# Adaptive Learning Process for the Evolution of Ontology-Described Classification Model in Big Data Context

Rafael Peixoto[1,2], Christophe Cruz[2], Nuno Silva[1]
[1]GECAD-ISEP, Polytechnic of Porto, Porto, Portugal, {rafpp, nps}@isep.ipp.pt
[2] LE2I UMR6306, CNRS, Arts et Métiers, Univ. Bourgogne Franche-Comté, F-21000 Dijon, France,
christophe.cruz@u-bourgogne.fr

***Abstract*—One of the biggest challenges in Big Data is to exploit value from large volumes of variable and changing data. For this, one must focus on analyzing the data in these Big Data sources and classify the data items according to a domain model (e.g. an ontology). To automatically classify unstructured text documents according to an ontology, a hierarchical multi-label classification process called Semantic HMC was proposed. This process uses ontologies to describe the classification model. To prevent cold start and user overload, the classification process automatically learns the ontology-described classification model from a very large set of unstructured text documents. However, data is always being generated and its statistical properties can change over time. In order to learn in such environment, the classification processes must handle streams of non-stationary data to adapt the classification model. This paper proposes a new adaptive learning process to consistently adapt the ontology-described classification model according to a non-stationary stream of unstructured text data in Big Data context. The adaptive process is then instantiated for the specific case of of the previously proposed Semantic HMC.**

***Keywords—Maintenance; multi-label classification; adaptive learning, ontology; machine learning***

## I. INTRODUCTION

In the current Big Data era, retrieving valuable information for each consumer is mandatory to reduce the consumer information overload and help corporations to improve their efficiency. An increasing number of V's has been used to characterize Big Data [1], [2]: Volume, Velocity, Variety, Veracity and Value. Volume concerns the large amount of data that is generated and stored by social media, sensor data, etc.[1]. Velocity concerns both the production and the process to meet a demand because Big Data is not only a huge volume of data but it must be processed quickly as new data is generated over time. Variety relates to the various types of data composing the Big Data. These types include semi-structured and unstructured data representing 90% of its content [3] such as audio, video, web content, text, as well as traditional structured data. Veracity concerns the truthfulness in data. In traditional data warehouses there was always the assumption that the data is certain, clean, precise and complete but in Big Data context, namely the user-generated data can be uncertain, erroneous, imprecise and incomplete. The Value dimension measures how valuable the information is to a Big Data consumer.

Big Data analysis can be deemed as the analysis technique for a special kind of data. Therefore, many traditional data analysis methods used in Data Mining (algorithms for classification, clustering, regression, among others) are utilized for Big Data Analysis [1]. Hierarchical Multi-Label Classification (HMC) is a traditional Data Mining method that combines Multi-Label classification with Hierarchical classification [4]. In HMC, the items can be assigned to different hierarchical paths and simultaneously may belong to different class labels in the same hierarchical level [4]. The work previously proposed aims to exploit value by analysing Big Data using a Semantic Hierarchical Multi-Label Classification process (Semantic HMC) [5], [6].

Despite the previous Semantic HMC process classifies large volumes of unstructutred text documens, the system is unable to adapt to the changes occurring in Big Data. In fact, Big Data is not only a huge volume of data but is also a big velocity stream of data with big variety. The underlying process generating a data stream can be characterized as stationary or nonstationary. In a stationary stream it is assumed that the data distribution will not change over time, where in nonstationary stream the probabilistic properties of the data will change over time. The learning algorithms that assume a stationary data distribution, in general, produces a classifier that does not change over time, hence called a static classifier. In many real-world scenarios, however, the data is nonstationary (or evolving or drifting). In nonstationary data distribution, the performance of a non-adaptive classification model trained under the false stationary assumption may degrade the classification performance [7]. On the other hand, adaptive learning refers to updating classification models online during their operation regarding nonstationary data streams [8].

This paper proposes a new Adaptive Learning process to consistently adapt an Ontology-described classification model used for hierarchical multi-label classification regarding a non-stationary stream of unstructured text data in Big Data context. The process is instantiated for the specific problem of the Semantic HMC.

The rest of paper covers six sections. The second section describes the Semantic HMC process. The third section presents the background and the related work. The forth section proposes the new classification model adaptative



978-1-4673-8460-5/16/$31.00 ©2016 IEEE

process. The fifth section describes in detail the adaptative process instantiation for the Semantic HMC. The sixth section describes the ongoing evaluation. Finally, the last section draws conclusions and suggests further research.

## II. The Semantic HMC

learning process using scalable Machine Learning techniques and Rule-based reasoning. The process is unsupervised in the sense that no previously classified examples or rules to relate the data items with the labels exist.

The classification model is represented by an ontology (Abox+Tbox) that is automatically learned from huge volumes of data through highly scalable Machine Learning and Big Data technologies. The Semantic HMC proposes five individually scalable steps to reach the aims of Big Data analytics [5] (Fig. 1):

*1) **Indexation** extracts terms from data items and creates an index of data items.*

*2) **Vectorization** calculates the term-frequency vectors of the indexed items.*

*3) **Hierarchization** creates the label taxonomy (i.e. subsumption hierarchy) using term-frequency vectors.*

*4) **Resolution** creates the reasoning rules to relate data items with the labels based on term-frequency vectors.*

*5) **Realization** first populates the ontology with items and then for each item determines the most specific label and all its subsuming labels.*

The first four steps learn the ontology-described classification model used in the last step (realization) to classify the items (i.e. documents). The classification model is composed by the classification rules and all ontological concepts (i.e. label hierarchy).

