SEMANTIC HMC: A PREDICTIVE MODEL USING MULTI-LABEL CLASSIFICATION FOR BIG DATA

Rafael Peixoto[1,2], Thomas Hassan[1], Christophe Cruz[1], Aurélie Bertaux[1], Nuno Silva[2]

[1] Le2i, University of Burgundy, Dijon, France, {thomas.hassan, christophe.cruz, aurelie.bertaux}@u-bourgogne.fr
[2]GECAD, Polytechnic of Porto, Porto, Portugal,{rafpp,nps}@isep.ipp.pt

Abstract— **One of the biggest challenges in Big Data is the exploitation of Value from large volumes of data. To exploit value one must focus on extracting knowledge from Big Data sources. In this paper we present a new simple but highly scalable process to automatically learn the label hierarchy from huge sets of unstructured text. We aim to extract knowledge from these sources using a Hierarchical Multi-Label Classification process called Semantic HMC. Five steps compose the Semantic HMC (Indexation, Vectorization, Hierarchization, Resolution and Realization). The first three steps construct the label hierarchy from data sources. The last two steps classify new items according to the hierarchy labels. To perform the classification without relying heavily on the user, the process is unsupervised, thus no thesaurus or label examples are required. The process is implemented in a scalable and distributed platform to process Big Data.**

Keywords- classification; multi-classify; Big-Data; ontology; semantic technologies; machine learning

I. INTRODUCTION

Nowadays, discovering knowledge and insights over web data is a major task for most corporations to increase their competitiveness. Determining the value of information relative to a particular customer is a complex task addressed by both the business intelligence and the data-mining field [1].

In classification field, an item is anything that can be classified (e.g. music, a video or a book). The item analysis is not a trivial process and proper techniques for analysis and representation are required. In the context of Big Data, this task is even more challenging, due to Big Data characteristics. An increasing number of V's has been used to characterize Big Data [2], [3]: Volume, Velocity, Variety and Value. Volume concerns the large amount of data that is generated and stored through the years by social media, sensor data, etc.[2]. 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 his content [4] such as audio, video, webpage, and text, as well as traditional structured data. Value measures how valuable the information to a Big Data consumer is. Value is the most important feature of Big Data and its "raison d'être", because the user expects to make profit out of valuable data.

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) may still be utilized for Big Data Analysis [2]. Hierarchical Multi-Label Classification (HMC) is a Data Mining method defined as the combination of Multi-Label classification and Hierarchical classification [6]. In Hierarchical classification, the labels are organized in classes and super-classes forming a hierarchical structure that can usually be in two ways [7]: (1) Tree-structured hierarchies and (2) Directed Acyclic Graph (DAG) in which a class can have multiple parents. In HMC, the items can be assigned to different hierarchical paths and may simultaneously belong to different class labels in the same hierarchical level.

The proposed Semantic Hierarchical Multi-label Classification process (Semantic HMC) [8], allows systems to automatically analyse and describe the items. To represent the knowledge in the Semantic HMC process, an Ontology-described Knowledge Base is used. Ontologies [9] are the most accepted way to represent semantics in the Semantic Web [10] and a good solution for intelligent computer systems that operate close to the human concept level, bridging the gap between human conceptions and computational requirements [11].

The Semantic HMC is based on a non-supervised ontology learning process using scalable Machine-Learning techniques and Rule-based reasoning. The process is unsupervised: no previously classified examples or rules to relate the data items with the labels are necessary. The ontology is automatically learned from the data through scalable Machine Learning techniques. The Semantic HMC process learns the ontology (Tbox) from the huge Volume and Variety of initial data using Machine learning and Big Data Technologies. First the taxonomy is automatically obtained and used as the first input for the ontology construction [12]. Then, for each taxonomical concept (representing the classification labels) a set of rules is created using the Semantic Web Rule Language (SWRL) to relate the data items to the taxonomy concepts. Then the learned ontology is populated with the data items (Abox) resulting in an ontology-described knowledge base (Abox+Tbox). The Semantic HMC proposes five individually scalable steps to reach the aims of Big Data analytics .

