Big Data Preprocessing: An Application on Online Social Networks

Androniki Sapountzi and Kostas E. Psannis

1 Introduction

Social networking data is a significant source of big data because they are voluminous, even from a single social network service (SNS), are mostly unstructured, and are evolving at an extremely fast pace. In big data theory, these dimensions are respectively called data volume, data variety, and data velocity. To mine insightful patterns from these data requires advanced data management and analysis methods, termed as Big Data Analytics. This is a complex process that demands expertise in gathering, processing and cleaning data, selecting proper analysis methods, understanding trade-offs of the chosen algorithms, and interpreting and exploiting the results.

An additional dimension of big data is that of data veracity [1] which came as a response to the weakness of messy data. Messy data are characterized by incompleteness, inconsistencies, and timeliness. In addition, data are unstructured, highly dimensional, ambiguous, or imbalanced, and they have to be preprocessed so as to satisfy the requirements of the algorithms used for analysis. The success of many machine learning algorithms depends highly on the preparation phase. Social networking data analysis faces all the abovementioned issues. Figure 1 depicts how the low data quality can affect all the stages of analytics process. Each process depicted in a rectangle is informed and affected by the previous process.

A. Sapountzi (✉)
Department of Behavioral and Movement Sciences, Vrije Universiteit, Amsterdam, The Netherlands
e-mail: a.sapountzi@vu.nl

K. E. Psannis
Department of Applied Informatics, University of Macedonia, Thessaloniki, Greece
e-mail: kpsannis@uom.edu.gr

© Springer Nature Switzerland AG 2020
H. R. Arabnia et al. (eds.), *Principles of Data Science*, Transactions on Computational Science and Computational Intelligence,
https://doi.org/10.1007/978-3-030-43981-1_4

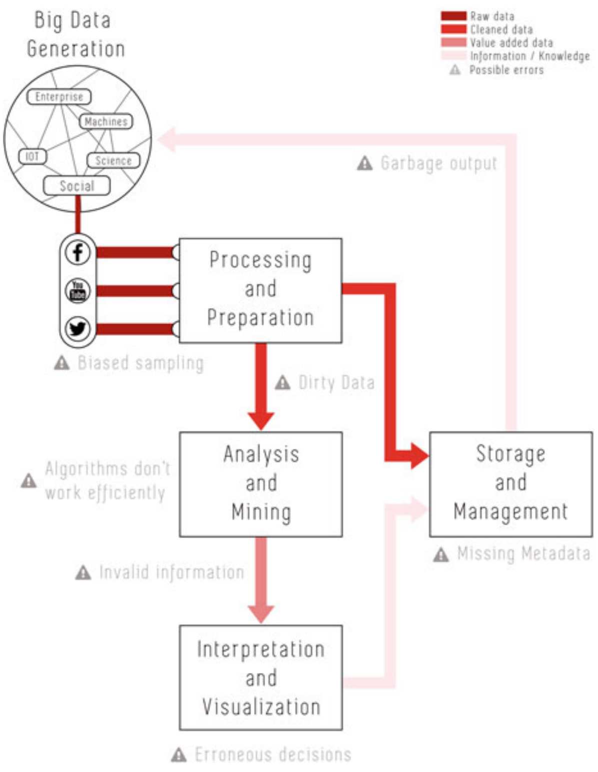


Fig. 1 The impact of low-quality data on each stage of big data analytics in online social networks

The purpose of data processing is both to revert the data to a format capable for the analysis process and to ensure the high quality of data. As the volume, variety, complexity, and velocity of data grow, data preprocessing becomes even more challenging. The absence of studies of data quality and preparation for online social network (OSN) data makes this process even more tedious.

Data cleansing and feature engineering are two of the involved tasks in a preprocessing pipeline that can play a vital role in improving model interpretability, prediction accuracy, and performance. Cleansing is a long-standing task in data analysis that deals with missing, inaccurate, and noisy data. Feature engineering is a task linked to machine learning algorithms and modifies, creates, or eliminates unnecessary features from the dataset. Both tasks require domain expertise and could reduce computational cost and complexity [2] of algorithms.

To the best of our knowledge, extant research regarding big data is primarily focused on analysis, whereas the preprocessing aspects remain largely unexplored. In this chapter, we aim to study and uncover issues of the preprocessing stage of big data by taking as an example content data created in OSNs. We define noise in the data source and provide a holistic overview and classification of preprocessing challenges and solutions. We illustrate both statistical and qualitative methods for data cleansing. We outline textual data preparation tasks and feature engineering methods. Although the research is oriented toward social data networking analysis, the methods and algorithms reviewed are applied to any data source that shares similar data types.

The rest of this chapter is structured as follows. In Sect. 2, we present data types extracted from big data sources taking as an example that of OSN along with their analysis practices. The preprocessing stages and text are then introduced. The low-quality data related to them and the issues with regard to gathering and integration are briefly discussed. Qualitative and quantitative big data cleansing techniques are investigated in Sects. 3 and 4. Feature engineering and text preprocessing techniques are briefly discussed in Sect. 5. Contributions of a broad range of big data preprocessing frameworks and distributed frameworks for scalable analysis and preprocessing are presented in Sect. 6. Section 7 summarizes the main findings of this study and discusses the open issues raised for preprocessing big data in OSNs. It is worth noting that a single article cannot be a complete review of all the methods and algorithms. Yet, references cited can be used as guidance in narrower research directions in the field.

2 Data Quality Issues

There are many definitions of data quality and noise because both terms are domain dependent [3]. Generally, high-quality data are considered those that are representative for the indented problem or question being analyzed. Provided that the data are representative, data preprocessing tasks structure the data in a way that the analysis algorithms applied afterward can operate well enough so as to achieve high-accuracy results. Table 1 summarizes the application of the phases of data preprocessing for OSN.

2.1 Data Context

The study of online social networks (OSNs) is a quickly widening multidisciplinary area creating new research challenges for the understanding of the development and progression of socioeconomic phenomena. An SNS provides a specific type of service such as developing professional connections. The content and structural data occurred by this service are commonly represented in a large-scale graph G = (V, E), where V is a set of users or groups, called nodes which are connected by E, a set of relationships, called edges. A graph sometimes is defined by G = (V, E, A), where A is the adjacency matrix which shows which of the vertices are neighbors/adjacent to each other in the graph.

Big data sources, including that of OSNs, typically contain a tremendous amount of content and linkage data which can be leveraged for analysis. Depending on whether data are organized in a predefined manner, they are divided into structured

Table 1 Social networking data preprocessing tasks

Preprocessing task	Online social network
Noise treatment	Text informality
	Temporal ambiguity
	Spatial ambiguity
	Entity resolution
	Irrelevant data
	Outliers
	Mislabeled data
	Imbalanced data
Incomplete data	Content
	Nodes
	Links
	Attributes of nodes
	Labels of links
Data integration	Vocabulary schema
	Profile matching
Feature selection and scaling	Words
	Posts
	Tags
	Emoticons
	User profile
	Social relationships
	Interest relationships
	Activity relationships
	Time, date, and place
Feature extraction	Low-dimensional representation (e.g., word embedding)
	Annotation

and unstructured or semi-structured data. Content data are unstructured, while linkage data are graph structured.

The linkage data, also called graph data, is about patterns of interactions between the network entities (e.g., people, organizations, and items). They are divided into the following types:

(i) Activity-based relationships like “posted by” or “retweet”
(ii) Interest-based relations such as the number of likes or user-item ratings
(iii) Social-based relations such as number of common friends or the ratio of follower-followee, and network-structure features such as graph density

Content data, on the other hand, include multimedia, text, and clickstream data shared and created in the network. Analysis of posts, demographic and other user information, keywords, hashtags, and tags are all examples leveraged by social networking examples.

2.2 Data Analysis Types

Analysis practices make use of machine learning models which can be categorized into supervised, unsupervised, and reinforcement learning models.

Supervised is the case where the data used to train the model comprises examples both of the input vectors and their corresponding target vectors, whereas unsupervised is the case where the corresponding target vectors are not available. The reinforcement learning model is not common for social networking data analysis, and it will not be discussed any further.