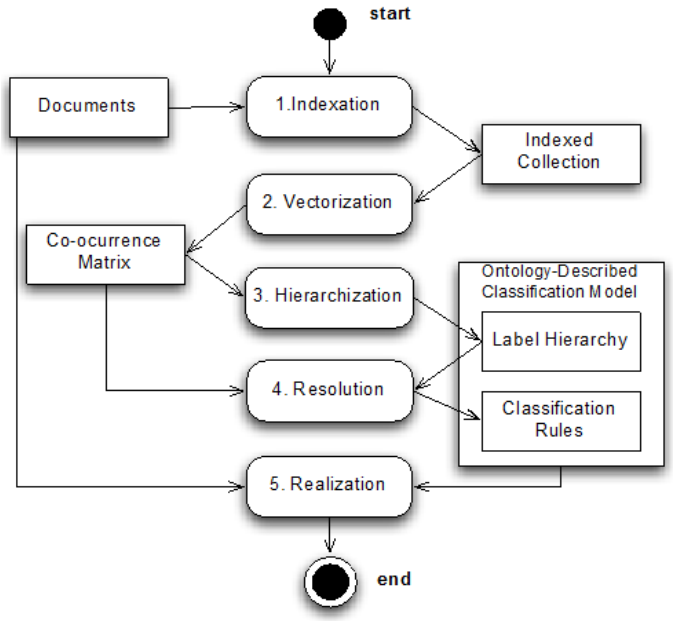


Fig. 1. Semantic HMC process

### B. Classification Model

The Semantic HMC's classification model is captured by a DL ontology with $\mathcal{ALCI}$ expressivity, as described in Table 1.

The $Item$ class represents data items to be labeled/classified and is populated with data items (e.g. text document, photos) in the assertional level (Abox). The $Term$ class defines the extracted terms from data items and is populated in the assertion level (Abox) with terms (e.g. words extracted from text documents, symbols representing subjects/objects in photos). The $Label$ class defines the terms that are considered to classify the items i.e. the most relevant terms. The $broader$ and $narrower$ relations define the subsumption hierarchy between labels. The relation's $hasAlpha$ and $hasBeta$ links $Label$ instances with $Term$ instances used to create the alpha and beta classification rules for that $Label$ as proposed in [9]. The $isClassified$ relation represents the categorization of an $Item$ . Predicting $isClassified$ relationships between item and label is the final goal of the prediction model application.

### C. Indexed Collection

The Index Collection (in the form of an inverted index) used by the Semantic HMC to learn the classification model is impacted by new items (text documents).

For each new text document $doc'$ in the stream, the collection of documents $C$ will evolve for a $C'$ impacting the Inverted Index. The inverted index is composed by a set of document vectors, one vector for each $term \in Term$, in the form: $vector_{term} < d_1, d_2, \dots, d_n >$ where $d \in Item$ . First terms $setTerms_{doc'}$ are extracted from the new text document $doc'$ using a term extraction approach proposed in [6] that includes spelling correction, stop-word and synonym detection and calculation of composed terms. The new terms are identified such as:

$$newTerms_{doc'} = \{term_i \notin Term : term_i \in doc'\} \quad (1)$$

Then the Inverted Index is updated with the new terms $newTerms_{doc'}$ identified in the new document $doc'$ and a new vector $vector_{term_x}$ is created for each $term_x \in newTerms_{doc'}$. Also all other vectors $vector_{term_y}$ where $term_y \in setTerms_{doc'} \wedge term_y \notin newTerms_{doc'}$ are updated with the new document $doc'$ such as: $vector_{term_y} < d_1, d_2, d_n, doc' >$.

TABLE I. Classification Model Concepts

| DL concepts | Description |
|---|---|
| $Item \sqsubseteq \exists hasTerm.Term$ | Items to classify (e.g. doc) has terms |
| $Term \sqsubseteq asString.String$ | Terms (e.g. word) extracted from items |
| $Label \sqsubseteq Term$ | Labels are terms used to classify items |
| $Label \sqsubseteq \forall broader.Label$ | Broader relation between labels |
| $Label \sqsubseteq \forall narrower.Label$ | Narrower relation between labels |
| $broader \equiv narrower^-$ | Broader and narrower are inverse |
| $Item \sqcap Term = \emptyset$ | Items and Terms are disjoint |
| $Label \sqsubseteq \forall hasAlpha.Term$ | Terms used to create Alpha rules |
| $Label \sqsubseteq \forall hasBeta.Term$ | Terms used to create Beta rules |
| $Item \sqsubseteq \forall isClassified.Label$ | Relation that links items with labels |



978-1-4673-8460-5/16/$31.00 ©2016 IEEE

### D. Co-occurrence Matrix

In stream context, the co-occurrence matrix $cfm$ is impacted with the inclusion of new text documents $doc'$ in the inverted index. The co-ocurrence values are impacted by the presence of the new document where new matrix lines and columns are added regarding the new terms $newTerms_{doc'}$.

The co-occurrence frequency matrix impact management is described in detail in [10] where for each pair of terms $(term_i, term_j) | \ term_i, term_j \in setTerms_{doc'}$ the co-occurrence matrix is impacted such as:

$$cfm_{C'}(term_i, term_j) = cfm_C(term_i, term_j) + 1 \quad (2)$$

### E. Labels Selection

To evaluate the term relevancy, the information retrieval method is used. The method exploits the item-frequency vectors to calculate for each $term_j$, the proportion $P_C(term_j)$ of items in a collection $C$ in which the $term_j$ appears:

$$P_C(term_j) = \frac{cfm(term_j, term_j)}{|C|} \quad (3)$$

where $|C|$ is the cardinality of the collection $C$.

Based on the terms that have a higher proportion $P_C(term_j)$ than a label threshold ($lT$) such that $P_C(term_j) \geq lT$ where $term_j \in Term$, the hierarchization process originates a set:

$$\omega_{lT} = \{term_j \in Term | P_C(term_j) \geq lT\} \quad (4)$$

The terms in this set are classified Label such that $Label \subseteq Term$. Further, a relation $label \subseteq Term \rightarrow Label$ and a function $term: Label \rightarrow Term$ are defined.