- *Indexation* extracts terms from data items and creates an index of data items.
- *Vectorization* calculates the term-frequency vectors of the indexed items.
- *Hierarchization* creates the label taxonomy (i.e. subsumption hierarchy) using term-frequency vectors.
- *Resolution* creates the reasoning rules to relate data items with the labels based on term-frequency vectors.
- *Realization* first populates the ontology with items and then for each item determines the most specific label and all its subsuming labels.

This paper focuses on the three first steps of the Semantic HMC process. It proposes a new simple but highly scalable process to learn automatically the label hierarchy from huge sets of unstructured texts. The process is implemented using technologies for Big Data that distributes the process by several machines in order to reach high performance and scalability.

The rest of the paper covers five sections. The second section presents background and related work. The third section describes the label hierarchy learning process. The forth section describes the process implementation in a scalable and distributed platform to process Big Data. The fifth section discusses the results. Finally, the last section draws conclusions and suggests further research.

II. Background and Related Work

In this section, we introduce some background and discuss the current related work about automatic taxonomy learning from unstructured text.

Unsupervised learning methods try to suitably organize the elements on classes based on statistical properties only [5] [13]. In an unsupervised Hierarchical Multi-Label Classification process, the labels and their organization in a hierarchy form are learned from statistical text analysis. Two main different aspects of label hierarchy extraction from text can be distinguished: (1) label extraction and filtering (2) hierarchy creation.

To extract terms from a set of documents several methods exists in literature. These methods can be categorized into two different approaches: linguistic and statistical approaches. Linguistic approaches use natural language processing (NLP) for term extraction [14]–[16]. A common linguistic approach is the part-of speech tagging that labels the part-of-speech (e.g., noun, adjective, verb, etc.) in a text and often selects the nouns and adjectives as terms [15]. On the other hand, the statistical approaches use probabilistic techniques to extract the terms from text, such as [17]–[19]. These methods have a good performance in evaluating the term relevancy but lack the advanced grammatical analysis of linguistic methods that can capture the function of words in a sentence.

Based on the identification of terms and concepts, several methods exist to create hierarchical relations between concepts, including [20], [21]:

- *Hierarchical clustering* that starts with one cluster and progressively merges clusters that are closest.
- *Subsumption methods* that construct the concept broader-narrower relations based on the co-occurrence of concepts [22].

The main advantage of the subsumption method comparing to the hierarchical clustering is its relation between the processing speed and the ability to provide good concept broader–narrower relations. The advantages and drawbacks of each method is deeply studied in [20].

Based on the proven advantages and disadvantages of each approach we propose a process to learn the hierarchy using: a hybrid approach between statistical and lexical approaches to learn the labels and a highly scalable subsumption method to learn the hierarchical relations between labels. This process allows to automatically learning a label hierarchy from huge volumes of unstructured data. This label hierarchy is then used to perform Hierarchical Multi-Label Classification.

III. Label Hierarchy Learning

In this section the automatic label hierarchy learning process is described in detail. The label hierarchy is automatically learned based on a Description Logic ontology presented in Table 1. 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 following subsections describe how the terms are extracted from text items, which terms are used to classify the items (Labels) and at last how the subsumption relations between the labels are created.

A. Indexation

The indexation step extracts terms from a collection of items and creates an index of items. In the example of indexing text documents, the extracted terms are relevant words to describe an item.

TABLE I. The Core Ontology

DL concepts	Description
$Term \sqsubseteq \top$	Extracted terms (e.g. word)
$Label \sqsubseteq Term$	Terms used to classify the items
$Label \sqsubseteq \exists broader.Label$	Broader relation between labels
$Label \sqsubseteq \exists narrower.Label$	Narrower relation between labels
$broader \equiv narrower^{-}$	Broader and narrower are inverse relations

The result is the inverted index (Fig.1.), represented by a set of vectors in the form: $vector_t < i_1, i_2, \dots, i_n >, i \in C$, where t is a term, i_j are the data items where the term is observed/occurs and C a collection of n items.

The inverted index allows efficient retrieval of terms.