Another distinction is between regression and classification. The former considers a continuous outcome variable while the latter a categorical one.

Prevalent analysis practices include the following:

1. Social network analysis (SNA)
2. Collaborative recommendation
3. Topic detection and tracking
4. Trend analysis
5. Sentiment analysis
6. Opinion mining

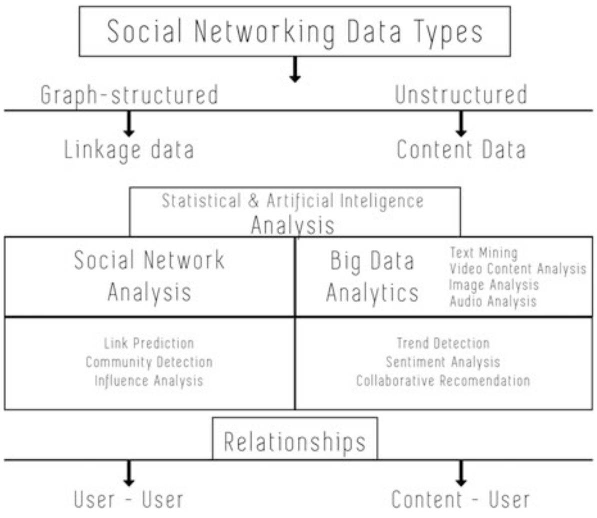
Supervised models are widely used in sentiment analysis and event detection while unsupervised learning in trend detection and collaborative filtering [4]. Clickstream and content data are enormously exploited in collaborative filtering, sentiment analysis, and opinion mining. Linkage data are extensively leveraged for social network analysis and collaborative filtering, and they are sometimes combined with content-based analytics [4]. Linkage and content data are usually grouped by geographical or temporal characteristics. Event detection, trend analysis, collaborative filtering, and SNA exploit frequently these features.

SNA does not fall into this categorization of models since it is a structured analysis for graphs. Other subsequent graph analytics techniques include information propagation, community detection, entity analysis, and link analysis. Table 2 summarizes the analysis practices and data structures in OSN.

2.3 Issues Solved by Preprocessing Tasks

Data preprocessing is divided into data cleansing, integration, transformation, and dimensionality reduction. It improves the overall generalization of the model by reducing variance (i.e., prevents overfitting) and reduces biased estimators (i.e., prevent underfitting). The aim of data reduction many times overlaps with that of transformation, and as such, there is no strict separation between the two. The two of them compose what is usually called feature engineering in a machine learning model.

Table 2 Social networking data analysis



Noise and missing values (MVs) are two long-standing issues on data instances. Building predictive or descriptive models on datasets containing missing or noisy values will affect the results. The magnitude of effect depends on the available data on hand and the processing requirements of the underlying analysis algorithm. To illustrate this with an example, naïve Bayes has a higher prediction accuracy on presence of missing data than decision trees. Data cleansing handles missing values, smoothes noisy data, and resolves inconsistencies and duplicates.

Noisy data are considered erroneous values, inconsistent data, and relevant data that get mixed with the irrelevant data. Noise is a term mostly being used in signals captured by sensors or machines. Random variation of brightness in a picture and high frequency in an electromagnetic signal are two examples of noise. Data inconsistencies commonly refer to the "noise" occurred because of erroneous entries by humans. The mismatched quantities between orders and sales of an inventory and between age and birth date of an individual are two examples of data inconsistencies. Noise and inconsistencies refer to the issue where different datasets that measure the same quantity do not agree with each other. It is worthy to note though that data inconsistencies are different to what is known as data outliers or anomalies, although the former can be seen as a generalization of the latter.

In structural analysis such as SNA, noise can be presented in terms of nodes. These include inactive nodes, artificial nodes, and duplicate nodes [5, 6]. Inactive nodes describe the fact of users who have an inactive account in social media, whereas artificial nodes represent spammers.

Analysis models cannot handle other values, e.g., blanks and Not A Number values, apart from numerical values. Additionally, missing data cause three main problems:

(i) Introducing a substantial amount of bias in the data generation process
(ii) Reducing the efficiency
(iii) Easiness of the implementation of analysis

The data generation process commands patterns of missing data which define the methods that should be used for resolving them. Specific data instances may be incomplete at random or not and may be caused by a human or a machine measurement error.

MVs in OSN are commonly caused by user-side values never edited in their profile or linkage data values that are not yet reflected in the network structure. Incomplete data could be distinguished in five forms for OSN: node, edge, attributes related to nodes, labels related to edges, and content components. MV in a structural analysis like SNA can have large negative effects on structural properties of the network [7–9].

Data integration covers the combination and loading of data residing in different sources that belong in the internal data ecosystem. An example includes merging data from different databases, such as a database with administrative data and dynamic data that come from the terminal of a client and being stored in the cloud.

Another challenge that big data has introduced is the high dimensionality in the feature or input space of unstructured or semi-structured data. Bellman referred to this phenomenon as the “curse of dimensionality” which denotes that as the dimensionality of the features rises, the amount of input data required to provide a reliable analysis grows exponentially [10]. The training time and memory usage then scale linearly with the number of input data points.

The feature space includes the independent variables, i.e., the attributes or otherwise called predictors, while target, class label, or outcome data include the dependent variable. The input space incorporates only the former or both. The literature sometimes refers to the feature space as the embedding space or as the vector space.

A large input space in terms of data instances can be handled with the utilization of scalable frameworks and architectures such as cloud computing, as briefly discussed in Sect. 6.1. A large feature space, on the other hand, is a much more sophisticated issue and introduces extra complexity to the algorithms. This is handled via feature engineering tasks, discussed in Sect. 5. Feature engineering shortens the training time of the algorithm which is a solution for overfitting.

2.4 Text Processing

Text can introduce inconsistencies, duplicates, and erroneous values, and it is handled via Natural Language Processing (NLP) techniques. Text analysis in online networks suffers additional problems because of the unstructured and often informal content posted in the network. The use of slang, abbreviations, and grammatically incorrect words should be taken into consideration. The fragmented and short in length posts challenge even more the analysis algorithms. Providing structure to emoticons is another common preprocessing task. In case of text retrieval, URLs, stop words, quotations, punctuations, special characters, foreign words, extra spaces, and line breaks are always removed.

2.5 Entity Resolution

Entity resolution (ER) is needed to deduplicate and disambiguate the data. It is further discussed in subsect. 5.3.2. There are different semantic types both within edges and nodes in a source and across sources. For example, the network of Facebook in addition to friendship relationships between persons has got relationships of other types, such as person-photo tagging relationships. Combining data from multiple social networks causes a bigger expansion of relationships which sometimes lead in overlapping. For instance, when developing an interest graph (what people care about) by integrating data from multiple OSNs, there is ambiguity of words for the same interests such as "Likes" on Facebook or "Interests" in a LinkedIn profile. This data heterogeneity stems from the following differences in:

1. Relationship types and content sharing defined by an SNS
2. Terminology frameworks and the absence of a cross-domain vocabulary matching [11]

Duplicate nodes may occur as for example in a discussion network when person A replies to person B on multiple occasions. This makes the measurement of network metrics such as degree to be inaccurate. A solution to duplicate nodes is to be combined into a single weighted edge.

2.6 Data Extraction

Imperfect data acquisition process and communication failure while extracting data from the web are common problems in large-scale data retrieval.

The standard means of gathering data from the web including social networking data are the Application Public Interfaces (APIs). It is important to consider the traffic and networking aspects of the websites while extracting data in order to avoid communication failures which lead to incomplete data.

Providing some form of structure while extracting information from OSN is the first preprocessing step. This is done with the usage of lexicon resources and matching algorithms, or supervised classification models. There are also filtering rules provided by SNS API functionality and by data providers like DataSift, Gnip, and Spinn3r. These rules are commonly based on keywords, geo boundaries, conversation details, hashtags, phrase matches, and the activity type like share, comment, and like. A quality issue that arises from the extraction process is whether the samples obtained via stream APIs and the filtering functions discard important data and bias the results.