### F. Classification Rules Creation

The rules creation process uses thresholds as proposed in [9] to select the necessary and sufficient terms to classify a item with a label. Let $P_C(term_j | term_i)$ be the conditional proportion (number) of the items from collection $C$ common to $term_j$ and $term_i$, in respect to the number of items in $term_i$ such that:

$$P_C(term_j | term_i) = \frac{cfm\ (term_j, term_i)}{cfm(term_i, term_i)} \quad (5)$$

Two thresholds are defined:

- Alpha threshold (α) such that $\alpha < P_C(term_j | term_i)$, where $term_i \in Label$ and $term_j \in Term$. This threshold originates the Alpha relations between $term_i$ and $term_j$.
- Beta threshold (β) such that $\beta \leq P_C(term_j | term_i) \leq \alpha$, where $term_i \in Label$ and $term_j \in Term$. This threshold originates the Beta relations between $term_i$ and $term_j$.

## III. Background and Related Work

In this section, the current related work in adaptive learning and ontology evolution is presented and discussed. The first subsection describes some background and related work in adaptive learning regarding the data stream classification problem. The second subsection describes some background and related work in ontology evolution. The last subsection discusses the related work.

### A. Adaptive Learning

Data stream classification problem has been widely studied in the literature [11]–[18]. The underlying process generating a data stream can be characterized as stationary or nonstationary. In a stationary stream is assumed that the data distribution will not change over time. The learning algorithms that assume a stationary data distribution, in general, produces a classifier that does not change over time, hence called a static classifier.

In many real-world scenarios, however, the data is nonstationary (also referred as evolving or drifting) where the probabilistic properties of the data change over time. These changes in the data distribution can induce more or less radical changes in the target concept, which is generally known in literature as concept drift [19]. In nonstationary data distribution, the performance of a non-adaptive classification model trained under the false stationary assumption may degrade the classification performance [7].

Two of the most challenging and well-studied characteristics of data streams are its infinite-lengthand concept-drift [8]. Most of the existing data stream classification techniques are designed to handle these two aspects of the classification process [11]–[18]. The most used approach to handle infinite-length and concept-drift problems is incremental learning, either classified as:

- Single-model incremental approach, where the classification model is adapted regarding new data [15], [16].
- Hybrid batch-incremental approach, in which the model is re-built using a batch learning technique and older models are replaced by newer models [17], [18]

In addition to infinite-length and concept-drift characteristics, some have studied also concept-evolution and feature-evolution [11]–[14].

Concept Evolution occurs when new classes (labels in Semantic HMC) used to classify the items emerge in the underlying stream of text items [11], [12]. For example, Spinosa et al. [12] apply a cluster-based technique to detect novel classes in data streams. The approach build a normal model that is a static model representing the initial model and remaining as a reference to the initial learning process. The normal model is composed of hyperspheres and is continuously updated with stream progression. If any cluster is formed outside this hypersphere, which satisfies a certain density constraint, then a novel class is declared. However this approach focuses in only one class and it is not directly applicable to multiclass data stream classification.

In the literature, the feature space is a vocabulary, usually composed of a high number of words, used to describe data items. In data stream, new words that can better describe the items must be introduced in the feature space. Feature-evolution occurs when new words (features) emerge from the data stream and old features fade away [11], [13], [14]. For example, Katakis et al. [13] propose a feature-evolution technique for text streaming where a dynamic feature space is



978-1-4673-8460-5/16/$31.00 ©2016 IEEE

adapted by an incremental feature selection. In the proposed technique, when a new text document arrives it is checked whether there is any new word in the document. If there is a new word, it is added to the vocabulary and then the statistics are updated. The experimental results showed that the proposed approach offers better predictive accuracy compared to classical incremental learning. In Big-Data context, MapReduce is not suitable to express streaming algorithms and traditional sequential online algorithms are limited by the memory and bandwidth of a single machine. For running data mining and machine learning algorithms on a distributed stream processing engine, A. Biffet [20] has proposed SAMOA (Scalable Advanced Massive Online Analysis), an open-source platform for mining big data streams. It provides a collection of distributed streaming algorithms for the most common data mining and machine learning tasks such as classification, clustering, and regression. It uses distributed stream processing engines (DSPEs) to express parallel computation on streams, and combine the scalability of distributed processing with the efficiency of streaming algorithms.

### B. Ontology Evolution

Ontologies [21] are one of the most accepted way to represent semantic information in the Semantic Web [22], seen as an extension of the World Wide Web, where semantic information can be understood by machines. On the other hand, ontologies are also a good solution for intelligent computer systems that operate close to the human concept level bridging the gap between the human requirements and the computational requirements [23].

Like other software models notably database schemas, ontologies inevitably change over time. An ontological change is an action performed over an ontology whose result is an ontology different from the original version [24]. Nowadays ontologies are increasingly used in mainstream applications that change ontologies during its lifetime. These applications requires a new research focus in ontology engineering that supports the complete lifecycle of ontologies, beyond the initial steps of acquisition, development and deployment [25]. An ontological change is an action performed over an ontology whose result is an ontology different from the original version [26]. Based on the most common uses of each term in the literature, Flouris et al. [27] identify and study ten subfields of ontology change. Several ontology evolution processes are proposed in literature [25], [26], [28], [29]. One of the most remarkable is proposed by Stojanovic et al. [28] that proposes an ontology evolution process composed by a six phase cyclic process: (1) Capturing changes to be applied to the ontology; (2) Representation process provides a way to represent the changes in a specific model called “evolution ontology”; (3) Semantics of Change process addresses syntactic and semantic inconsistencies that could arise as a result of the changes; (4) Implementation process coupled with user interaction for approving or cancelling changes; (5) Propagation process allows the update of outdated instances and reflecting the impact in referenced ontologies; (6) Validation checks that the performed changes led to the desirable result allowing the user to accept or reject changes.

Current work done by Liu et al. [30] is focused in the phase of change representation, more specifically, modeling the elementary changes via a novel mathematical calculus named SetPi. Migrating ontologies to their last versions can generate gaps that must be re-examined and others that no re-examination is justified. In [31], Tzitzikas et al. formalize and propose principles, techniques and tools for managing uncertainly incurred by such migrations specifically for: Identifying automatically the descriptions that are candidates for specialization, Recommending possible specializations and Updating the available descriptions (and their candidate specializations), after the user (curator of the repository) accepts/rejects such recommendations. More recently, Pittet et al. [29] propose a change management methodology for managing the ontology life-cycle called OntoVersionGraph. The methodology includes ontology evolution and versioning features, in conjunction with contextual view modeling.