B. Vectorization

In the Vectorization step, the identified terms from a collection of documents give rise to two types of vectors [17]:

- *Item-frequency* (document frequency), $\mathcal{V}_{df} = \{(t, df_{t,C})\}$ where t is the term, C is the collection of items and $df_{t,C}$ is the number of items from the collection C where the term t is observed.
- *Term-frequency* in each item, $\mathcal{V}^i_{tfidf} = \{(t, tfidf_{i,t,C})\}$, where t is the term, i is the item, C is the collection of items and $tfidf_{i,t,C}$ is the TF-IDF value [17] of term t in item i regarding the collection of items C.

Further, a term co-occurrence frequency matrix cfm is created to represent the co-occurrence of any pair of terms $(term_i, term_j)$ in the collection of items C, such that:

$$cfm(term_i, term_j) = \left|\left\{item_n \in C \middle| item_n \in vector_{term_i} \wedge item_n \in vector_{term_j}\right\}\right| \quad (1)$$

where $item_n \in Item$, $vector_{term_i}$ is the vector of the inverted index for $term_i$ and $vector_{term_j}$ is the vector of the inverted index for term $term_j$.

Let n denote the number of different terms in a collection of items C, the term co-occurrence matrix for the collection C is a $n \times n$ symmetric matrix. The main diagonal of the co-occurrence matrix $cfm\ (term_i, term_i)$ denotes the occurrence of $term_i$ in all items of C (i.e. the document frequency). Hence the main diagonal $cfm\ (term_i, term_i)$ is the maximum value for the line $term_i$ and the column $term_i$ of the co-occurrence table. Table 2 depicts an example of a 3×3 terms co-occurrence frequency-matrix, where $cfm(Term_1, Term_2) = cfm(Term_2, Term_1) = 70$. The main diagonal is depicted in black and all co-occurrence terms are depicted in white.

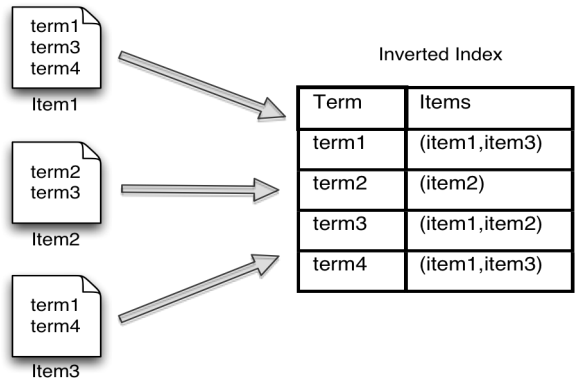

Fig. 1. Inverted Index example

TABLE II. TERM CO-OCURRENCE MATRIX EXAMPLE

	Term₁	**Term₂**	**Term₃**
Term₁	95	70	80
Term₂	70	80	60
Term₃	80	60	90

C. Hierarchization

The most relevant terms are designated as labels. The hierarchization step learns the labels and their subsumption relations. To evaluate the term relevancy the information retrieval method described in [23] 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:

$$P_C(term_j) = \frac{df_C(term_j)}{|C|} \quad (2)$$

where $df_C(term_j)$ is the document frequency of the $term_j$ in the collection of items C and $|C|$ 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:

$$\omega_{lT} = \{term_j \in Term | P_C(term_j) \geq lT\} \quad (3)$$

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.

To build the subsumption hierarchy, we adopt the subsumption method because of its proven performance. A scalable method based on [22] is used, that exploits the co-occurrence matrix. Label x potentially subsumes label y if (Fig. 2 (A)):

$$(P_C(x|y) = 1) \wedge (P_C(y|x) < 1) \quad (4)$$

where:

- $P_C(x|y)$ is the conditional proportion (number) of the items from collection C common to x and y, in respect to the number of items in y such that:

$$P_C(x|y) = \frac{x \cap y}{y} = \frac{cfm\ (x,y)}{df_y} \quad (5)$$

- $P(y|x)$ is the conditional proportion (number) of the items from collection C common to x and y, in respect to the number of items in x such that

