in this section are open source unless specified otherwise.

5.2.1 *Physical Data Layer*

PostgreSQL is an open source RDBMS which provides high availability, horizontal scalability, and performance. Replication and auto-sharding features are also available. It is an ideal choice for storing structured data. As of this writing, there are over 300 DBMS available for data management and most of these systems are open source (Solid IT, 2016). A number of NoSQL databases are available for storing text corpora and other unstructured data. Virtuoso, Sedna, BaseX, and eXist-db are native XML databases. Database systems for time series data include InfluxDB, RRDtool, Graphite, and OpenTSDB. Jena, Virtuoso, and Sesame are database systems for RDF data. For graph data management, Neo4j, OrientDB, Titan, Virtuoso, and ArangoDB are popular choices. The reader should consult (Solid IT, 2016) to explore unprecedented choices for data management.

5.2.2 *Physical Hardware Layer*

Though one can develop the computing infrastructure for Cognalytics in-house, it is often economical to use a cloud platform such as Amazon Web Services. On the other hand, developing an in-house infrastructure has its advantages. Special compute processors such as neuromorphic chips and neural network accelerators are available for developing the infrastructure. For example, True North (Merolla et al., 2014) is a brain-inspired neuromorphic chip. It is a self-contained chip with 5.4 billion transistors. True North features 1 million programmable neurons, 256 million programmable synapses on the chip, 4096 parallel and distributed cores which are interconnected via an on-chip mesh network, and 400 million bits of local on-chip memory. How True North has been used to implement convolutional networks for classification problems is described in Esser et al. (2016).

A class of microprocessors, called AI accelerators, are emerging to accelerate machine learning algorithms. For example, tensor processing units are application-specific processors developed for Google's TensorFlow framework (TensorFlow, 2016). As of this writing, Nvidia released Tesla P100 GPU chip. The chip specifically targets machine learning algorithms that employ deep learning. Tesla P100 features 150 billion transistors on a single chip. DGX-1, Nvidia's newest supercomputer, is powered by 8 Tesla P100 GPUs and ships with deep-learning software preinstalled. Zeroth is a cognitive computing platform developed by Qualcomm. The platform runs on a neural processing unit AI accelerator chip and deep-learning algorithms are available through an API. The latter is specifically designed for mobile devices to process image and speech data. Other neurocomputing engines include Chen et al. (2015), Du et al. (2015), Kim et al. (2015), and Liu et al. (2013).

5.2.3 Hardware Abstractions Layer

This layer provides libraries and frameworks to ease the application development process using specialized processors such as neuromorphic chips. The libraries and frameworks enable application developers to write code without concerns for the underlying special hardware. The application code is automatically transformed to enable efficient execution. Currently, Hadoop and Spark are popular choices for realizing this layer. Typically, neuromorphic and other chip manufactures provide APIs using which applications development can be accelerated. As neuromorphic processors' use becomes more widespread, we expect more advanced libraries and frameworks.

5.2.4 Low-level Data Services Layer

Data ingestion into a cognitive analytics system is a major task given that the volume of data is generally petabyte scale and, in some cases, even exabytes. Sqoop and Flume are two tools in the Hadoop ecosystem for extracting data from different sources and loading it into the Hadoop Distributed File System. Sqoop is used for extracting and loading structured data, whereas Flume does the same for unstructured data.

Many cognitive analytics applications acquire data from diverse data vendors to complement internally generated data. Algorithms and workflows for data cleaning are required for detecting and eliminating duplicates, resolving conflicting and inconsistent data, inferring missing data, detecting integrity constraint violations, and detecting and resolving outliers. Ganti and Sarma (2013) discuss a few popular approaches for developing data cleaning solutions. Other works in this direction include Osborne (2012) and McCallum (2012).

Protecting privacy rights is a tremendous challenge. Differential privacy restricts users' access to data based on their job roles. Data encryption simplifies data security and privacy protection. Especially in the medical and health care domains, the notion of *personally identifiable information* is central. Some techniques such as *data perturbation* enable data analytics without compromising privacy requirements. Data perturbation is a more effective approach for privacy preservation of electronic health records than deidentification and reidentification procedures.

Provenance involves maintaining a history of processing that has been applied to a data item. The history is maintained in the form of metadata graphs, which grow very rapidly. Analyzing these graphs is computationally expensive (Cheah, 2014). Tracking provenance may not be a concern in some cognitive analytics applications. The Open Provenance Model is a collection of specifications for implementing provenance. Pentaho Kettle, eBioFlow, PLIER, and SPADE are tools for implementing provenance.