Other irrelevant information that is considered noise in OSN and is omitted includes web robots, extensions of CSS, GTF, FLV, and the records with failed HTTP request.

3 Big Data Cleansing Approaches

The beforementioned issues can be handled via preprocessing tasks, implemented in the following:

(i) A qualitative way via the specification of rules and constraints
(ii) A quantitative way via the specification of statistical or machine learning models

Cleansing low-quality data is a hard-computational task that has been studied for decades. Cleansing data includes exploration and pattern recognition.

Understanding data dimensions and possible sources of errors is a first step toward developing a data cleansing technique [3], known as exploratory data analysis or understanding. Exploratory analysis is based on statistical measures such as frequencies, means, variances, and standard deviations used to collect statistics about individual attributes so as to understand the validness of the dataset and discover metadata. Regarding unstructured data, a searchable "map" of the metadata and full-text searches is created via exploratory analysis.

Pattern recognition includes usually unsupervised tasks like correlation and association rules. Pattern recognition and machine learning can be viewed as two facets of the same field.

Data cleansing is a multistage process that includes the following stages [12–14]:

1. Determine and/or identify data errors.
2. Correct errors by repairing or deleting them.
3. Document error examples and error types.
4. Modify data entry procedures to reduce future errors.

After data profiling and pattern detection, rules are derived for step (1), and then the data cleansing process iteratively runs steps (2) and (3). This process is also known as data munging in the case of predictive models. Effective big data munging requires the identification of a policy that specifies which correction has to be applied when an error can be corrected in several ways. Repair can be performed by using machines, human crowds, or a hybrid of both.

3.1 Error Identification

In step (1), errors can be specified either logically (qualitative) or statistically (quantitative) by knowledge bases [15], crowdsourcing [16], and domain experts or in a data-driven manner [17]. Table 3 illustrates big data preprocessing according to three variables: how, where, and by whom the data errors are detected and repaired. We analyze it gradually in this chapter.

Data-driven cleaning allows the rules to be learned from the dirty data itself and be validated incrementally as more data are gathered. Researchers [1] provide a big data quality management view that corrects data via a dependency model whose rules are learned by the data itself. Using information from the same OSN to cleanse and structure data produces more accurate results.

However, since data themselves are dirty, the need is to make these rules robust against outliers and to allow approximation [1]. Soft computing techniques are tolerant on imprecision and approximation.

Crowdsourcing [18] is used for labeling data for data cleansing and supervised learning. Active learning in crowdsourcing is often being used in combination with crowdsourcing, in which learning algorithms interactively query the user to provide labels to the "necessary" (i.e., unusual, uncertain) data points.

Knowledge bases and lexical resources have been extensively used to filter out noise [19–21]. WordNet is widely used for topic detection, while SenticNet is widely used for sentiment detection. The former covers semantic and lexical relations between terms and their meaning such as synonymy, hyponymy, and polysemy. SenticNet, on the other hand, includes the polarity of common-sense concepts. The limitation of using lexical resources in OSNs is that they do not contain many words or concepts being utilized in informal contexts. In addition to that, the development of non-English bases is needed. Many social networking data analyses have been done through the use of manually built keyword lexicons whose limitations for big data are apparent with the most obvious one to be their inability to scale.

The two approaches to cleanse and preprocess messy data are the qualitative, which follows the dependency theory, and the quantitative [1, 18]. Quantitative methods are listed as follows:

Table 3 Data cleansing techniques categorization

	How		
Who	Qualitative	Quantitative	Where
Knowledge base	Dependency rules: FD, CFD, DC	Natural language processing	Source (e.g., ETL)
Data		Statistics	Store (e.g., ELT)
Domain experts	Matching rules	Machine learning	
Crowdsourcing		Graph mining	

1. Statistics
2. Machine learning
3. NLP
4. Graph theory

3.2 Dependency Rules

The qualitative process uses logical reasoning over declarative data dependencies to detect errors and cleanse data. There are several classes of qualitative rules including functional dependencies (FD) and their extensions to conditional functional dependencies (CFD), and denial constraints (DC). Although these dependencies have been of paramount importance in database theory, they have been used in different fields like artificial intelligence and data processing.

Each rule usually targets a specific data quality error, which may be an instance feature or the whole dataset level, but interacting two types of quality rules may produce higher-quality results [22, 23].

Researchers [1, 12, 24, 25] refer to big data quality with a focus on CFD and FD because they specify a fundamental part of the semantics of data, and they are hence particularly important to real-life graphs. Finally, unlike relational data, graphs [24] and unstructured data typically do not come with a schema, and dependency rules have to specify not only the regularity between attribute values of the data points but also their topological structures.

Additionally, dependency rules like FD, CFD, and DC are considered as hard constraints where a specific set of conditions for the variables have to be satisfied. Data-driven cleansing approximated rules should be preferred against hard coded rules [1] in case of big data. The advantage of qualitative methods is the domain knowledge incorporation [1, 18]; quantitative techniques though are less prone to error introduced by domain knowledge.

3.3 Statistics and Machine Learning

Many qualitative techniques are amenable to a statistical analysis, and sometimes, there is an overlapping. There are also hybrid techniques [23, 26] that combine dependency rules with statistical methods. Qualitative techniques are more difficult to be implemented for big data sources except if there is an explicit set of rules and constraints that these should satisfy; hence, quantitative techniques are many times preferred. However promising, we do not cover graph theory for big data preprocessing apart from referring a few distinct examples.

Discovering dependency-based rules and computing a repair are cost prohibitive for big data. To cope with this, parallel scalable algorithms and distributed map-reduce processing may be used to lessen the issue of data volume; parallel incremental algorithms, on the other hand, can deal with the velocity problem [1, 15]. Yet, rules are dependent both on the application and the data source hindering thus the scalability of a qualitative data cleanser to another application.

The quantitative approach relies traditionally on the use of statistical methods; recently, machine learning models are incorporated in this process [18]. Machine learning is closely related to statistics on that they both find structures and patterns in data. However, statistics emphasizes in low-dimensional problems, whereas machine learning in high-dimensional problems. Additionally, algorithms inspired by biological procedures found in nature belong to machine learning. Both machine learning and statistical models need to be modified in order to work for big data cases [27, 28].

Modifications of statistical models for big data should handle the unique features of incidental endogeneity, heterogeneity, spurious correlation, and noise accumulation [27–30]. Sophisticated model selection with modified regularized least square methods should handle incidental endogeneity. Efficient computation algorithms with regularization can deal with the heterogeneity to tackle the issue of statistical significance. Cross validation is a model selection technique. It subtracts the effects that spurious correlation has on feature selection and statistical significance. Feature engineering, discussed in Sect. 5, is proposed for addressing noise accumulation issues.

Many times, different terms in the areas of machine learning, statistics, or computer science are being used to describe the same entity. Table 4 depicts a few of these terms so as to help readers to have a broad idea of the terminology. Technically, these terms are not equivalent.

4 Machine Learning Algorithms for Preprocessing and Analysis

Machine learning models can produce false predictions in presence of noise and MVs.

In case of classification analysis, noise is divided into attribute and class noise. The latter includes contradictory examples (examples with identical input attribute values having different class labels) and misclassifications (examples which are incorrectly labeled). The objective of data extraction [19, 20], discussed in Sect. 2.6, is to distinguish relevant from irrelevant data. Supervised classifiers are widely used for this task.

In case of regression models, the independent variables are also considered noisy when there is dependence among them, namely, when they are not sampled independently from some underlying distribution.

Table 4 Different terminology used in machine learning, statistics, and computer science

Term	Similar terms
Observations	Data points
	Samples
	Instances
	Records
	Rows
Features	Independent variable
	Explanatory variable
	Attributes
	Columns
Target	Dependent variable
	Outcome variable
	Output
	Class or label (in classification)
Feature space	Vector space
	Embedding space
Data structure	Matrix
	Array
	Vector
	Tensor
	Graph
Penalization	Regularization
Estimation	Prediction
	Learning

4.1 Supervised Machine Learning

Predictive classifiers have been extensively studied and have shown remarkable success for analysis practices such as text classification, sentiment analysis, and event or topic detection [4, 19–21]. They are employed for preprocessing tasks such as classifying irrelevant data [6] and predicting missing values.