$$P_C(y|x) = \frac{x \cap y}{x} = \frac{cfm\ (x,y)}{df_x} \quad (6)$$

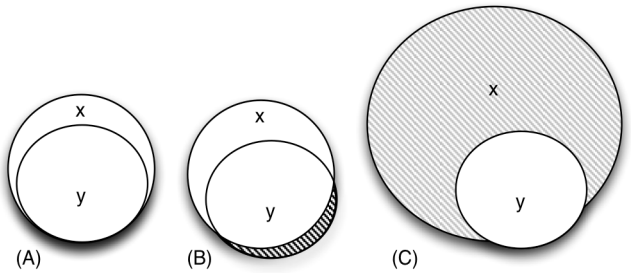

Fig. 2. Heuler diagrams of the subsumption method

Although in [22] the authors noticed that many subsumed concepts are failing to be included because a few occurrences of the subsumed term y does not co-occur with the term x. By relaxing the first condition ($P_C(x|y) = 1$) it is possible to capture these failing concepts (Fig. 2(B)). The subsumption with the first condition relaxed is redefined as:

$$(P_C(x|y) \geq st_1 \wedge (P_C(y|x) \ < 1) \tag{7}$$

where $st_1 \in [0,1]$ is the subsumption co-occurrence threshold for the first condition.

In [20] the authors also propose to relax the second condition ($P_C(y|x) < 1$) allowing to define how larger the set of items containing the subsuming term x must be in relation to the subsumed term y (Fig. 2 (C)). The subsumption is then redefined as:

$$(P_C(x|y) \geq st_1) \wedge (P_C(y|x) \ < st_2) \tag{8}$$

where $st_2 \in [0,1]$ is the subsumption co-occurrence threshold for the second condition and$st_1 \geq st_2$.

Hence if x appears in at least proportion st_1 of the documents in which y also appears, and y appears in less than proportion st_2 in which x appears, then x potentially subsumes y, i.e. $x \prec y$.

By applying this method to all the labels, the result is a subsumption hierarchy of labels, and more specifically a Directed Acyclic Graph (DAG).

IV. IMPLEMENTATION

This section describes the implementation of the proposed label hierarchy learning process. The process is implemented as a Java application, as many Java libraries that support the proposed methods are available. The next subsections describe the implementation details of each step of the process.

A. Indexation

In this implementation, the indexation step extracts terms from a collection of text documents and creates an index of text documents.

Term extraction includes spelling correction, stop-word and synonym detection and calculation of composed terms. Composed terms are calculated by collocation. A collocation is a sequence of words (n-grams) which co-occur more often than it would be expected by chance. An association measure algorithm evaluates whether the co-occurrence is purely by chance or statistically significant. In this implementation a parser and a tokenizer are used to extract terms from text documents. Statistically based methods are used to identify relevant combinations of words and the Log-Likelihood [24] association measure is used for composed term calculation.

Solr server [25] is used as parser, tokenizer and indexer. Collocation's computation is performed using Mahout library [25]. Mahout is a scalable open source machine-learning library especially addressed to process collections of data that are too large for a single machine. Solr and Mahout are deployed on a Hadoop cluster [https://hadoop.apache.org/]. We choose the Hadoop implementation of MapReduce because of its open-source nature and its ability for integration with the used tools.

MapReduce [26] is a programming model, which addresses large scale data processing on several machines. In the MapReduce paradigm, "users specify a map function that processes a key/value pair to generate a set of intermediate key/value pairs, and a reduce function that merges all intermediate values associated with the same intermediate key".

The output is an inverse index of the resulting terms stored in HDFS (Hadoop distributed file system), which will be used in vectorization step.

B. Vectorization

In the vectorization step, term frequency vectors and the co-occurrence matrix are calculated.

Term frequency vectors calculation includes the calculation of document frequency $\mathcal{V}_{\text{df}}$ and term frequency in each text document (item) $\mathcal{V}_{\text{tfidf}}^{\text{item}}$. Calculate term frequency vectors in a large collection of text documents are very intensive tasks. We aim to distribute this process on several machines using distributed programming technologies such as MapReduce.

The term co-occurrence matrix cfm is created to represent the co-occurrence of any pair of terms in the items $cfm\left(term_i, term_j\right)$. As calculating each co-occurrence is a very intensive task we develop a MapReduce algorithm to create the co-occurrence matrix.

Two methods are commonly used to compute a co-occurrence matrix from text: the pair method and the stripes method. These algorithms are adapted to the MapReduce paradigm [27] and both use the same statistical principle to find co-occurrences in text.