Given the data volumes, data compression is an important consideration. Generally, text compression requires lossless algorithms—original data and data recovered from the compressed data are identical. Image and video data may tolerate some data loss when decompressed. RainStor/Teradata, a database specifically developed for big data, seems to provide a compression ratio of 40:1, and in some cases, the ratio is as high as 100:1.

5.2.5 High-Level Data Services Layer

Tools are required for integrating data from multiple sources. This data fusion requires normalizing data so that it conforms to a canonical form, identifying related data about an entity from different sources, specifying transformation rules, and resolving any conflicts. ETL are a set of tools which originated from the data warehousing area are used for this purpose. Scriptella, KETL, Pentaho Data Integrator—Kettle, Talend Open Source Data Integrator, Jaspersoft ETL, GeoKettle, Jedox, Apatar, CloverETL, and HPCC Systems are excellent ETL tools.

Pivotal Greenplum (originally Greenplum Database) is a massively parallel data warehouse. Greenplum branched off from PostgreSQL and added several data warehousing features. Pivotal Greenplum is uniquely suitable for big data analytics. Apache MADlib is a library for scalable in-database analytics (Hellerstein et al., 2012). MADlib provides parallel implementations of machine learning and mathematical and statistical functions. MADlib currently supports Pivotal Greenplum, PostgreSQL and Apache HAWQ (Hadoop Native SQL platform) databases, and data warehouses. Many NoSQL databases compute analytics efficiently in batch mode using MapReduce frameworks (Gudivada et al., 2016).

Several tools are available for extracting features and information from unstructured data, primarily natural language text. Apache UIMA project provides frameworks, tools, and annotators for facilitating the analysis of unstructured content such as text, audio, and video. Tools from the Stanford NLP group for solving major computational linguistics problems include statistical NLP, deep-learning NLP, and rule-based NLP. Other tools for solving natural language problems include openNLP and GATE. Apache Lucene Core is a full-featured text search engine Java library. GPText from Greenplum is a statistical text analysis framework optimized for execution on parallel computing platforms. GPText is also available as a cloud service (Li et al., 2013).

SyntaxNet is an open source neural network framework for developing natural language understanding systems. Parsey McParseface is a pretrained SyntaxNet model for parsing the standard English language. TensorFlow is another software library for machine intelligence. NuPIC is a platform for cognitive computing, which is based on a theory of neocortex called Hierarchical Temporal Memory (HTM).

Weka 3 is a Java software library for data mining. The R project provides a platform for statistical computing and visualization. OpenCV and ImageJ are libraries for computer vision tasks. Praat is a tool for speech manipulation, analysis, and synthesis. openSMILE is another tool for extracting audio features in real time.

5.2.6 Cognitive Analytics Layer

This layer brings all components and subsystems together by serving as an integrator and coordinator. Some of the libraries and tools we indicated in Section 5.2.5 are also useful for implementing this layer. This is because the distinction between low- and high-level features is subjective and fluid. And so is the distinction between data and information, as well as information and knowledge.

There are several tools for implementing this layer. Their functions are often complementary and multiple tools are needed. FRED is a machine reader for the Semantic Web (Presutti et al., 2012). It parses natural language text in 48 languages and transforms it to linked data. It is available as both a REST service and a Python library suite. Apache Stanbol is a software stack and reusable set of components for semantic content management. Federated knOwledge eXtraction Framework (FOX) is a tool for RDF extraction from text using ensemble learning (Speck and Ngonga Ngomo, 2014). Named Entity Recognition and Disambiguation (NERD) is another framework which unifies 10 popular named entity extractors and compares their performance (Rizzo and Troncy, 2012). Accurate Online Disambiguation of Named Entities in Text and Tables (AIDA) is another tool for extracting named entities from natural language texts (Yosef, 2016). AlchemyAPI provides 12 semantic text analysis APIs for natural language understanding (Feyisetan et al., 2014). Machine learning libraries for data mining include PyML, Apache Mahout, MLib, dlibml, WEKA, and scikit-learn.