Supervised classification algorithms are affected from unnormalized data and missing values. Normalization is discussed in Sect. 5.1. Imputing missing values with the mean for all data points belonging to the same class is considered as a simple and effective strategy. A comparative study of imputing methods for supervised models is provided by [31].

4.1.1 Imbalanced Data

An additional issue is that of imbalanced data which occurs when there are more training examples for one class than another. This can cause the decision boundary to be biased toward the majority class.

Resampling techniques are used to balance the class distribution. The two main groups within resampling are under-sampling and oversampling methods [2, 32]. The first one creates a subset of the original dataset by eliminating the majority examples, whereas the second creates a superset of the original dataset by replicating or generating examples from existing ones. The advantageous part of resampling is that it is independent with the analysis algorithm applied afterward.

SMOTe is a recent oversampling technique that employs the Euclidean distance in nearest neighbors between data points in feature space.

Ensemble classification methods are capable of dealing with the imbalanced dataset problem [32, 33]. The core idea imitates the human desire to obtain several opinions before making any crucial decision. Combining different types of classifiers helps in improving predictive performance; this occurs mainly due to the phenomenon that various types of classifiers have different "inductive biases" [33].

Ensemble classifiers are divided into boosting, bagging, and stacking; and algorithms include AdaBoost, Rotation Forest, Random Forest, and LogitBoost. These are well suited when small differences in the training data produce very different classifiers like in decision trees.

Algorithmic modifications are also proposed to deal with the imbalanced data issue. Learning the features for a class using all training data being not in that class [34] is one such technique.

4.1.2 Algorithms and Preprocessing Requirements

Naïve Bayes is a probabilistic classifier that uses the Bayes theorem to predict the probability that a given feature (x) belongs to a particular class (c). Let us consider an example of classifying malicious mails, $c = \textit{malicious}$, and a feature may be the presence of a text such as $x{=}'\textit{urgent action required}'$. Equation (1) shows the posterior probability of $P(c|\,x)$ of the mail being malicious given the specific feature. The prior probability $P(c)$ is the initial belief of the mail being malicious before observing any data, P(x) is the marginal probability of observing that feature, and $P(x|\,c)$ is the likelihood of having x given c.

$$P\,(c|x) = \frac{P(c) * P\,(x|c)}{P(x)} \tag{1}$$

Given the naïve assumption of n number features being all independent, (1) is rewritten as (2) depicts

$$P\,(c|x) = \frac{P(c) * P\,(x_1|c) * P\,(x_2|c) * \ldots P\,(x_n|c)}{P(x)} \tag{2}$$

Because of the independence assumption, this classifier is highly scalable and can quickly learn to use high-dimensional features with limited training data. It has higher prediction accuracy on presence of missing data than many other classifiers like decision trees and neural networks. It can automatically ignore them while preparing the model and calculating the probability for a class value.

Support vector machines (SVM) is a kernel method and can be used for classification or regression. Each data instance is plotted as a point in an n-dimensional space with the value of each feature being the value of a particular coordinate. Then, the next step is to determine the decision boundary, which can best classify the different labeled classes. Equation (3) shows the optimization function of SVM.

$$\min_{\theta} C \begin{bmatrix} \sum_{i=1}^{m} y_i \left(-\log h_{\theta}\left(x_i\right)\right) + \\ \left(1-y\right)\left(\left(-\log\left(1-h_{\theta}\left(x_i\right)\right)\right)\right) \end{bmatrix} + \frac{1}{2}\sum_{j=1}^{n} \vartheta_j^2 \tag{3}$$

where $C = \lambda/m$, where λ is a regularization term such as ridge and m is the number of labeled data with x_i being the input and $y_i$the corresponding label, presented as a matrix of $\{(x_i, y_i))\}$ where $\boldsymbol{i} \in (1 \ldots m$); j $\in (1 \ldots n)$ is the weight for a specific feature and n x_i, n is the total number of features. Equation (4) calculates the linear hypothesis $h_{\theta}(x)$ as follows:

$$h_{\vartheta}(x) = \theta_0 + \theta_1 x_1 \cdots + \theta_n x_n \tag{4}$$

Reliance on boundary cases enables it to handle missing data. Heuristics and resampling may reduce the effect of imbalanced data in SVM modeling. Changing the value of C so that weight is inversely proportional to class is one simple solution. SVM has been used for filtering breaking news-related tweets [19] and for name resolution in a disambiguation framework [25].

Logistic regression is used to predict types of repairs needed to resolve data inconsistencies based in a set of statistics computed over the data and declarative rules [17]. It is a probabilistic, linear classifier that used to transform a linear combination of input features into a nonlinear output via the sigmoid function as shown in Eq. (5). It can also be used for missing values imputation.

$$\sigma\ (\mathrm{t}) = \frac{1}{1 + e^{-h_{\theta}(x)}} \tag{5}$$

LogitBoost and Logistic Model Tree algorithms are more appropriate algorithms for imbalanced datasets [6]. Logistics Model Tree combines logistic regression and decision tree learning, while LogitBoost combines AdaBoost and logistic regression.

The traditional algorithms cannot work for high number of features or examples. But, many attempts have been done to make it scalable for large data sets. Online stochastic gradient descent algorithm can respond to the data volume.

Decision tree is a supervised learning method used for classification and regression. It decomposes the data space into homogeneous groups based on independent variables' conditions, called as split points. The division is done recursively until the leaf nodes contain certain minimum numbers of records which are used for the purpose of classification. One of the benefits of decision trees is that input data does not require any scaling. However, they cannot work with missing values. Decision tree classifiers can handle imbalanced data better than other algorithms, but this always depends on the distribution of features in the space.

Decision trees are usually utilized with boosting, such as in logistic model tree, in order to reduce the bias and variance [6]. Random Forest and Rotation Forest use multiple decision trees. The latter is a state-of-the-art method where decision trees are independently trained via principal component analysis (PCA).

4.2 Unsupervised Machine Learning

There has been extensive work on unsupervised models and specifically those of anomaly or outlier detection, regression, and clustering for noise treatment and missing values imputation.

4.2.1 Imputation of Missing Values

A basic strategy to use incomplete datasets is to remove entire data instances, i.e., rows – known as listwise imputation and/or columns containing missing values. However, this many times comes at the price of discarding valuable information. Listwise deletion works well only when the data are missed at random. Regarding column deletion, if the variable with the missing value depends on at least one of the other variables of the model [35], it is not a good practice.

Inferring the missing values from the known part of the data is a better strategy, known as imputation. The inferred values may be constant or statistical measures. Conventional methods include marginal mean imputation and conditional mean imputation.

Cold-deck imputation selects the replaced value from central tendency measures such as the mean or mode of the feature. Marginal mean imputation is computing the overall mean to impute the missing value. For a categorical feature, a mode is used instead. In conditional mean imputation, the values for the missing rows are imputed conditioned on the values of the complete attributes that are most similar to it. To illustrate this with an example, a conditional imputation rule could state that the mean value for the "chest pain type" feature is imputed only if the value of "diagnosis" feature is absent. Although these are both simple and fast methods, they come with many disadvantages with the most obvious being the reduction of variance in the dataset. This in general may lead to biased estimators for the predictive model. A more advanced method called hot-deck imputation has been proposed as an alternative [35].

Hot-deck imputation, by contrast, selects the replaced value from a similar data instance. Similarity is computed via proximity-based measures. This technique may yield biased results irrespective of the missing data mechanism, it may become less likely to find matches if the number of variables is large, and a random value is chosen when many instances are found to be similar.

Statistical imputation, also known as a sequential regressor, is a multivariate imputation technique that imputes the values based on all the feature dimensions of the dataset. The feature with the missing values is treated as the dependent variable of the regression function and the other features as independent variables. The estimator is fitted on the data and then predicts the missing value. This is done iteratively for each feature and repeated for a given number of rounds. The results of the final imputation round are returned.