In the MapReduce's map function, for all given items, all pairs of terms $\left(term_i, term_j\right)$ within a window of neighborhood are counted as a co-occurrence. In the pair algorithm, each co-occurrence is outputted while in the stripes approach, each output consists in a term and the list of its co-occurring terms. Considering our cluster specifications, we used the stripes approach, as the pair algorithm output is more important, hence it is limited by disk input/output.

The co-occurrences are emitted and passed to the reducer for counting. The $(key_i, value_i)$ pairs emitted are defined as:

- key_i is a term, $Term_i$
- $value_i$ is the list of co-occurring terms of size n: $(term_0, term_1, \dots, term_n)$

Where n is the number of terms in the current item, as in our implementation, the neighborhood window used to define a co-occurrence is the whole item.

According to the MapReduce paradigm, the pairs are shuffled by key_i and the MapReduce's reduce function is executed for each set of pairs with the same key_i. The reduce function aggregates the terms from the $value_i$ of all pairs, counting each occurrence. The reduce function output is a list of pairs in the format $(key_o, value_o)$ where the key_o is composed by each combination of key_i and each term in $value_i$ of all input pairs, and the $value_o$ is the counted term occurrence.

Term-frequency vectors calculation is handled by the Mahout library [25] deployed in a Hadoop Cluster. Also, the Hadoop cluster handles the MapReduce algorithm and stores the vectors and the matrix in HDFS.

The calculated matrix is used in the hierarchization step to create the subsumption relations.

C. Hierarchization

The hierarchization step learns the labels and their subsumption relations automatically from text documents based on a statistical approach. The labels are calculated by an information retrieval approach based in the proportion $P_C(term)$ of a $term \in Term$ and the collection C.

Detecting subsumption relations between labels is a very intensive task. To distribute the process for several machines the previously described subsumption algorithm was developed in MapReduce paradigm, by turning the process into a set of MapReduce operations. As previously described, the Vectorization phase outputs a co-occurrence matrix cfm for representing the co-occurrence of any pair of terms $(term_i, term_j)$ in the collection of items C. Considering the Item (document) frequencies of $term_i$ and $term_j$ in collection C, this matrix can be viewed as a set of pairs $< (term_i, term_j), P(x|y) >$, where $P(x|y)$ is the conditional proportion of the items common to x and y, in respect to the number of items in y.

The set of pairs $< (term_i, term_j), P(x|y) >$ are used as the input of the map function. The $(key, value)$ pairs are defined as:

- key is a tuple $(term_i, term_j)$ where both $term_i$ and $Term_j$ are terms identified in the Vectorization step.
- $value$ is the proportion $P(x|y)$

In the map phase, the Threshold lT is applied to each pair. If both terms are included in the set ω_{lT}, the map function emits the couple $((label_i, label_j), P(x|y))$ where $(label_i, label_j)$, are terms in the set ω_{lT}.

The reduce function will group outputs by Label, which allows to compute the subsumption potential of each couple $(label\ i, label\ j)$ according to the relation defined above. The output of the reduce function consists of the set of couples $< (label_i, label_j) >$, where $label_j$ is a potencial parent of $label_i$. Selection of the best potential parents can be done in the reducer, but this selection is out of the scope of this paper.

The subsumption algorithm is deployed in a Hadoop Cluster and the hierarchy of Labels is stored in HDFS. The output is a Label hierarchy that is used to classify the items. However, the item classification is out of the scope of this paper.

V. RESULTS

In this section the preliminary results of the proposed label hierarchy learning process are discussed. First the dataset, the environment and the settings used to test the process are described. Then the process results are presented and discussed.

A. DataSet

The dataset used to process the preliminary results are unstructured text articles extracted from dumps of the French version of Wikipedia. We use four sub-datasets with different sizes as presented in Table 3.

B. Test environment

The prototype is deployed on remote servers provided by Google, through their Google Compute Engine service that allows several ways to deploy a virtual Hadoop cluster and execute MapReduce Jobs over it. For this perliminary test, the cluster is composed of 4 nodes (1 master node and 3 workers). Each node corresponds to an instance in Google Compute Engine, i.e. a virtual machine with its proper resources.

Table 4 describes the specifications of the instances used for this test. We use the native Hadoop web interfaces in conjunction with Apache Ambari [https://ambari.apache.org/] to monitor the different sub-processes of our prototype. In particular, we are interested in time and resources costs for the different phases.