There are several options for implementing the Results Presentation & Visualization subsystem. Results presentation is tied to web application development frameworks used for implementing Cognalytics. User interface development frameworks such as Bootstrap, Foundation, GroundworkCSS, Gumby, HTML KickStart, IVORY, and Kube provide rich functionality for results presentation and navigating the application. D3, chart, dygraphs, FusionCharts, and Highcharts are libraries for visualization which run in a web browser.

5.2.7 API Layer

Cognalytics provides several APIs for interaction with the outside world. SQL, SPARQL, and XQuery are standard languages for querying RDBMS, RDF, and native XML databases declaratively. Representational State Transfer (REST) is a minimal overhead Hypertext Transfer Protocol (HTTP) API for interacting with Cognalytics system. REST uses four HTTP methods GET (reading data), POST (writing data), PUT (updating data), and DELETE (removing data). Natural language and query interfaces provide a natural means for interacting with the system. The first two classes of interfaces primarily serve the needs of interactive users who pose structured queries, whereas the last class enables a more powerful and flexible way to submit queries.

5.2.8 Administration Layer

System administration functions include user management, system monitoring, backup and recovery, and access control. System monitoring, backup, and recovery functions are typically integrated into an organization-wide application. User management functions include user creation and assigning roles to users. Single sign-on (SSO) is a user authentication service that permits the same login ID and password to access multiple systems across an organization. Often software libraries combine authentication and authorization functions into one component.

Shibboleth is an open source software which provides federated identity solution. It enables users to connect to applications within and outside an organization using SSO. Apache Shiro is a Java security framework for integrating authentication, authorization, cryptography, and session management functions into applications. Other solutions include OpenDJ, OpenIDM, OpenAM, and DACS.

6 COGNITIVE ANALYTICS APPLICATIONS

Though data warehouses-driven analytics has been in existence for over 28 years (Devlin and Murphy, 1988), only recently has there been tremendous thrust on incorporating unstructured data into data analytics. The power of cognitive analytics stems from the complementary and synergistic value heterogeneous data sources bring. Cognitive analytics applications range from improving student engagement and developing intervention measures, developing more effective Intelligent Tutoring Systems (ITS) to developing cognitive assistants and personalized learning environments.

6.1 Learning Analytics

EDM and LA are two areas in the education and learning domain that draw upon data analytics. EDM can be viewed as descriptive analytics. The current EDM systems are tied to course management systems (CMS) such as Blackboard and Moodle, which provide structured data for analytics. Such data includes the number of CMS logins, time spent on each learning activity, and test scores. Based on this data, students are classified into various groups and appropriate intervention measures are designed for each group. There is no human involvement in this process.

LA takes EDM a step further. LA combines EDM with human judgment (Siemens, 2012). It is best viewed as prescriptive analytics. It uses machine learning algorithm techniques to reveal hidden patterns and generate actionable intelligence. The latter is used to design personalized intervention measures. In addition to structured data, LA includes semistructured data such as emails and discussion board postings into analytics. Recent efforts in LA aim to propel both EDM and LA into the realm of predictive analytics and beyond into cognitive analytics.

6.2 Personalized Learning

Personalized learning can be defined from multiple perspectives. One approach is to allow learners to proceed at their own pace. The order in which topics are learned by one user may be different from the order of topics for another learner. In other words, learners are not bound to a lock-step synchronization scheme. They are free to explore topics on a subject in any order, only constrained by the prerequisite dependencies. Another aspect is automated generation of assessments, providing contextualized and incremental scaffolding, and supplying immediate feedback on assessments. Descriptive analytics will help to suggest next topics to pursue to the learner. A personalized learning system, called ISPeL, which is based on the above principles is described in Gudivada (2016). How ISPeL can be extended to incorporate cognitive analytics is also described.

6.3 Cognitive Businesses

This is perhaps the single domain that has been deeply impacted by cognitive analytics already. Cognitive businesses are those that use cognitive analytics for both operational management and strategic decision making. The primary thrust is on extracting information from natural language texts and combining it with structured data. Cognitive analytics uses are vast and varied. It has been used to improve workflow processes, detecting fraud before it happens, ensuring regulatory compliance, repurposing content, and knowledge management. Technology companies such as IBM, Nvidia, Google, Microsoft, LinkedIn, Facebook, and Netflix have already incorporated cognitive analytics into their software products.