These imputation methods usually are restricted to cases with a small number of features. The same is applied in large-scale linear regression with conventional missing values imputation methods like eyeballing and interpolation [30]. An optimized offline version of linear regression is proposed in [30] based on adding penalty factors to the original output value in order to diminish the error further. Researchers also enabled high-dimensional linear regression [36] with noisy, missing, and/or dependent data.

A popular unsupervised approach that uses proximity measures is the nearest neighbor method [37]. The relationship among features is considered when computing the degree of distance and can be implemented for high-dimensional data.

Distribution-based methods assign the value based on the probability distribution of the non-missing data. The expectation–maximization algorithm and heuristic methods are the most common and are used also for resolving noisy values. However, they can only be used for linear and log-linear models [35], and it is difficult to guarantee that the local optimum is close to a global optimum [36].

For categorical data, the following techniques are utilized for advanced imputation of missing values:

1. Multinomial inverse regression
2. Regression topic models
3. Penalized regression like lasso
4. Contextual anomaly detection
5. Latent factor models

Structural features of the network together with graph analysis algorithms can also be used as imputation methods [30]. Predicting missing attributes' values of users' can be done with a community detection algorithm proposed by [38].

There are imputation techniques for two-dimensional regression models, but only recently such techniques extended to high-dimensional data [30, 36].

4.2.2 Anomaly Detection

Anomaly detection deals with the identification of abnormal or unusual observations in the data by defining a decision boundary around normal observations. However, defining the region which separates outliers from normal data points is not a straightforward task, and it changes over time. Feature engineering choices have also to be carefully considered in order to not bias anomaly detection.

Outlier detection and novelty detection are both used for anomaly detection. Outlier detection is then also known as unsupervised anomaly detection and novelty detection as semi-supervised anomaly detection. Outlier detection includes the whole dataset, and it defines a decision boundary in the training data. Novelty detection defines a dataset of "good" data and in real time keeps or discards observations that fit the dataset. It is similar to "one-class classification," in which a model is constructed to describe "normal" observations. The difficulty lies on that the quantity of available "abnormal" data is insufficient to construct explicit models. SVMs have been used for novelty detection.

Distributed strategies for outlier detection in large data sets are Distributed Solving Set and Lazy Distributed Solving Set algorithms.

Duplicates are considered a form of noise. Matching rules and ER are common methods to address this issue, discussed in Sect. 5.3.

However, outliers in OSN may reflect real behaviors which necessitate an interpretable model in which explanations of "normal" data are to be provided.

4.3 Extracting and Integrating Data

Sanity checking is usually done after the extraction process in order to ensure that the desired, filtered data structure is gathered. A common cleansing task after extraction is to convert the data to the one format needed for the analysis algorithm. APIs exchange data mostly in the two formats of XML and JSON. These are semi-structured formats whose cleansing is largely unexplored [18]. Additionally, API's responses format and structure differ from one OSN to another demanding data conversion.

Social networking data are semi-structured and not consistent in "schema" level. This inconsistency grows when combining different social networks in order to perform cross-domain analytics, which are considered highly valuable. A solution [11] is the development of a unified conceptual data model that captures differences and similarities in structures and terminologies. Considering that there might be hidden relationship among features from different source, correlation-based algorithms can be used to find the most relevant relationship among features from different sources [30].

Dealing with the granularity and the hierarchies of data residing in different sources is a general issue for big data. Comprehending the scope, hierarchy, and granularity [27] of the social networking data via domain-encompassing models is necessary in data integration.

4.3.1 Data Warehouses

Data warehouses are the mainstream database for integrating data. They require concise, entity-oriented schemas that facilitate the analysis of data. Data cleansing in these databases is a major part of the so-called Extract, Transform, and Load (ETL) process. With cloud and similar infrastructures, an alternative approach called ELT has emerged. Its advantage over ETL is that data can be loaded in their raw form and thus be used at any point in time. ETL can be viewed as "bring the code to the data," whereas ELT as "bring the data to the code" [27]. In any case, researchers can keep track of data provenance for different analysis tasks. The issue that arises from data warehouses is the timeliness dimension of data that having the data residing for a long time into a database is that they may become obsolete for a specific analysis. The data quality problems most often encountered in OSN and the preprocessing tasks related to them are briefly summarized in Table 1.

5 Feature Engineering and Text Preparation

Having a good representation of features is linked with the success of the machine learning models [50]. Feature engineering is composed of transformation and dimensionality reduction techniques.

5.1 Transformation of Features or Samples

Two common transformation techniques that deal with the range of values of features include discretization and encoding. The former converts the continuous features into their discrete counterparts, and it is necessary in algorithms that handle discrete only data, such as in probabilistic density estimators. Data encoding converts categorical values of features to their numerical counterparts; it is necessary to any algorithm since no algorithm can handle categorical values.

Aggregation is a simple exploration-transformation method that groups or summarizes data (e.g., computing the minimum of samples within a feature or creating a pivot table by combining three features). It is an informative step for checking the quantity of outliers or sparseness of features, or correlations in the data.

Feature scaling scales all samples within an individual feature to satisfy a particular property in the entire range of their values. Standardization ranges them to have unit variance and mean of zero, like the samples are drawn from a standard Gaussian distribution. In practice, however, when there is no interest about the shape of a distribution, it centers the values by removing the mean of each feature and then scales them by dividing it with standard deviation. Sometimes, this is referred to as normalization. In the documentation of the scikit python library though normalization is the process that scales individual samples rather than features to have a unit form; this is useful when applying a dot product or computing the similarity between samples or creating a vector space model in text preparation.

There are other choices for feature scaling. They depend on the available characteristics of the data set and the analysis algorithm. For instance, if there are many outliers in the data, then standardization biases the information available in the data, while if there is a lot of sparsity, then centering the values can also harm the model. Nonlinear scaling is applied when the rank of the values along each feature has to be preserved.

Feature selection selects the most relevant features from the original dataset. It can be divided into lexicon-based that make use of knowledge bases, discussed in Sect. 3.1, and statistical methods. The latter include lasso and ridge and the selection of the value of the lambda hyperparameter in SVM depicted in Eq. (3), which controls the magnitude of the selected features. Decision tree estimator is another technique that computes the feature importance to the model's prediction. Recursive and backward elimination and rank algorithms like information gain and correlation coefficient are also employed. A feature selection survey is provided in [10].

5.2 Dimensionality Reduction

In the dimensionality reduction or otherwise called feature extraction, the original data gathered are replaced with a lower-dimensional approximate representation obtained via a matrix or multidirectional array factorization or decomposition. They can be divided into two groups, linear and nonlinear methods [10].

Linear feature extraction assumes that the data lies on a lower-dimensional linear subspace and projects them on this subspace using matrix factorization. The linear methods transform original variables into a new variable using the linear combination of the original variables. Some examples include the following:

- Principal component analysis (PCA)
- Linear discriminant analysis
- Probabilistic PCA
- Factor analysis
- Linear regression
- Singular value decomposition (SVD)

Nonlinear dimensionality reduction works in the following different ways:

(i) Kernel-based methods where a low-dimensional space can be mapped into a high-dimensional space so that a nonlinear relationship among the features can be found

Table 5 N-gram model

Unit	Sample sequence	Unigram BoW	Bigram BoW
Word	. . . As knowledge increases wonder . . .	. . . As, knowledge, increases, wonder, . . .	. . . As knowledge, increases wonder, . . .
Character	. . . to_be_or_not_to_be . . .	. . . , t, o, _, b, e, _, o, r, _, n, o, t, _, t, o, _, b, e, . . .	. . . , to, o_, _b, be, e_, _o, or, r_, _n, no, ot, t_, _t, to, o_, _b, be, . . .

(ii) Kernel PCA for nonlinear classification
(iii) Self-organizing maps or Kohonen, which creates a lower-dimensional mapping of an input by preserving its topological characteristics

Feature construction/extraction plays an important role often in reducing the misclassification error [2]. It creates new variables and increases the variety of data by constructing new features either on transforming the original data for dimensionality reduction or on addition of domain knowledge for algorithm performance improvement. The former is also known as data augmentation. A feature engineering task that combines weathers' data with citizens' transportations patterns could increase the performance of the algorithm if there is such a relation in the data. Adding noise such as changing the curvature of letters in a speech recognition task or blurring the image in an image recognition task includes two illustrations of data augmentation.