TABLE III. WIKIPEDIA-BASED DATASETS

Dataset	Number of articles	Size (Gb)
Wikipedia 1	174900	1.65
Wikipedia 2	407000	2.21
Wikipedia 3	994000	5
Wikipedia 4	2788500	11

TABLE IV. TEST NODE SPECIFICATION

Resource type	Description
CPU (per node)	2.5GHz Intel Xeon E5 v2 (Ivy Bridge)
RAM (per node)	7.5GB
Disk (per node)	500GB

C. Settings

Some thresholds and settings used in the process have a high impact on the results. In the indexation step we used seq2sparse tool from Mahout to retrieve the collocation terms with the Lucene French analyzer and pruning with the Minimum support, n-gram size, Maximum document frequency, Minimum document frequency and Log-Likelihood Ratio settings. More details about each option can be found in [25]. Table 5 shows the different parameters and their values used. The same values are used for all datasets.

D. Results

The aim of the perliminary test is to check the scalability of the system according to the number of items from the same dataset. For that we monitor: (1) The execution time for each step and dataset; (2) The number of extracted Terms from each dataset; (3) The number of learned Labels from each dataset; (4) The number of learned Subsumption relations.

The number of extracted terms is depicted in Fig. 3 where the reader can observe a growth of the number of terms according to the number of items. As the analyzed data is textual data, a limit of growth (not visible with the dataset used) is expected to happen, i.e. the language dictionary and its derivations size. Text processing such as stemming and synonym detection can be optimized to further restrict the number of terms detected.

TABLE V. EXECUTION SETTINGS

Parameter	Step	Value
Minimum support	Indexation	1750
n-gram size	Indexation	2
Maximum document frequency	Indexation	90%
Minimum document frequency	Indexation	1
Log-Likelihood Ratio	Indexation	17500
Label Detection Threshold (*lT*)	Hierarchization	2%
Subsumption st2 Threshold	Hierarchization	10%
Subsumpion st1 Threshold	Hierarchization	90%

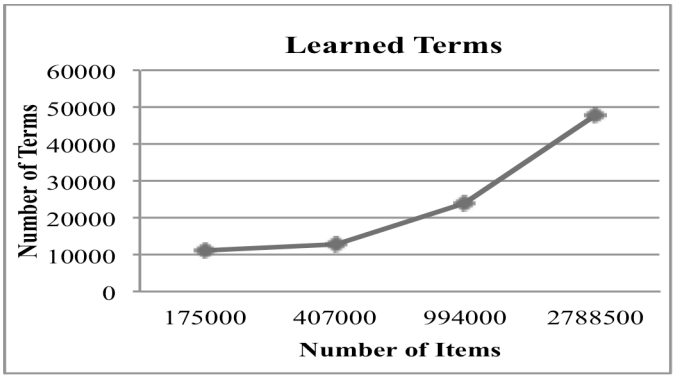

Fig. 3. Number of extracted Terms according to each dataset.

The number of learned labels for each dataset is depicted in Fig. 4. As the label set is a subset of the set of terms extracted from an information retrieval approach, the extracted number of terms impacts the number of learned Labels as observed. Nethertheless we can observe that, the number of labels decreases while the size of the dataset grows. This decrease is a consequence of the label detection threshold (2%, Table 5). As the dataset grows, this threshold makes it unliky for a term to appear at the same rate, because of the dataset diversity. We expect to show the impact of changing this threshold dynamically as the dataset grows in future work.

The number of learned subsumption relations for each dataset is depicted in Fig. 5. We can observe a decrease in the number of learned relations as a consequence of the decrease of the number of learned labels (Fig. 4).

The execution time of each step by dataset is depicted in Fig. 6. By observing the results, the computation time of the matrix creation is the most expensive of the process, regardless of the data size. In future work, we aim to add a combiner between the Map and the Reduce phase of the matrix creation to decrease computation time as well as the size of the data emitted to the reducers.