Multiple Sclerosis Association of America uses cognitive analytics and natural language understanding to return evidence-based answers to clinicians' complex questions. To find an answer, their system parses a corpus of 1500 question-and-answer pairs and also incorporates content from medical resources. Baylor College of Medicine used IBM Watson to develop Baylor Knowledge Integration Toolkit (KnIT). The latter's goal is to help researchers by discovering patterns in the research literature. KnIT helped researchers identify proteins that modify p53, a protein related to many cancers. The system analyzed 70,000 scientific articles on p53 to predict other proteins that turn on/off of p53's activity. This finding was accomplished in a matter of weeks, which would have taken researchers years without IBM Watson.

6.4 BCI and Assistive Technologies

The human brain is perhaps the most complex system in terms of its structure and function. Functional magnetic resonance imaging and electroencephalogram are two functional brain imaging techniques that help to establish an association between brain and behavior. BCI is a new technology that provides a direct communication pathway between a wired brain and an external device such as a robot or wheel chair. Cognitive analytics offers an exciting opportunity to develop new assistive technologies using BCI. The study reported in Harnarinesingh and Syan (2013) discusses how a three-axis industrial robot was used to generate writing. The next logical step is to investigate connecting the brain and the robot using BCI. This is just one example, and cognitive analytics and BCI have the potential to help develop many assistive technologies to help physically impaired people.

7 CURRENT TRENDS AND RESEARCH ISSUES

Cognitive analytics will be increasingly driven by special computing processors that mimic the neural computations of the brain. Advances in neuroscience and cognitive science are critical for propelling neuromorphic computing further. It is ironic that computing discipline itself will be enabling new discoveries in these sciences. Rapid advances in big data will exacerbate the need to move more and more processing into hardware to meet performance at scale requirements.

There is a mismatch between the current programming languages and software development environments with respect to neuromorphic architectures powered by neurosynaptic cores. IBM has already begun designing simulators and programming environments including a new programming language and associated libraries for the True North processor. Nvidia, Google, and Facebook have similar projects in the pipeline.

Cognitive computing and cognitive analytics will play a transformational role in the Internet of Things (IoT) domain. *Embedded analytics* in general and cognitive IoT in particular will endow wireless sensors and cameras to perform intelligent processing at the source. This has multiple benefits including improved data quality, adaptive sampling to reduce the volume of streaming sensor data, and increased opportunities for a community of sensors to work as collaborative agents. Another use for embedded analytics is to integrate discovered actionable insights into products that would benefit from such insights. More and more future applications will have embedded analytics, which will enable them to deliver additional value. For example, wearable medical devices will not only be able to generate timely alerts but also provide contextualized information about how to react to the alerts.

Current research in both feature and information extraction from unstructured data is primarily focused on natural language text. Recently, revived interest in neural computing and in particular convolutional networks has begun to yield new algorithms and approaches to image classification and object recognition problems. Similar emphasis is needed for speech and video data.

It is hypothesized that the human brain uses statistical learning. Creating neural models to simulate the brain is not easy given the current computing processors. The emergence of neuromorphic chips offers hope and excitement.

Currently, neuromorphic chips can simulate neurons in the order of millions and synaptic connections in the order of billions. To propel cognitive analytics to the next stage, we need neuromorphic processors that can simulate neurons in the order of billions and synaptic connections in the order of trillions.

It is widely believed that our brain uses statistical learning. Creating neural models to simulate a brain is not easy. Current artificial neural models: nodes in the order of millions and connections in the order of billions. What we want is nodes in the order of billions and connections in the order of trillions.

Cognitive analytics will play an increasingly critical role in smart cities. Insights will help in planning evacuation routes, prioritize resource allocations in implementing disaster relief, optimize energy usage, promote public safety, and prevent maintenance on city infrastructure. Personalized learning will be another beneficiary of cognitive analytics. However, significant research is needed in these areas to reap the benefits.

8 CONCLUSIONS

In this chapter, we defined analytics and traced its evolution. Cognitive analytics is pursued from two complementary perspectives: computer science, and cognitive and neurosciences. This chapter focused primarily on the computer science perspective. We introduced learning types and discussed several classes of machine learning algorithms. We proposed a reference architecture for cognitive analytics and indicated ways to implement it. We described a few cognitive analytics applications and indicated current trends and future research in cognitive analytics. Cognitive computing and analytics have immense potential to contribute to a new generation of applications for which learning is intrinsic and communication is through spoken and written natural language. It is a proven technology which is searching for applications.