5.3 Natural Language Processing

The preprocessing of textual data extracted from a document starts with a tokenization step, namely, breaking a stream of text into words, phrases, symbols, or other meaningful lexical items that will create a dictionary of tokens.

5.3.1 Text Extraction

The most common representation of these tokens is the Bag of Words (BoW) model, also referred as N-gram model. An N-gram model via language units is shown in Table 5.

The BoW model correctly separates phrases, words that have meaning only when they occur together, in clusters that refer to the same categories, but it introduces spurious correlations [20]. That is, phrases are added to a cluster just because they share a word in common. To get rid of this spurious correlation, the words that appear together all the time (i.e., San Francisco) were merged, whereas words that appear both solely and in phrases (i.e., Barack Obama) were distinguished, and only the word that appeared more often was chosen (i.e., Obama).

Despite the wide use of BoW, there is a need for richer and distributed representations. Bag of Concept has been proposed as a replacement which makes use of deep learning. This is capable of capturing high-level structures of natural language and incorporating semantic relationships among them.

5.3.2 Feature Selection and Extraction

The vector space model is the data structure for a piece of text composed by a feature vector of terms and term weight pairs.

Word embedding is a feature extraction technique in NLP that uses distributed word representations. Words or phrases are mapped to vectors of real numbers resulting in a low-dimensional feature space. Methods that generate this mapping include neural networks, probabilistic models, and dimensionality reduction on the word co-occurrence matrix. The word2vec is a tool that takes a text corpus as input and produces the word vectors as output via neural networks. It first constructs a vocabulary from the training text data and then learns vector representation of words.

Normalized data are needed for the vector space model used in text classification and clustering contexts. Discretization is also common among the text processing community (probably to simplify the probabilistic reasoning); despite that, normalized counts (i.e., term frequencies) or TF-IDF valued features often perform slightly better in practice.

Feature selection algorithms applied in text preprocessing include TF-IDF, Entity Resolution, Mutual Information, Information Gain, and Chi square. These statistical analysis methods used to measure the dependency between the word and the target output like the category of a tweet as they are mentioned in Sect. 3.2. Semantic relations between words can be found via stemming and lemmatization. Both of them aim to reduce inflectional forms, but the former removes derivational affixes by applying a set of rules, whereas the latter applies morphological analysis of words with the use of a vocabulary. Stemming is just structural, whereas lemmatization is contextual and can consider synonyms and antonyms.

TF-IDF measures the significance of words in text. TF is the occurrence of the term appearing in the document: $dt_f = (tf_1, tf_2, tf_3, \ldots tf_n)$, where tf_i is the frequency of the i-th term of the document d and n is the total number of terms. IDF gives higher weight to terms that only occur in a few documents, and it is defined as the fraction: N/df_i, where N is the total number of documents. This means that common tokens will receive a lower score because they are terms that occur in several documents. TF-IDF is used to deal with typographical variations, e.g., "John Smith" versus "Smith, John," but does cannot take misspellings into account [6].

ER is used in detecting data duplicate representations for the same entities and merging them into single representations. It compares pairs of entities and evaluates them via multiple similarity measures. It is widely employed in NLP, but it is associated with high complexity. Using blocking techniques in order to place similar entity descriptions into blocks and thus only compare descriptions within the same block is recommended as a solution [14, 39]. Matching dependencies is the corresponding qualitative technique where declarative similarity rules are specified. SVM [25] has also been proposed among other classification models for disambiguation. Cluster computing frameworks such as MapReduce and Spark are well suited for these techniques due to their inherent complexity.

In contrast to TF-IDF, ER does not ignore the sentence structure. It is used in repairing misspellings, name resolution [6], toponym resolution, duplicates (D-dupe), slang, acronyms, and grammar issues. In text deduplication, semantic-based ER measures how two values, lexicographically different, are semantically similar, while syntactic-based ER computes the distance between two values that have a limited number of different characters. Knowledge bases [15] can be useful in similarity and matching functions. Addressing slang words has also been approached via BoW and conditional random fields [40]. Correcting typographical errors is done by [5] via random walks on a contextual similarity bipartite graph constructed from n-gram sequences.

Named entity recognition (NER), part of speech (POS), and stemming are employed to resolve the ambiguities of time and location expressions [19–21]. NER detects an entity in a sentence (London) and assigns a semantic class on it (city). POS tagging is used to assign part of speech tags to a sequence of words and is widely used for language disambiguation tasks. Stochastic models of POS are used by researchers in [41, 42] for social media texts. The former research used conditional random field and the latter the first-order maximum entropy Markov model.

However, it is not easy to accurately identify neither named entities or part of speech words in social networked data which contain a lot of misspellings and abbreviations. POS and NER are trained on full-text documents instead of text extracted from OSN. Hence, researchers often resort in poorer representations like that of TF-IDF [20].

6 Preprocessing Frameworks

Platforms for distributed computing like cloud and grid together with distributed frameworks like Apache Spark and Hadoop MapReduce are used to deal with the high volume of data. They compute scalable, data-intensive machine learning algorithms for preprocessing data. Time-consuming algorithms though should be redesigned in order to allow for scalability and parallelization.

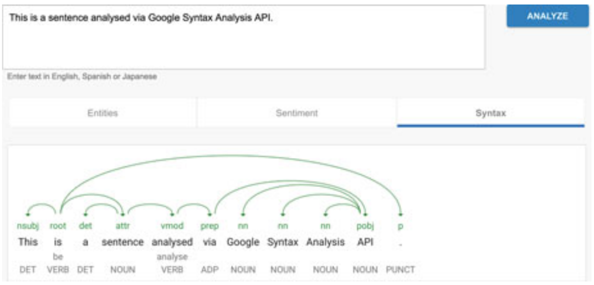
6.1 Distributed Frameworks

Cluster computing frameworks can be divided into online and offline processing. Apache Hadoop is a very popular implementation of MapReduce distributed processing model which is based on offline processing. Although it supports the analysis of unstructured data, it cannot deal with streaming data and is not suitable for iterative processes. A map-reduce procedure can only conduct one pass over the data points while trying to optimize the execution of a single iteration which is done manually by tuning each map-reduce step.

Spark can deal with iterative processes effectively and efficiently compared to Hadoop's two-stage MapReduce paradigm. Its parallel execution model enables all data to be loaded into memory and avoid the I/O bottleneck which benefits the iterative computation. The execution plan is optimized via flexible directed acyclic graph-based data flows which can significantly speed up the computation of the iterative algorithms. Spark is the most common clustering framework used for in real-time big data analytics. Other frameworks that process incoming continuous data streams include Indoop, Muppet, and SCALLA.

Mahoot is the machine learning library provided by Hadoop. NLP tasks like deduplication, typographical error correction, normalization of dates, places and acronyms, and other word error correction that can be implemented on social media posts can be executed. Hadoop MapReduce has gained popularity implementing ELT processing. MLlib is the corresponding machine learning library of Spark. It supports TF-IDF, syntactic ER, word embeddings, and stop word removal among other functionalities.

Cloud computing makes available hundreds or thousands of machines to provide services such as computing and storage. Text processing, data preparation, and predictive modeling are all supported by cloud-based API services. Google Cloud Platform hosts machine learning algorithms for natural language processing like POS tagging and NER. An example of data parsing with Google NLP API is depicted in Fig. 2. Google Cloud Dataprep can be used to prepare data through basic statistical measures given that these are transformed in a relational form like BigQuery tables. Although cloud computing technology has been applied in the field of machine learning, there is still no real application to big data cleansing algorithms [43].