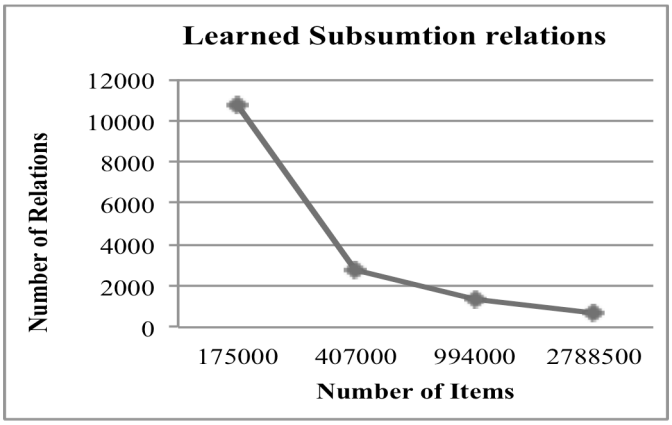

Fig. 4. Number of learned Labels according to each dataset.

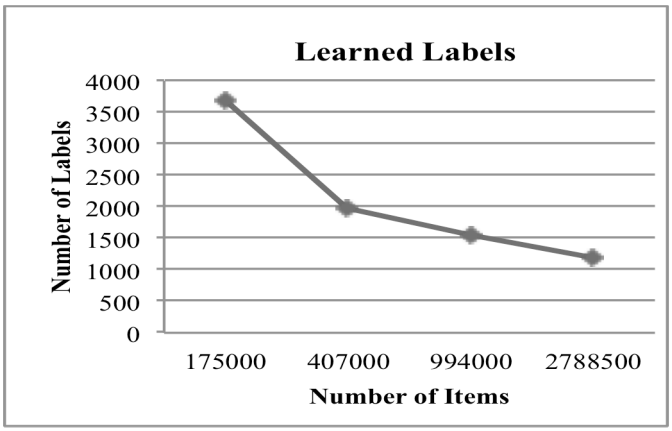

Fig. 5. Number of learned relations according to each dataset.

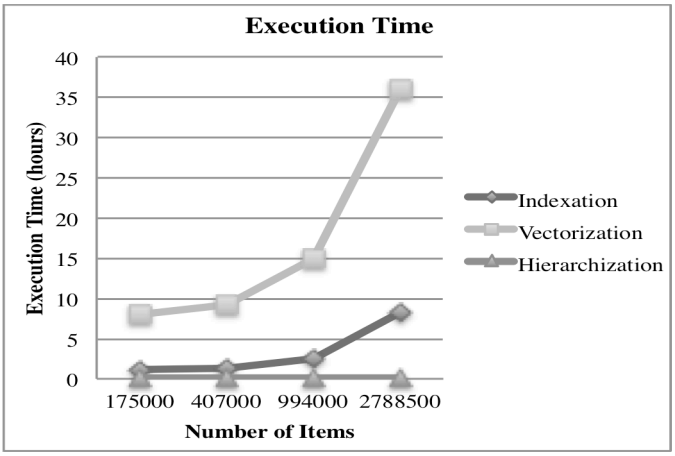

Fig. 6. Execution time of each step by dataset.

VI. CONCLUSIONS

To perform Hierarchical Multi-Label classification in Big Data without overload the users, this paper describes in detail an unsupervised process to learn a label hierarchy from unstructured text. The process is based on a hybrid approach between statistical and lexical approaches to learn the labels and a highly scalable subsumption method to learn the hierarchical relations between labels.

The process prototype was successfully implemented in a scalable and distributed platform to process Big Data. In preliminary results the label hierarchy is automatically learned from large datasets of unstructured text data.

In future work we aim to improve the term detection process by optimizing the different parameters used in the tests, as well as refining our statistical results with advanced NLP processing such as word sense disambiguation and named entity detection. We also aim at measuring the quality of the automatically learned hierarchy by using a gold standard to evaluate the process.

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] I. H. Witten and E. Frank, *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2005.

[2] M. Chen, S. Mao, and Y. Liu, "Big Data: A Survey," *Mob. Networks Appl.*, vol. 19, no. 2, pp. 171–209, Jan. 2014.

[3] P. Hitzler and K. Janowicz, "Linked data, big data, and the 4th paradigm," *Semant. Web*, vol. 4, pp. 233–235, 2013.