### C. Discussion

Diverse applications of adaptive learning exist in literatue, including adapt classification models regarding streams of data. Several studies where proposed to detect feature-evolution, concept-evolution and concept-drift. Also efforts to adapt data mining and machine learning algorithms on distributed stream processing engines were recently published. However current adaptive learning processes don’t consider the possibility of using ontologies to describe the learned knowledge in Big Data context. On the other hand, processes for ontology evolution and change management are adequate for general ontologies and consistently handle changes to the ontology. However, these general ontology evolution processes do not capture the specific semantics of a classification model like the one used in Semantic HMC.

To the extent of our knowledge, our approach is the first to propose an adaptive learning process that consistently and semantically evolves an ontology-described classification model used for hierarchical multi-label classification regarding a non-stationary stream of unstructured text data in Big Data context.

## IV. Semantic Hmc Adaptive Learning

This section proposes a new Adaptive Learning process to consistently adapt an Ontology-described classification model used for hierarchical multi-label classification. The adaptive learning process focus in adapting the ontology-described classification model regarding a stream of unstructured text documents in Big Data context.

Three characteristics of data stream processing are addressed in the proposed adaptive learning process:

- Feature-evolution that occurs when new terms/words (features) emerge from the data stream and old features fade away;
- Concept-evolution occurs when new classes/labels used to classify items emerge in the underlying stream of text items;
- Concept-drift where the probabilistic properties of the data change over time and induce more or less radical changes in the target concept/label (i.e. changes in the Alpha and Beta sets and in the hierarchical relations).



978-1-4673-8460-5/16/$31.00 ©2016 IEEE

The adaptive learning process manages the impact of changes in the classification model according to the specific semantic of the classification process.

### *A. Adaptive Process Ontology*

The adaptive learning process is described by an ontology presented in Fig. 2. Seven main concepts compose the ontology:

- Input: Input for the adaptive learning process. E.g. the Data Stream and the already applied modifications as inputs.

Change Sensor: Adaptive learning component that detect changes in the inputs. In example, the Data Stream sensor that reads the data stream to detect changes on their data distribution properties and the Impact Change sensor that detects changes performed to the classification model.

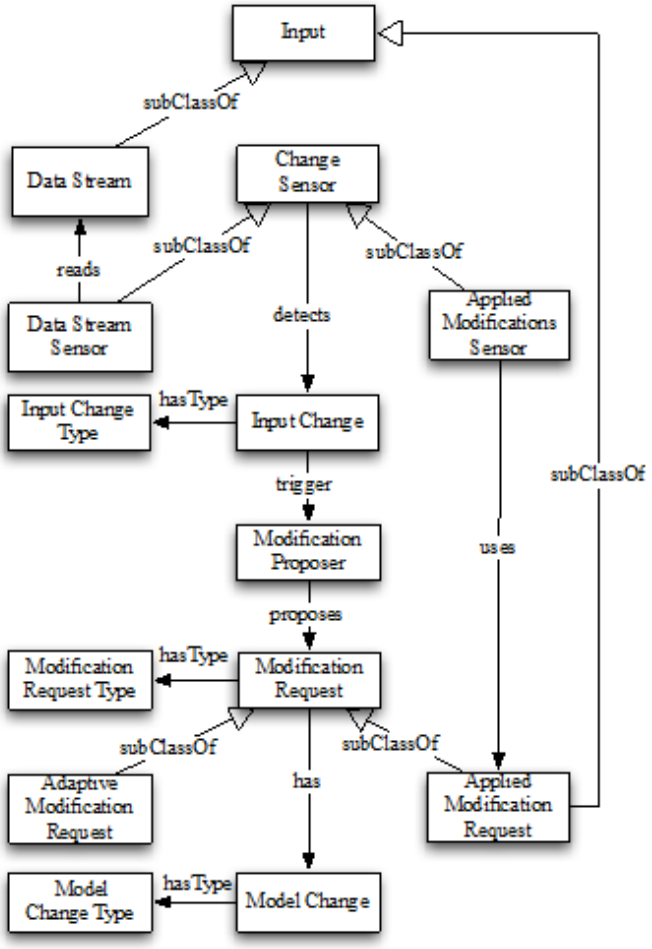


Fig. 2. Adaptive learning process ontology

- Input Change: Change detected in an Input by a change sensor. The input change is described by a type of change that is detected.
- Modification Proposer: Adaptive Learning component that proposes Modifications to adapt the classification model according the Input Changes.
- Modification Request: Modification proposed by a Modification Proposer to adapt the classification model. The Modification Request is described by a Modification Request Type and composed by changes to the classification model that together have specific meaning.
- Model Change: Ontological changes performed to the ontology-described classification model. The changes are described by a Model Change Type.

The semantics of the maintenance process is captured in the Modification Request Type and Model Change Type classes. I.e. the proposed generic maintenance process is driven/constrained by the types of modification request and model change defined in the adaptive process ontology, hence supporting the configurability of the process.

### *B. Classification Model Adaptive Process*

The classification model adaptive process is the set of activities necessary to maintain the classification model based on the data stream, but independent of any particularities of the modification and model changes types. The process uses a Single-model incremental approach where the classification model is incrementally adapted regarding new data.

Four main steps compose the adaptive process (Fig. 3): (1) Input Change Detection, (2) Modification Request Derivation, (3) Model changes Derivation and (4) classification model Evolution. The classification model Evolution step is composed by four sub-steps: (4.1) Apply Changes, (4.2) Consistency Check, (4.3) Resolve Inconsistencies and (4.4) Modification Commit.

#### *1) Input Change Detection*

This step detects the Input Changes according to (i) the Input Data Stre0am and (ii) the previously applied modifications (i.e. these may cause new modifications requests).

#### *2) Modification Request Derivation*

This step derives the modifications requests of the classification model, according the detected Input Changes. Each Modification Request Type defines the necessary and sufficient conditions (i.e. input changes that must be present) for its instantiation.