This is a sentence analysed via Google Syntax Analysis API.

```
nsubj  root  det   attr      vmod      prep  nn      nn      nn        pobj  p
This   is    a     sentence  analysed  via   Google  Syntax  Analysis  API   .
       be                    analyse
DET    VERB  DET   NOUN      VERB      ADP   NOUN    NOUN    NOUN      NOUN  PUNCT
```

Fig. 2 Dependency parsed tree generated via Google NLP API

6.2 Preprocessing Frameworks for OSN

Table 6 categorizes big data preprocessing frameworks discussed in this chapter, according to the approach they follow, the data errors they are capable to handle, and the kind of service they can provide at social networks.

Analysis of opinionated text like sentiment analysis and opinion mining could incorporate preprocessing frameworks by [5, 44] to normalize text. In [44], researchers built a corpus with cleaned data that can harm sentiment analysis algorithms. Lemmatization, TF-IDF, and the removal of stop words and other noisy information to handle abbreviations and typographical errors are then employed. This is oriented to Twitter, while in [5], a text normalization system adaptable to any social media and language is proposed. It corrects typographical errors via unsupervised approach that learns the normalization candidates from unlabeled text data and maps the noisy form of the word to a normalized form.

Regarding deduplication, many frameworks have been proposed. Dedoop [39] deals with parallel deduplication via implementing ER on MapReduce, and cloud is used to handle the volume of big data. The ER workflow consists of three consecutive steps: blocking, similarity computation, and the actual match decision. In [14], a distributed big data deduplication system which utilizes MapReduce is also developed. The difference with Dedoop is that [14] provided a communication cost model. D-Dupe [45] is another ER tool oriented for social networks. It integrates data mining algorithms with an interactive information visualization interface that can be also used from non-experts.

Dependency rules have been utilized in big data preprocessing frameworks. BigDansing [13] deals with the volume by transforming functional dependencies, denial constraints, or user-defined function in order to enable distributed computations for a scalable data cleansing. It can run on top of most common general-purpose data processing platforms, ranging from DBMSs to MapReduce-like framework. NADEEF [22, 46] is a cleaning system that hosts ETL rules, CFDs, FDs, deduplication rules, and user-defined rules. Another data cleaning framework is LLUNATIC [47] which develops a parallel-chase procedure that detects violations of these rules and guarantees both generality and scalability. A generalized framework for repair computation given semantic qualitative rules is Katara [16]. It bridges crowdsourcing and knowledge bases to find semantic table patterns at the instance level via SPARQL queries. These are visualized as a labeled graph where a node represents an attribute and an edge the relationship between two attributes. An adaptive cleaning framework [17] capable of dealing with velocity is proposed for dynamic environments without fixed constraints. They presented a classifier that predicts the type of repair needed to resolve an inconsistency and automatically learns user repair preferences over time. A framework for missing consistent attributes that combines Bayesian inferential statistics and accuracy rules based on data semantics is proposed by [23]. Aetas system [26] combines FDs with probabilistic and outlier detection models on data modeled as dependency cubes.

Table 6 Big data preprocessing frameworks

	Approach					Data errors			
	Dependency rules	NLP	Machine learning	Graph theory	Distributed	MV's	Noise	Duplicate	Service
[6]		TF-IDF, ER	Boosting, Random Forest, Logistic model tree, Rotation Forest						Name Resolution
[38]				Community Detection, Betweeness centrality					Missing attributes
[23]	FD,CFD		Bayesian model						Deduplicate attributes values
[33]			Decision tree, Bagging, Random forest						Missing links
[30]			Linear regression		Spark				Missing numerical values
[39]		TF-IDF, ER			MapReduce				Entity Deduplication
[14]			ER		MapReduce				Distributed deduplication
[5]		ER		Graph random walks	Algorithm				Text normalization
[19]		NER, POS	SVM, Bayessian inference						Filter noise
[44]		TF-IDF			MapReduce				Text normalization
[17]	Dynamic FD		Logistic regression						Adaptive rules
[46]	CFD,FD, ER								Data cleansing Platform
[41]		POS		Conditional random field					POS tagger for Twitter
[42]		POS		Markov model	Algorithm				POS tagger for online conversational text

Big data quality frameworks proposed in [12, 48, 49] can be used to address the dimensions of data quality within big data. In particular, researchers [12] proposed a big data preprocessing system that aims to manage data quality through rules which can be user defined, auto-discovery, or domain related. Intrinsic and contextual quality dimensions are also classified in [12]. A big data processing architecture is presented by [49] that manages the quality of social media data by utilizing domain policy data rules and metadata creation to support provenance. Finally, in [48], a big data quality assessment framework whose hierarchical structure of data quality rules is composed of data quality dimensions and indexes is provided.

7 Conclusion

There are many data analyses that acknowledged the difficulties faced in dealing with either imperfect data or algorithmic limitations due to large-scale data. Data cleansing and feature engineering can play a vital role into eliminating these issues and improving the generalization of predictive models. Considering these aspects, we define noise in the source of social networks and explore new practices toward data quality and preprocessing tasks oriented toward big data analytics. Since text is a rich source for data in OSNs, we present natural language processing techniques.

The main technical challenges apart from the long-standing issues of incomplete and noisy data come from the unique discourse structure and grammar of the content, the computational complexity of data-intensive preprocessing algorithms, the poor representation of complex data, and the high-dimensional space. Imbalanced datasets, data integration, and analysis of network structures are also challenging issues which are only meagerly discussed in this chapter.

The following list outlines future interesting topics for research and new avenues proposed for dealing with these challenges:

1. Distributed and richer data representation
2. Distributed data preprocessing
3. Development of NLP tools trained in the data scope of social networks
4. Modified algorithms for imperfect data
5. Real-time preprocessing and noise detection
6. Active learning in crowdsourcing for label provision or validation of algorithmic output

The broader objective of this work is to set a beginning of developing a framework toward social networking data preprocessing in order to reach higher analysis insights and to alleviate the issue of developing automated tools for streamlining big data analytics. This chapter can be found helpful to any data scientist that conducts any type of data analysis, to newcomers that would like to obtain a panoramic view of the preprocessing task in big data sources, to researchers that will use social networks as a data source, and to researchers that are familiar with certain issues and algorithms and would like to enhance them.