Cognitive computing and analytics are more than just the AI. For example, AI is 1 of the 28 APIs provided by the IBM Watson. Cognitive computing in general and cognitive analytics in particular exacerbate data security, privacy, and provenance issues. There are also practical concerns with cognitive analytics. Would this technology lead to significant unemployment? Would it only enable people to do their jobs better or totally replace them? On a philosophical level, how far can we advance cognitive technologies? Will they advance to a level at which they surpass human intelligence? If so, what are its implications to individuals as well as the society at large?

The computing industry has never been before invested this level of effort and resources into machine learning research. The availability of inexpensive, cloud-based computing power, and the ubiquity of big data are the catalysts for the transformational advances we are witnessing in machine learning and cognitive computing. The synergistic confluence of computing, neuroscience, and cognitive science is poised for groundbreaking discoveries and compelling cognitive applications in the years ahead.

REFERENCES

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., et al., 2016. TensorFlow: large-scale machine learning on heterogeneous distributed systems. ArXiv preprint arXiv:1603.04467.

Aizerman, A., Braverman, E.M., Rozoner, L., 1964. Theoretical foundations of the potential function method in pattern recognition learning. Autom. Remote Control. 25, 821–837.

Ayalew, L., Yamagishi, H., 2005. The application of GIS-based logistic regression for landslide susceptibility mapping in the Kakuda-Yahiko mountains, central Japan. Geomorphology 65 (1), 15–31.

Battiti, R., Brunato, M., Mascia, F., 2008. Reactive Search and Intelligent Optimization. Operations Research/Computer Science Interfaces, vol. 45. Springer Science & Business Media, Berlin, Germany.

Bell, J., 2014. Machine Learning: Hands-on for Developers and Technical Professionals. John Wiley & Sons, Hoboken, NJ.

Bishop, C.M., 2006. Pattern recognition and machine learning. Information Science and Statistics. Springer, New York, NY.

Boser, B.E., Guyon, I.M., Vapnik, V.N., 1992. A training algorithm for optimal margin classifiers. In: Proceedings of the Fifth Annual Workshop on Computational Learning Theory. ACM, New York, NY, pp. 144–152.

Breiman, L., 2001. Random forests. Mach. Learn. 45 (1), 5–32.

Breiman, L., Friedman, J., Stone, C.J., Olshen, R.A., 1984. Classification and Regression Trees. CRC Press, Boca Raton, FL.

Cai, C.Z., Han, L.Y., Ji, Z.L., Chen, X., Chen, Y.Z., 2003. SVM-Prot: web-based support vector machine software for functional classification of a protein from its primary sequence. Nucleic Acids Res. 31 (13), 3692–3697.

Chakraborty, A., Harrison, B., Yang, P., Roberts, D., St. Amant, R., 2014. Exploring key-level analytics for computational modeling of typing behavior. In: Proceedings of the 2014 Symposium and Bootcamp on the Science of Security, Raleigh, North Carolina, USA, HotSoS'14. ACM, New York, NY, pp. 34:1–34:2. http://doi.acm.org/10.1145/2600176.2600210.

Chang, M.-W., Yih, W.-T., Meek, C., 2008. Partitioned logistic regression for spam filtering. In: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, New York, NY, pp. 97–105.

Cheah, Y.-W., 2014. Quality, retrieval and analysis of provenance in large-scale data. Ph.D. thesis, Indiana University, Indianapolis, IN. Plale, Beth.

Chen, P.-Y., Kadetotad, D., Xu, Z., Mohanty, A., Lin, B., Ye, J., Vrudhula, S., Seo, J.-S., Cao, Y., Yu, S., 2015. Technology-design co-optimization of resistive cross-point array for accelerating learning algorithms on chip. In: Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition, DATE'15. EDA Consortium, San Jose, CA, pp. 854–859.

Devlin, B.A., Murphy, P.T., 1988. An architecture for a business and information system. IBM Syst. J. 27 (1), 60–80.

Du, Z., Ben-Dayan Rubin, D.D., Chen, Y., He, L., Chen, T., Zhang, L., Wu, C., Temam, O., 2015. Neuromorphic accelerators: a comparison between neuroscience and machine-learning approaches. In: Proceedings of the 48th International Symposium on Microarchitecture, MICRO-48. ACM, New York, NY, pp. 494–507.