#### *3) Model Changes Derivation*

During this step the process analyzes the proposed Modification Requests and derives the Model Changes necessary to apply the modification request. The type of the modification request determines the model changes. Model changes are either additions or deletions axioms to the ontology-described classification model, corresponding to Feature Evolution (i.e. Term evolution), Concept Evolution (i.e. Label evolution) or Concept-Drift (i.e. Hierarchy and rules evolution).

#### *4) Classification Model Evolution*

During this step the modification requests (i.e. the set of model changes derived in step 3) are applied to to classification model (4.1 Apply Changes). Once applied, a consistency check is executed (4.2 Consistency Check). If inconsistencies are


978-1-4673-8460-5/16/$31.00 ©2016 IEEE

observed, they are reviewed by an external actor (i.e. a supervisor) and changes are deferred until inconsistency are resolved (4.3 Resolve Inconsistencies). Otherwise the changes are commited (4.4 Modification Commit). This step corresponds to the ontology evolution approach proposed in the literature (e.g. [28],[26]).

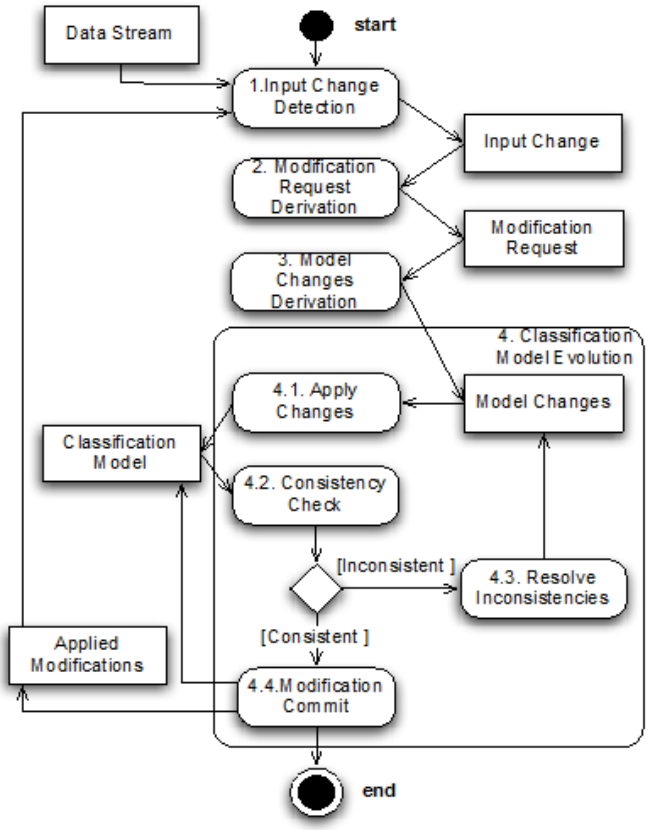


Fig. 3. Adaptive Process of the ontology-described classification model

## V. Semantic HMC Adaptive Process Instantiation

The conceptual classification model adaptive process presented in the previous section is adopted/adapted to the adapt the classification model of the Semantic HMC process described in section II.A.

### A. Input Change Detection

Regarding the Specific semantic of the Semantic HMC (i.e. Inverted Index, Co-occurrence Matrix), four Input Change Types are identified (Table 2).

Let $iChange(ChangeType, parameters)$ be the input change composed by a change type and a list of relevant parameters. To detect the Input Changes, four Input Change Sensors are proposed to detect changes in each input: New

TABLE II. Input Change Types

| Input Change Type | Description |
|---|---|
| newDocument | New document is added to the collection from the Data Stream increasing the Collection. |
| newTerm | New term is deteced in the Data Stream |
| coOcurrence | Document frequency of a term has changed |
| appliedModification | Modification Request applied with success to the classification model. |

Term Sensor, Document Sensor, Co-ocurrence Matrix Frequency Sensor and the Applied Modifications Sensor.

*1) Document Sensor*

The Document Sensor detects when a new document from the Data Stream $doc'$ is added to the collection C increasing their cardinality from $|C|$ to $|C'|$ such as: $|C'| = |C| + 1$. Hence an Input Change $iChange$ of the type *CardinalityChange* is detected i.e. $iChange(newDocument, \text{doc})$.

*2) New Term Sensor*

The New Term Sensor detects new terms in the Data Stream using the new terms $newTerms_{doc'}$ identified in each new document $doc'$ to update the Inverted Index structure. Hence a Input Change $iChange$ of the type *NewTerm* is detected for each $term_d \in newTerms_{doc'}$, i.e. $iChange(newTerm, term_d)$.

*3) Co-ocurrence Matrix Frequency Sensor*

The Co-ocurrence Matrix Frequency Sensor detects changes in the co-ocurrence frequency values. The change is detected when the Co-ocurrence Matrix $cfm_C$ is impacted by the Data Stream as described in equation (2). For each impact $cfm_{C'}(term_i, term_j)$ an Input Change $iChange$ of the type $coOcurrence$ is detected i.e. $iChange(coOcurrence, \{term_i, term_j\})$.

*4) Applied Modifications Sensor*

The Applied Modifications Sensor detects new Applied Modification Requests. For each new Applied Modification, an Input Change $iChange$ of the type $appliedModification$ is detected i.e. $iChange(appliedModification, mod)$.

### B. Modification Request Derivation

A modification request captures the high-level semantics of the modifications to occur in the classification model (Table 3). Each modification request type defines the necessary and sufficient conditions for its instantiation:

- AddTerm: An $newTerm$ input change is detected.
- AddLabel: An $iChange$ is detected that either:
  - $iChange(appliedModification, mod)$ such that $mod \in AddTerm$.
  - $iChange(newDocument, doc)$ and $P_c(term_i) \geq lT$ where $(i)$ $P_c(term_i)$ is the proportion of items in a collection $C$ in which the $term_i$ appears, (ii) $lT$ is the label threshold, (iii) $term_i \in doc$ and (iv) $term_i \notin Label$.
- DeleteLabel: An $iChange$ is detected that either:
  - $iChange(newDocument, doc)$ and $P_c(term_i) < lT$ where (i) $P_c(term_i)$ is the proportion of items in a collection $C$ in which the $term_i \in doc$ appears, (ii) $lT$ is the label threshold and (iii) $term_i \in Label$.
- AddHRelation: An $iChange$ is detected that either:



978-1-4673-8460-5/16/$31.00 ©2016 IEEE

— $iChange(appliedModification, mod)$ such that $mod \in AddLabel$.
— $iChange(coOcurrence, \{term_i, term_j\})$ and by apply the Hierarchichal Relation Change Detection approach described in the section 3.3 of [10] for $term_i$and $term_j$ is detected an *"Add"* change that represents the creation of a Hierarchical Relation between $term_i \in Label$ and $term_j \in Label$.