References

1. Saha, B., & Srivastava, D. (2014). *Data quality: The other face of Big Data*. In *2014 IEEE 30th international conference on data engineering*, pp. 1294–1297.
2. Amin, A., et al. (2016). Comparing oversampling techniques to handle the class imbalance problem: A customer churn prediction case study. *IEEE Access, 4*, 7940–7957. https://doi.org/10.1109/ACCESS.2016.2619719.
3. Jagadish, H. V., Gehrke, J., Labrinidis, A., Papakonstantinou, Y., Patel, J. M., Ramakrishnan, R., & Shahabi, C. (Jul. 2014). Big data and its technical challenges. *Communications of the ACM, 57*(7), 86–94.
4. Sapountzi, A., & Psannis, K. E. (2016). Social networking data analysis tools & challenges. *Future Generation Computer Systems*. https://doi.org/10.1016/j.future.2016.10.019.
5. Hassan, H., & Menezes, A. (2013). *Social text normalization using contextual graph random walks* (pp. 1577–1586). Sofia: Association for Computational Linguistics.
6. Peled, O., Fire, M., Rokach, L., & Elovici, Y. (2016). Matching entities across online social networks. *Neurocomputing, 210*, 91–106.
7. Huisman, M. (2014). Imputation of missing network data: Some simple procedures. In *Encyclopedia of social network analysis and mining* (pp. 707–715). New York: Springer New York.
8. Kossinets, G. (2006). Effects of missing data in social networks. *Social Networks, 28*(3), 247–268.
9. Kim, M., & Leskovec, J. (2011). The network completion problem: Inferring missing nodes and edges in networks. In *Proceedings of the 2011 SIAM International Conference on Data Mining* (pp. 47–58). Philadelphia: Society for Industrial and Applied Mathematics.
10. Hira, Z. M., & Gillies, D. F. (2015). A review of feature selection and feature extraction methods applied on microarray data. *Advances in Bioinformatics, 2015*, 198363.
11. Tan, W., Blake, M. B., Saleh, I., & Dustdar, S. (2013, September). Social-network-sourced big data analytics. *IEEE Internet Computing, 17*(5), 62–69.
12. Taleb, I., Dssouli, R., & Serhani, M. A. (2015). Big data pre-processing: A quality framework. *2015 IEEE International Congress on Big Data*, pp. 191–198.
13. Khayyat, Z., Ilyas, I. F., Jindal, A., Madden, S., Ouzzani, M., Papotti, P., Quiané-Ruiz, J.-A., Tang, N., & Yin, S. (2015). BigDansing. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data – SIGMOD'15*, pp. 1215–1230.
14. Chu, X., Ilyas, I. F., & Koutris, P. (2016). Distributed data deduplication. *Proceedings of the VLDB Endowment, 9*(11), 864–875.
15. Fan, W., & Wenfei. (December 2015). Data quality: From theory to practice. *ACM SIGMOD Record, 44*(3), 7–18.
16. Chu, X., Morcos, J., Ilyas, I. F., Ouzzani, M., Papotti, P., Tang, N., & Ye, Y. (2015). KATARA. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data – SIGMOD'15*, pp. 1247–1261.
17. Volkovs, M., Chiang, F., Szlichta, J., & Miller, R. J. (2014, March). Continuous data cleaning. In *2014 IEEE 30th International Conference on Data Engineering* (pp. 244–255). IEEE.
18. Chu, X., Ilyas, I. F., Krishnan, S., & Wang, J. (2016). Data cleaning: Overview and emerging challenges. In *SIGMOD'16 Proceedings of the 2016 International Conference on Management of Data*, pp. 2201–2206.
19. Zhou, D., Chen, L., & He, Y. (2015). An unsupervised framework of exploring events on twitter: Filtering, extraction and categorization. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*.
20. Sankaranarayanan, J., Samet, H., Teitler, B. E., Lieberman, M. D., & Sperling, J. (2009). TwitterStand. In *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems – GIS'09*, p. 42.
21. Ritter, A., Mausam, Etzioni, O., & Clark, S. (2012). Open domain event extraction from Twitter. In *Proceedings of the 18th ACM SIGKDD – KDD'12*, p. 1104.
22. Tang, N. (2014). *Big data cleaning* (pp. 13–24). Cham: Springer.

23. Cao, Y., Fan, W., & Yu, W. (2013). Determining the relative accuracy of attributes. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, pp. 565–576.
24. Fan, W., Wu, Y., & Xu, J. (2016). Functional dependencies for graphs. In *Proceedings of the 2016 International Conference on Management of Data – SIGMOD'16*, pp. 1843–1857.
25. Wang, P., Zhao, J., Huang, K., & Xu, B. (2014). *A unified semi-supervised framework for author disambiguation in academic social network* (pp. 1–16). Cham: Springer.
26. Abedjan, Z., Akcora, C. G., Ouzzani, M., Papotti, P., & Stonebraker, M. (Dec. 2015). Temporal rules discovery for web data cleaning. *Proceedings of the VLDB Endowment, 9*(4), 336–347.
27. Kaisler, S., Armour, F., Espinosa, J. A., & Money, W. (2013). Big data: Issues and challenges moving forward. In *2013 46th Hawaii International Conference on System Sciences*, pp. 995–1004.
28. Fan, J., Han, F., & Liu, H. (Jun. 2014). Challenges of big data analysis. *National Science Review, 1*(2), 293–314.
29. Gandomi, A., & Haider, M. (April 2015). Beyond the hype: Big data concepts, methods, and analytics. *International Journal of Information Management, 35*(2), 137–144.
30. Shi, W., Zhu, Y., Huang, T., Sheng, G., Lian, Y., Wang, G., & Chen, Y. (2016, March). An integrated data preprocessing framework based on apache spark for fault diagnosis of power grid equipment. *Journal of Signal Processing Systems, 86*, 1–16.
31. Poulos, J., & Valle, R. (2018). Missing data imputation for supervised learning. *Applied Artificial Intelligence, 32*(2), 186–196. https://doi.org/10.1080/08839514.2018.1448143.
32. Galar, M., Fernández, A., Barrenechea, E., Bustince, H., & Herrera, F. (2012). A review on ensembles for class imbalance problem: Bagging, boosting and hybrid-based approaches. *IEEE Transactions on Systems, Man, and Cybernetics – Part C: Applications and Reviews, 42*(4), 463–484.
33. Fire, M., Tenenboim-Chekina, L., Puzis, R., Lesser, O., Rokach, L., & Elovici, Y. (December 2013). Computationally efficient link prediction in a variety of social networks. *ACM Transactions on Intelligent Systems and Technology, 5*(1), 1–25.
34. Rennie, J., Shih, L., Teevan, J., & Karger, D. (2003) Tackling the poor assumptions of naive Bayes text classifiers. In *Proceedings of the ICLM-2003*.
35. Soley-Bori, M. (2013). *Dealing with missing data: Key assumptions and methods for applied analysis* (Vol. 4, pp. 1–19). Boston University.
36. Loh, P., & Wainwright, M. J. (2012). High-dimensional regression with noisy and missing data: Provable guarantees with non-convexity. *Annals of Statistics, 40*(3), 1637–1664.
37. Stekhoven, D. J., & Buhlmann, P. (2012). Missforest – Non-parametric missing value imputation for mixed-type data. *Bioinformatics, 28*(1), 112–118.
38. Mislove, A., Viswanath, B., Gummadi, K. P., & Druschel, P. (2010). You are who you know. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining – WSDM'10*, p. 251.
39. Kolb, L., Thor, A., & Rahm, E. (2012). Dedoop: Efficient deduplication with Hadoop. In *Proceedings of the VLDB endowment* (Vol. 5, p. 1878).
40. Singh, T., & Kumari, M. (2016). Role of text pre-processing in Twitter sentiment analysis. *Procedia Computer Science, 89*, 549–554.
41. Gimpel, K., Schneider, N., O'Connor, B., Das, D., Mills, D., Eisenstein, J., Heilman, M., Yogatama, D., Flanigan, J., & Smith, N. A. (2010). *Part-of-speech tagging for twitter: Annotation, features, and experiments*. Carnegie-Mellon Univ Pittsburgh Pa School of Computer Science.
42. Owoputi, O., Owoputi, O., Dyer, C., Gimpel, K., Schneider, N., & Smith, N. A. (2013). *Improved part-of-speech tagging for online conversational text with word clusters.* In *Proceedings of NAACL.*
43. Al-Hamami, M. A. H. (2015). The impact of big data on security. In *Handbook of research on threat detection and countermeasures in network security* (Vol. 3, pp. 276–298). Pennsylvania: IGI Global.

44. Nirmal, V. J., Amalarethinam, D. I. G., & Author, C. (2015). Parallel implementation of big data pre-processing algorithms for sentiment analysis of social networking data. *International Journal of Fuzzy Mathematical Archive, 6*(2), 149–159.
45. Bilgic, M., Licamele, L., Getoor, L., & Shneiderman, B. (2006). D-dupe: An interactive tool for entity resolution in social networks. In *2006 IEEE Symposium on Visual Analytics and Technology*, pp. 43–50.
46. Ebaid, A., Elmagarmid, A., Ilyas, I. F., Ouzzani, M., Quiane-Ruiz, J. A., Tang, N., & Yin, S. (2013). NADEEF: A generalized data cleaning system. *Proceedings of the VLDB Endowment, 6*(12), 1218–1221.
47. Geerts, F., Mecca, G., Papotti, p. & Santoro, D., 2014. That's all folks! LLUNATIC goes open source. *Proceedings of the VLDB Endowment, 7*(13), pp. 1565–1568.
48. Cai, L., & Zhu, Y. (2015). The challenges of data quality and data quality assessment in the big data era. *Data Science Journal, 14*(May), 2. https://dx.doi.org/10.5334/dsj-2015-002.
49. Immonen, A., Paakkonen, P., & Ovaska, E. (2015). Evaluating the quality of social media data in big data architecture. *IEEE Access, 3*, 2028–2043.
50. Domingos, P. (2012). A few useful things to know about machine learning. *Communications of the ACM, 55*(10), 78–87.