Esser, S.K., Merolla, P.A., Arthur, J.V., Cassidy, A.S., Appuswamy, R., Andreopoulos, A., Berg, D.J., McKinstry, J.L., Melano, T., Barch, D.R., Nolfo, C.D., Datta, P., Amir, A., Taba, B., Flickner, M.D., Modha, D.S., 2016. Convolutional networks for fast, energy-efficient neuromorphic computing. Comput. Res. Reposit. abs/1603.08270. http://arxiv.org/abs/1603.08270.

Feng, X., Kumar, A., Recht, B., Ré, C., 2012. Towards a unified architecture for in-RDBMS analytics. In: Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data, SIGMOD'12. ACM, New York, NY, pp. 325–336. http://doi.acm.org/10.1145/2213836.2213874.

Feyisetan, O., Simperl, E., Tinati, R., Luczak-Roesch, M., Shadbolt, N., 2014. Quick-and-clean extraction of linked data entities from microblogs. In: Proceedings of the 10th International Conference on Semantic Systems, SEM '14. ACM, New York, NY, pp. 5–12.

Freund, Y., Schapire, R.E., 1995. A decision-theoretic generalization of on-line learning and an application to boosting. In: European Conference on Computational Learning Theory. Springer, Berlin, Germany, pp. 23–37.

Freund, Y., Schapire, R., Abe, N., 1999. A short introduction to boosting. J. Jpn. Soc. Artif. Intell. 14 (771–780), 1612.

Ganti, V., Sarma, A.D., 2013. Data Cleaning: A Practical Perspective. Synthesis Lectures on Data Management, Morgan & Claypool Publishers, Williston, VT.

Geman, S., Bienenstock, E., Doursat, R., 1992. Neural networks and the bias/variance dilemma. Neural Comput. 4 (1), 1–58.

Goodfellow, I.J., Bulatov, Y., Ibarz, J., Arnoud, S., Shet, V., 2014. Multi-digit number recognition from street view imagery using deep convolutional neural networks. https://arxiv.org/pdf/1312.6082.pdf. arXiv.org.

Gudivada, V.N., 2016. Cognitive analytics driven personalized learning. Educ. Technol. Mag. Special Issue on Big Data in E-Learning, Educational Technology Publications, Englewood Cliffs, NJ, in press.

Gudivada, V., Baeza-Yates, R., Raghavan, V., 2015a. Big data: promises and problems. IEEE Comput. 48 (3), 20–23.

Gudivada, V., Rao, D., Raghavan, V., 2015b. Big data driven natural language processing research and applications. In: Govindaraju, V., Raghavan, V., Rao, C.R. (Eds.), Big Data Analytics, Handbook of Statistics. vol. 33. Elsevier, Amsterdam, The Netherlands, pp. 203–238.

Gudivada, V., Rao, D., Raghavan, V., 2016. Renaissance in database management: navigating the landscape of candidate systems. IEEE Comput. 49 (4), 31–42.

Harnarinesingh, R.E.S., Syan, C.S., 2013. Investigating the feasibility of a robot-based writing agent. In: Proceedings of the Second International Conference on Innovative Computing and Cloud Computing, ICCC'13. ACM, New York, NY, p. 60:60:65. 60.

Heckerman, D., Mamdani, A., Wellman, M.P., 1995. Real-world applications of Bayesian networks. Commun. ACM 38 (3), 24–26.

Hellerstein, J.M., Ré, C., Schoppmann, F., Wang, D.Z., Fratkin, E., Gorajek, A., Ng, K.S., Welton, C., Feng, X., Li, K., Kumar, A., 2012. The MADlib analytics library: or MAD skills, the SQL. Proc. VLDB Endow. 5 (12), 1700–1711. http://dx.doi.org/10.14778/2367502. 2367510.

Hinton, G., Deng, L., Yu, D., Dahl, G.E., Mohamed, A.-R., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T.N., et al., 2012. Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups. IEEE Signal Process. Mag. 29 (6), 82–97.

Ho, T.K., 1995. Random decision forests. Proceedings of the Third International Conference on Document Analysis and Recognition, vol. 1, pp. 278–282.

Honorio, J., Ortiz, L., 2015. Learning the structure and parameters of large-population graphical games from behavioral data. J. Mach. Learn. Res. 16, 1157–1210.

Irfan, M.T., Ortiz, L.E., 2011. A game-theoretic approach to influence in networks. In: Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, AAAI'11. AAAI Press, San Francisco, CA, pp. 688–694.