TABLE III. MODIFICATION REQUEST TYPES

| Modification Request Type | Description |
|---|---|
| AddTerm | Add a new term to the classification model |
| AddLabel | Add a new Label to the classification model |
| DeleteLabel | Delete a Label from the classification model |
| AddHRelation | Add a Hierarchichal Relation between two Labels to the classification model |
| DeleteHRelation | Delete a Hierarchichal Relation between two Labels from the classification model |
| AddAlphaTerm | Add a Alpha Relation between a Term and a Label to the Classificaiton Model |
| DeleteAlphaTerm | Delete a Alpha Relation between a Term and a Label from the Classificaiton Model |
| AddBetaTerm | Add a Beta Relation between a Term and a Label to the Classificaiton Model |
| DeleteBetaTerm | Delete a Beta Relation between a Term and a Label from the Classificaiton Model |

- DeleteHRelation: An $iChange$ is detected that either:

— $iChange(appliedModification, mod)$ such that $mod \in DeleteLabel$.
— $iChange(coOcurrence, \{term_i, term_j\})$ and by apply the Hierarchichal Relation Change Detection approach described in the section 3.3 of [10] for $term_i$and $term_j$ is detected a *"Delete"* change that represents the deletion of a Hierarchical Relation between $term_i \in Label$ and $term_j \in Label$.

- AddAlphaTerm: An $iChange$ is detected that either:

— $iChange(appliedModification, mod)$ such that $mod \in AddLabel$ and $\exists term_j \in Term: P_{C'}(term_j|term_i) > \alpha$ where (i) $\alpha$ is the Alpha threshold and (ii) $term_i \in mod$.
— $iChange(coOcurrence, \{term_i, term_j\})$ and $P_{C'}(term_j|term_i) > \alpha$ where $term_j \in Term$ is not an Alpha term of $term_i \in Label$.

- DeleteAlphaTerm: An $iChange$ is detected that either:

— $iChange(appliedModification, mod)$ such that $mod \in DeleteLabel$ and exist a $term_j \in Term$ that is an Alpha term of the $term_i \in mod$.
— $iChange(coOcurrence, \{term_i, term_j\})$ and $P_{C'}(term_j|term_i) \leq \alpha$ where $term_j \in Term$ is an Alpha term of $term_i \in Label$.

- AddBetaTerm: An $iChange$ is detected that either:

— $iChange(appliedModification, mod)$ such that $mod \in AddLabel$ and $\exists term_j \in Term: \beta \leq P_{C'}(term_j|term_i) \leq \alpha$ where (i) $\beta$ is the Beta threshold and (ii) $term_i \in mod$.
— $iChange(coOcurrence, \{term_i, term_j\})$ and $\beta \leq P_{C'}(term_j|term_i) \leq \alpha$ where $term_j \in Term$ is not a Beta term of $term_i \in Label$.

— DeleteBetaTerm: An $iChange$ is detected that either:
— $iChange(appliedModification, mod)$ such that $mod \in DeleteLabel$ and exist a $term_j \in Term$ that is a Beta term of the $term_i \in mod$.
— $iChange(coOcurrence, \{term_i, term_j\})$ and $P_{C'}(term_j|term_i) < \beta$ or $P_{C'}(term_j|term_i) > \alpha$ where $term_j \in Term$ is a Beta term of $term_i \in Label$.

### C. Model Changes Derivation

Based on the derived modification requests, this step generates the classification model changes, i.e. a set of ontology-evolution changes that once applied correspond to the modification.

According to Pittet et al. [29], there are four types of basic Description Logics $\mathcal{SHOIN}(\mathcal{D})$ ontology changes (Table 4). The first column of Table 4 gives the Description Logics syntax for the construction of axioms and facts of a knowledge base in and the second column gives the basic changes description in an abstract syntax: classes (C), data types (T), relations (R), attributes (A), instances (I) and datavalues (V). Each change has an associated operation parameter $Operation = \{Add, Delete\}$ where Delete is the inverse operation of Addition (Add).

Therefore, each modification type defines the classification model changes to apply:

- AddTerm: Considering the $termString$ as the word (string) that originates the new term to include in the classification model, the model changes to apply are:

— $Instance(Add, term)$
— $InstancesOf(Add, term, Term)$, i.e. $term \in Term$
— $InstanceOfDatatypeProperty(Add, term, "termString", asString)$

- AddLabel: Considering that $term \in Term$ originates the Label to include in the classification model, the model change to apply is:

— $InstancesOf(Add, term, Label)$

TABLE IV. MODEL CHANGES BY TYPE

| DL Syntax | Basic changes abstract syntax |
|---|---|
| $I$ | Instance(Operation, Instance) |
| $I \in Ci$ | InstancesOf(Operation, Instance, Class) |
| $\{I1, I2\} \in Ri$ | InstancesOfObjectProperty(Operation, Instance1, Instance2, ObjectProperty) |
| $\{I, Vi\} \in Ai$ | InstanceOfDatatypeProperty(Operation, Instance, Datavalue, DatatypeProperty) |

- DeleteLabel: Considering that $term \in Term$ originates the Label to delete from the classification model, the model change to apply is:

— $InstancesOf(Delete, term, Label)$


978-1-4673-8460-5/16/$31.00 ©2016 IEEE

- AddHRelation: Considering $label1 \in Label$ as the broader label and $label2 \in Label$ as the narrower label to include in the classification model, the model changes to apply are either:

– $InstancesOfObjectProperty(Add, label1, label2, broader)$
– $InstancesOfObjectProperty(Add, label2, label1, narrower)$

- DeleteHRelation: Considering $label1 \in Label$ as the broader label and $label2 \in Label$ as the narrower label to delete from the classification model, the model changes to apply are:

– $InstancesOfObjectProperty(Delete, label1, label2, broader)$
– $InstancesOfObjectProperty\ (Delete, label2, label1, narrower)$

- AddAlphaTerm: Considering $term \in Term$ as the relevant term to be Alpha term of the label $label \in Label$, the model change to apply is:

– $InstancesOfObjectProperty(Add, label, term, hasAlpha)$

- DeleteAlphaTerm: Considering $term \in Term$ as the term that is no longer relevant to be Alpha term of the label $label \in Label$, the model change to apply is:

– $InstancesOfObjectProperty(Delete, label, term, hasAlpha)$

- AddBetaTerm: Considering $term \in Term$ as the relevant term to be classified as Beta term of the label $label \in Label$, the model change to apply is:

– $InstancesOfObjectProperty(Add, label, term, hasBeta)$

- DeleteBetaTerm: Considering $term \in Term$ as the term that is no longer relevant to be Beta term of the label $label \in Label$, the model change to apply is:

– $InstancesOfObjectProperty(Delete, label, term, hasBeta)$

### *D. Classification Model Evolution*

Because the classification model is described by an ontology, the evolution process will adopt/adapt a state of the art ontology evolution approach. Several approaches are proposed in the literature to evolve ontologies (e.g. [28][26][29]). Based in the elementary evolution phases proposed in [28], four conceptual sub-steps are proposed: Apply Changes, Consistency Check, Resolve Inconsistencies and Modification Commit.

#### *1) Apply Changes*

The model changes derived in step 3 (Model Changes Derivation) are applied to the classification model. Instead of applying the model changes to the classification model in use by the Semantic HMC, they are applied to a copy of it. This avoids turning the classification model incompatible or inaccessible to the Semantic HMC before a final version is committed.

#### *2) Consistency Check*

Once the model changes are applied to the classification model, a consistency check is performed by an ontology reasonner. If inconsistencies are detected, a new change modeling phase takes place.

#### *3) Resolve Inconsistencies*

If inconsistencies are detected, new additional changes to the classification model are proposed to resolve the inconsistencies. The proposal of new additional changes is a human-based process.

#### *4) Modification Commit*

If the adapted classification model is consistent, then the changes are committed and available to be used by the Semantic HMC. Also, the modification request is classified as applied modification and exploited in step 1 (i.e. Input Change Detection).

## VI. ONGOING EVALUATION

The evaluation of the adaptive classification model process for the Semantic HMC is performed by comparing the classification performance at time $t_i$ of (i) a static classification model created at time $t_1$ and (ii) a classification model created at time $t_1$ but adapted by the adaptive process to $t_i$. The predictive performance of the classification model is evaluated using Precision, Recall and F1 metrics upon the adaptation (i.e. evolution) of three dimensions of the classification model.

### *A. Feature-Evolution Evaluation*

The Feature-evolution performance is evaluated by comparing (i) the classification model adapted with a dynamic feature space and (ii) the same classification model with a static feature space for $t_i$, regarding:

- The number of terms used to describe the text items.
- The accuracy of the classification model.

### *B. Concept-Evolution Evaluation*

The concept-evolution performance is evaluated by comparing (i) the classification model adapted with a dynamic set of labels and (ii) the same classification model with a static set of label for $t_i$, regarding:

- The number of labels used to classify the items.
- The accuracy of the classification model.

### *C. Concept-drift Adaptation Evaluation*

The concept-drift performance is evaluated by comparing (i) the classification model adapted with a dynamic rule set and a dynamic hierarchical relations and (ii) the same model with a static rule sets and hierarchical relations for $t_i$, regarding:

- The number of hierarchical relations between the labels.
- The number of rules used to relate the items and the labels.
- The accuracy of the classification model.



978-1-4673-8460-5/16/$31.00 ©2016 IEEE

## VII. CONCLUSIONS

To adapt the Semantic HMC to Big Data context, this paper proposes a new adaptive process for the ontology-described classification model according to non-stationary data streams, regarding feature evolution, concept-evolution and concept-drift.

To the extent of our knowledge, our approach is the first to propose an adaptive learning process that consistently evolves an ontology-described classification model used for hierarchical multi-label classification regarding a non-stationary stream of unstructured text data in Big Data context.

Four steps comprise the proposed conceptual maintenance process: (1) Input Change Detection, (2) Modification Request Derivation, (3) Model Changes Derivation and (4) classification model Evolution. This process suggests capturing the semantics of the process in two core entities:

- Input Sensors, that derive the input changes (either data stream or previously applied modifications) into input changes.
- Modification Request Types, that define:

– the necessary and sufficient conditions to derive each modification request according to the input changes;
– the (ontology) model changes necessary to carry out the derived modification request.

Hence, the conceptual process is adaptable to different type of ontology-described classification models used in Big Data classification processes (e.g. either considering words, images, sounds, GPS coordinates).

This conceptual process was then sucessfully adopted in the adaptation of a Semantic HMC classification model through the definition/configuration of the Input Sensors and Modification Request Types.

Our current research efforts focus in the evaluation of the adaptive classification model process, for which the team already defined the main inputs, constraints and metrics. As further work we aim to extend the maintenance process beyond the adaptation to a stream of data, by supporting semantic expressive and controlled corrections from external actors.

## ACKNOWLEDGMENT

This project is founded by the company Actualis SARL, the French agency ANRT and through the Portuguese COMPETE Program under the project AAL4ALL (QREN13852).

## REFERENCES

[1] M. Chen, S. Mao, and Y. Liu, “Big data: A survey,” Mob. Networks Appl., vol. 19, no. 2, pp. 171–209, Jan. 2014.

[2] P. Hitzler and K. Janowicz, “Linked Data, Big Data, and the 4th Paradigm,” Semant. Web, vol. 4, no. 3, pp. 333–335, 2013.