[4] A. Syed, K. Gillela, and C. Venugopal, "The Future Revolution on Big Data," *Future*, vol. 2, no. 6, pp. 2446–2451, 2013.

[5] F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, *Recommender Systems Handbook*, 1st Editio. Springer, 2010.

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

[7] R. Cerri, R. C. Barros, and A. C. P. L. F. de Carvalho, "Hierarchical multi-label classification using local neural networks," *J. Comput. Syst. Sci.*, vol. 80, no. 1, pp. 39–56, 2014.

[8] T. Hassan, R. Peixoto, C. Cruz, A. Bertaux, and N. Silva, "Semantic HMC for big data analysis," in *Big Data (Big Data), 2014 IEEE International Conference on*, 2014, pp. 26–28.

[9] T. R. Gruber, "A Translation Approach to Portable Ontology Specifications by A Translation Approach to Portable Ontology Specifications," *Knowl. Creat. Diffus. Util.*, vol. 5, no. April, pp. 199–220, 1993.

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

[11] 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.

[12] D. Fensel, *Ontologies: Silver Bullet for Knowledge Management and Electronic Commerce*. Heidelberg, Germany: Springer-Verlag, 2001.

[13] N. K. Amandeep Kaur Mann, "Review Paper on Clustering Techniques," *GJCST*, pp. 43–47, 2013.

[14] M. a. Hearst, "Automatic Acquisition of Hyponyms ftom Large Text Corpora," *Proc. 14th Conf. Comput. Linguist.*, vol. 2, pp. 23–28, 1992.

[15] K. Toutanova and C. D. Manning, "Enriching the Knowledge Sources Used in a Maximum Entropy Part-of-Speech Tagger," *Proc. Jt. SIGDAT Conf. Empir. Methods Nat. Lang. Process. Very Large Corpora*, pp. 63–70, 2000.

[16] P. Cimiano, S. Staab, and J. Tane, "Automatic acquisition of taxonomies from text: FCA meets NLP," in *Proceedings of the International Workshop & Tutorial on Adaptive Text Extraction and Mining*, 2003.

[17] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Inf. Process. Manag.*, vol. 24, pp. 513–523, 1988.

[18] A. Maedche and R. Volz, "The ontology extraction & maintenance framework Text-To-Onto," *Proc. Work. Integr. Data ...*, pp. 1–12, 2001.

[19] S.-T. W. S.-T. Wu, Y. L. Y. Li, Y. X. Y. Xu, B. P. B. Pham, and P. C. P. Chen, "Automatic Pattern-Taxonomy Extraction for Web Mining," *IEEE/WIC/ACM Int. Conf. Web Intell.*, 2004.

[20] J. de Knijff, F. Frasincar, and F. Hogenboom, "Domain taxonomy learning from text: The subsumption method versus hierarchical clustering," *Data Knowl. Eng.*, vol. 83, pp. 54–69, Jan. 2013.

[21] K. Meijer, F. Frasincar, and F. Hogenboom, "A Semantic Approach for Extracting Domain Taxonomies from Text," *Decis. Support Syst.*, Mar. 2014.

[22] M. Sanderson and B. Croft, "Deriving concept hierarchies from text," *Proc. 22nd Annu. Int. ACM SIGIR Conf. Res. Dev. Inf. Retr. - SIGIR '99*, pp. 206–213, 1999.

[23] R. Feldman, M. Fresko, Y. Kinar, Y. Lindell, O. Liphstat, M. Rajman, Y. Schler, and O. Zamir, "Text Mining at the Term Level," in *Second European Symposium, PKDD '98*, 1998, pp. 65–73.

[24] T. Dunning, "Accurate methods for the statistics of surprise and coincidence," *Comput. Linguist.*, vol. 19, no. 1, pp. 61–74, 1993.

[25] The Apache Software Foundation, "Mahout," 2013. [Online]. Available: http://mahout.apache.org. [Accessed: 15-Feb-2015].

[26] J. Dean and S. Ghemawat, "MapReduce : Simplified Data Processing on Large Clusters," *Commun. ACM*, vol. 51, no. 1, pp. 1–13, 2008.

[27] J. Lin, "Monoidify! monoids as a design principle for efficient mapreduce algorithms," *arXiv Prepr. arXiv1304.7544*, 2013.