Irfan, M.T., Ortiz, L.E., 2014. On influence, stable behavior, and the most influential individuals in networks: a game-theoretic approach. Artif. Intell. 215, 79–119.

Jurafsky, D., Martin, J.H., 2009. Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, second ed., Pearson Prentice Hall, Upper Saddle River, NJ.

Kim, Y., Zhang, Y., Li, P., 2015. A reconfigurable digital neuromorphic processor with memristive synaptic crossbar for cognitive computing. ACM J. Emerg. Technol. Comput. Syst. 11 (4), 38:1–38:25. http://dx.doi.org/10.1145/2700234.

Koller, D., Friedman, N., 2009. Probabilistic Graphical Models: Principles and Techniques. MIT Press, Cambridge, MA.

Krizhevsky, A., Sutskever, I., Hinton, G.E., 2012. ImageNet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems. Neural Information Processing Society (NIPS), Inc., La Jolla, CA, pp. 1097–1105.

Laitinen, E.K., Laitinen, T., 2001. Bankruptcy prediction: application of the Taylor's expansion in logistic regression. Int. Rev. Financ. Anal. 9 (4), 327–349.

Lake, B.M., Salakhutdinov, R., Tenenbaum, J.B., 2015. Human-level concept learning through probabilistic program induction. Science 350 (6266), 1332–1338. http://dx.doi.org/10.1126/science.aab3050. http://science.sciencemag.org/content/350/6266/1332.

LeCun, Y., Bengio, Y., Hinton, G., 2015. Deep learning. Nature 521 (7553), 436–444. http://dx.doi.org/10.1038/nature14539.

Lee, S., 2005. Application of logistic regression model and its validation for landslide susceptibility mapping using GIS and remote sensing data. Int. J. Remote Sens. 26 (7), 1477–1491.

Li, K., Grant, C., Wang, D.Z., Khatri, S., Chitouras, G., 2013. GPText: greenplum parallel statistical text analysis framework. In: Proceedings of the Second Workshop on Data Analytics in the Cloud, DanaC'13. ACM, New York, NY, pp. 31–35. http://doi.acm.org/10.1145/2486767.2486774.

Liu, B., Hu, M., Li, H., Chen, Y., Xue, C.J., 2013. Bio-inspired ultra lower-power neuromorphic computing engine for embedded systems. In: Proceedings of the Ninth IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis, CODES + ISSS'13. IEEE Press, Piscataway, NJ, pp. 23:1–23:1.

Maher, J.J., Sen, T.K., 1997. Predicting bond ratings using neural networks: a comparison with logistic regression. Intell. Syst. Acc. Finan. Manag. 6 (1), 59–72.

McCallum, Q.E., 2012. Bad Data Handbook: Cleaning up the Data so You Can Get Back to Work. O'Reilly Media, Sebastopol, CA.

McCulloch, W.S., Pitts, W., 1943. A logical calculus of the ideas immanent in nervous activity. Bull. Math. Biophys. 5 (4), 115–133.

Merolla, P.A., Arthur, J.V., Alvarez-Icaza, R., Cassidy, A.S., Sawada, J., Akopyan, F., Jackson, B.L., Imam, N., Guo, C., Nakamura, Y., Brezzo, B., Vo, I., Esser, S.K., Appuswamy, R., Taba, B., Amir, A., Flickner, M.D., Risk, W.P., Manohar, R., Modha, D.S., 2014. A million spiking-neuron integrated circuit with a scalable communication network and interface. Science 345 (6197), 668–673. http://dx.doi.org/10.1126/science.1254642.

Mittal, A., 2007. Bayesian Network Technologies: Applications and Graphical Models: Applications and Graphical Models. IGI Global, Hershey, PA.

Murphy, K.P., 2012. Machine Learning: A Probabilistic Perspective. MIT Press, Cambridge, MA.

Nadkarni, P.M., Ohno-Machado, L., Chapman, W.W., 2011. Natural language processing: an introduction. J. Am. Med. Inform. Assoc. 18 (5), 544–551.

Osborne, J.W., 2012. Best Practices in Data Cleaning: A Complete Guide to Everything You Need to Do Before and After Collecting Your Data. SAGE Publications, Thousand Oaks, CA.

Polikar, R., 2009. Ensemble learning. Scholarpedia 4 (1), 2776. http://www.scholarpedia.org/article/Ensemble_learning.