[3] A. R. Syed, K. Gillela, and C. Venugopal, “The Future Revolution on Big Data,” Int. J. Adv. Res. Comput. Commun. Eng., vol. 2, no. 6, pp. 2446–2451, 2013.

[4] W. Bi and J. Kwok, “Multi-label classification on tree-and DAG-structured hierarchies,” Yeast, pp. 1–8, 2011.

[5] T. Hassan, R. Peixoto, C. Cruz, A. Bertaux, and N. Silva, “Semantic HMC for big data analysis, ” in The IEEE Big Data 2014, pp. 26–28.

[6] R. Peixoto, T. Hassan, C. Cruz, A. Bertaux, and N. Silva, “Semantic HMC: A Predictive Model using Multi-Label Classification For Big Data,” in The 9th IEEE BigDataSE, 2015.

[7] A. Tsymbal, “The problem of concept drift : definitions and related work,” Computer Science Department, Trinity College Dublin, 2004.

[8] J. Gama, A. Bifet, M. Pechenizkiy, and A. Bouchachia, “A Survey on Concept Drift Adaptation,” C. Comput. Surv., vol. 46, no. 4, 2014.

[9] D. Werner, N. Silva, C. Cruz, and A. Bertaux, “Using DL-reasoner for hierarchical multilabel classification applied to economical e-news,” in Science and Information Conference, 2014, pp. 313–320.

[10] R. Peixoto, C. Cruz, and N. Silva, “Semantic HMC: Ontology-Described Hierarchy Maintenance in Big Data Context,” in On the Move to Meaningful Internet Systems: OTM 2015 Workshops, 2015, pp. 492–501.

[11] M. M. Masud, C. Qing, L. Khan, C. Aggarwal, G. Jing, H. Jiawei, and B. Thuraisingham, Addressing concept-evolution in concept-drifting data streams. 2010, pp. 929–934.

[12] E. J. Spinosa, A. P. D. L. F. De Carvalho, and J. Gama, “Cluster-based novel concept detection in data streams applied to intrusion detection in computer networks,” in Proceedings of the 2008 ACM symposium on Applied computing, 2008, pp. 976–980.

[13] I. Katakis, G. Tsoumakas, and I. Vlahavas, “Dynamic feature space and incremental feature selection for the classification of textual data streams,” Pkdd, pp. 102–116, 2006.

[14] B. Wenerstrom and C. Giraud-Carrier, “Temporal Data Mining in Dynamic Feature Spaces,” Data Mining, 2006. ICDM '06. Sixth Int. Conf., pp. 1141–1145, 2006.

[15] J. Wang, “A framework for on-demand classification of evolving data streams,” IEEE Trans. Knowl. Data Eng., vol. 18, pp. 577–589, 2006.

[16] G. Hulten, L. Spencer, and P. Domingos, “Mining time-changing data streams,” Proc. seventh ACM SIGKDD Int. Conf. Knowl. Discov. data Min. - KDD ’01, pp. 97–106, 2001.

[17] J. Gao, W. Fan, and J. Han, “On appropriate assumptions to mine data streams: Analysis and practice,” Proc. - IEEE Int. Conf. Data Mining, ICDM, pp. 143–152, 2007.

[18] H. Wang, W. Fan, P. S. Yu, and J. Han, “Mining concept-drifting data streams using ensemble classifiers,” Ninth ACM SIGKDD Int. Conf. Knowl. Discov. Data Min., vol. 2, no. 1, pp. 226–235, 2003.

[19] G. Widmer and M. Kubat, “Learning in the presence of concept drift and hidden contexts,” Mach. Learn., vol. 23, no. 1, pp. 69–101, 1996.

[20] A. Bifet and G. D. F. Morales, “Big Data Stream Learning with SAMOA,” 2014 IEEE Int. Conf. Data Min. Work., pp. 1199–1202, 2014.

[21] T. R. Gruber, “A Translation Approach to Portable Ontology Specifications,” Knowl. Creat. Diffus. Util., vol. 5, pp. 199–220, 1993.

[22] T. Berners-Lee, J. Hendler, and O. Lassila, “The Semantic Web,” Sci. Am., vol. 284, no. 5, pp. 34–43, 2001.

[23] L. Obrst, “Ontologies for semantically interoperable systems,” in Proceedings of the twelfth international conference on Information and knowledge management - CIKM ’03, 2003, pp. 366–369.

[24] N. F. Noy and M. Klein, “Ontology Evolution: Not the Same as Schema Evolution,” Knowl. Inf. Syst., vol. 6, no. 4, pp. 428–440, Mar. 2004.

[25] F. Zablith, G. Antoniou, M. D’Aquin, G. Flouris, H. Kondylakis, E. Motta, D. Plexousakis, and M. Sabou, “Ontology evolution: a process-centric survey,” Knowl. Eng. Rev., pp. 1–31, 2013.

[26] M. Klein and N. F. Noy, “A component-based framework for ontology evolution,” in Proceedings of the IJCAI, 2003, vol. 3.

[27] G. Flouris, D. Manakanatas, H. Kondylakis, D. Plexousakis, and G. Antoniou, “Ontology change: classification and survey.” 2007.

[28] L. Stojanovic, A. Maedche, B. Motik, and N. Stojanovic, “User-driven ontology evolution management,” in Knowledge Engineering and Knowledge Management, 2002, vol. 2473, pp. 133–140.

[29] P. Pittet, C. Cruz, and C. Nicolle, “An ontology change management approach for facility management,” Comput. Ind., vol. 65, no. 9, pp. 1301–1315, 2014.

[30] L. Liu, P. Zhang, R. Fan, R. Zhang, and H. Yang, “Modeling ontology evolution with SetPi,” Inf. Sci. (Ny)., vol. 255, pp. 155–169, Jan. 2014.

[31] Y. Tzitzikas, M. Kampouraki, and A. Analyti, "Curating the Specificity of Ontological Descriptions under Ontology Evolution," J. Data Semant., Jun. 2013.



978-1-4673-8460-5/16/$31.00 ©2016 IEEE