Presutti, V., Draicchio, F., Gangemi, A., 2012. Knowledge extraction based on discourse representation theory and linguistic frames. In: Proceedings of the 18th International Conference on Knowledge Engineering and Knowledge Management, Galway City, Ireland, EKAW'12. Scite Press, Setúbal, Portugal, pp. 114–129.

Quinlan, J.R., 1986. Induction of decision trees. Mach. learn. 1 (1), 81–106.

Quinlan, J.R., 2014. C4.5: Programs for Machine Learning. Elsevier, Amsterdam, The Netherlands.

Quinlan, J.R., 2016. C5.0: Programs for Machine Learning. RuleQuest Research, Empire Bay. https://www.rulequest.com.

Rampasek, L., Goldenberg, A., 2016. TensorFlow: biology's gateway to deep learning? Cell Syst. 2 (1), 12–14.

Rizzo, G., Troncy, R., 2012. NERD: a framework for unifying named entity recognition and disambiguation extraction tools. In: Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics, EACL'12. Association for Computational Linguistics, Stroudsburg, PA, pp. 73–76. http://dl.acm.org/citation.cfm?id=2380921.2380936.

Rosenblatt, F., 1958. The perceptron: a probabilistic model for information storage and organization in the brain. Psychol. Rev. 65 (6), 386.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L., 2015. ImageNet large scale visual recognition challenge. Int. J. Comput. Vis. 115 (3), 211–252. http://dx.doi.org/10.1007/ s11263-015-0816-y.

Schölkopf, B., Burges, C.J., 1999. Advances in Kernel Methods: Support Vector Learning. MIT Press, Cambridge, MA.

Settles, B., 2009. Active learning literature survey. University of Wisconsin–Madison. Computer Sciences Technical Report 1648.

Shalev-Shwartz, S., Ben-David, S., 2014. Understanding Machine Learning: From Theory to Algorithms. Cambridge University Press, Cambridge.

Siemens, G., 2012. Learning analytics: envisioning a research discipline and a domain of practice. In: Proceedings of the 2nd International Conference on Learning Analytics and Knowledge, Vancouver, British Columbia, Canada. ACM, New York, NY, pp. 4–8.

Solid, I.T., 2016. Knowledge base of relational and NoSQL database management systems. http://db-engines.com/en/ranking. Retrieved: July 2016.

Speck, R., Ngonga Ngomo, A.-C., 2014. Ensemble learning for named entity recognition. In: Mika, P., Tudorache, T., Bernstein, A., Welty, C., Knoblock, C., Vrandečić, D., Groth, P., Noy, N., Janowicz, K., Goble, C. (Eds.), The Semantic Web—ISWC 2014. Lecture Notes in Computer Science, vol. 8796. Springer, Berlin, Germany, pp. 519–534. http://svn.aksw.org/papers/2014/ISWC_EL4NER/public.pdf.

Sutskever, I., Vinyals, O., Le, Q.V., 2014. Sequence to sequence learning with neural networks. In: Advances in Neural Information Processing Systems. Neural Information Processing Systems Foundation, Inc., La Jolla, CA, pp. 3104–3112.

Taigman, Y., Yang, M., Ranzato, M., Wolf, L., 2014. DeepFace: closing the gap to human-level performance in face verification. In: Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR'14. IEEE Computer Society, Washington, DC, pp. 1701–1708. http://dx.doi.org/10.1109/CVPR.2014.220.

TensorFlow, 2016. An open source software library for numerical computation using data flow graphs. https://www.tensorflow.org/.

Tong, S., Koller, D., 2001. Support vector machine active learning with applications to text classification. J. Mach. Learn. Res. 2, 45–66.

Weber, P., Medina-Oliva, G., Simon, C., Iung, B., 2012. Overview on Bayesian networks applications for dependability, risk analysis and maintenance areas. Eng. Appl. Artif. Intell. 25 (4), 671–682.

Werbos, P., 1974. Beyond regression: new tools for prediction and analysis in the behavioral sciences. Ph.D. thesis, Harvard University, Cambridge, MA.

Yosef, M.A., 2016. U-AIDA: a customizable system for named entity recognition, classification, and disambiguation. Ph.D. thesis, Saarland University.

Zou, M., Conzen, S.D., 2005. A new dynamic Bayesian network (DBN) approach for identifying gene regulatory networks from time course microarray data. Bioinformatics 21 (1), 71–79.