Industrial Requirements Classification for Redundancy and Inconsistency Detection in SEMIOS

Manel Mezghani, Juyeon Kang, Florence Sèdes

Industrial requirements classification for redundancy and inconsistency detection in SEMIOS

Manel Mezghani
Semios for requirements
Prometil
Toulouse, France
mezghani.manel@gmail.com

Juyeon Kang
SEMIOS for requirements
Prometil
Toulouse, France
j.kang@semiosapp.com

Florence Sèdes
IRIT, University of Toulouse
CNRS, INPT, UPS, UT1, UT2J,
France
florence.sedes@irit.fr

***Abstract*—Requirements are usually "hand-written" and suffers from several problems like redundancy and inconsistency. The problems of redundancy and inconsistency between requirements or sets of requirements impact negatively the success of final products. Manually processing these issues requires too much time and it is very costly. The main contribution of this paper is the use of k-means algorithm for a redundancy and inconsistency detection in a new context, which is Requirements Engineering context. Also, we introduce a filtering approach to eliminate "noisy" requirements and a preprocessing step based on the Natural Language Processing (NLP) technique to see the impact of this latter on the k-means results. We use Part-Of-Speech (POS) tagging and noun chunking to detect technical business terms associated to the requirements documents that we analyze. We experiment this approach on real industrial datasets. The results show the efficiency of the k-means clustering algorithm, especially with the filtering and preprocessing steps. Our approach is using the software SEMIOS and will be integrated as a new functionality.**

***Index Terms*—Requirements engineering, redundancy, inconsistency, clustering, NLP, technical documents**

I. Introduction

In order for a system to become operational in real applications, several stages of conception, development, production, use, support and retirement must be followed (ISO/IEC TR 24748-1, 2010). During the conception stage, we identify and document the stakeholder's needs in the system requirements specification [1]. Writing clearly all required elements without ambiguities [2] in the specifications is an essential task before passing to the development stage [3], [4]. According to the *2015 Chaos report* by the Standish Group[1], only 29% of projects were successful[2], 50% of the challenged projects are related to the errors from the Requirements Engineering (RE) and 70% of them come from the difficulties of understanding implicit requirements. All these errors do not lead to project failure but generate useless information. It is well known that the costs to fix errors increase much more after that the product is built than it would if the requirements defects [5] were discovered during the requirements phase of a project [6], [7]. When writing or revising a set of requirements, or any technical document, it is particularly challenging to make sure that texts are easily readable and are unambiguous for any domain actor. Experience shows that even with several levels of proofreading and validation, most texts still contain a large number of language errors (lexical, grammatical, style), and also a lack of overall concordance, or redundancy and inconsistency in the underlying meaning of requirements. In particular, manually identifying redundant or inconsistent requirements is an obviously time-consuming and costly task. We tackle these problems in terms of similarity between requirements since more than two similar requirements can be classified as redundant or inconsistent requirements.

The problems of redundancy and inconsistency can be handled according to different technologies. We focus on artificial intelligence approaches and more precisely classification approaches. Automatic classification of requirements is widely used in the literature using convolutional neural networks [8], naives bayes classifier [9], text classification algorithms [10]. Data classification approaches could be data clustering through algorithm such as k-means. This latter is studied in different contexts due to its efficiency [11]. However, in requirements engineering context, we could not find advanced works on the redundancy and inconsistency issues using k-means algorithm.

The main contribution of this paper is the use of k-means algorithm for a redundancy and inconsistency detection in a new context, which is requirements engineering context. Also, we introduce a preprocessing step based on Natural Language Processing (NLP) techniques in order to assess the impact of this latter to the k-means results. We use Part-Of-Speech (POS) tagging and noun chunking to detect technical business terms associated to the requirements documents that we analyze.

This paper is structured as follows: In Section II, we present related works on the redundancy and inconsistency detection through artificial intelligence approaches by focusing on the k-means technique. In Section III, we present our clustering approach. In Section IV, we explain the validation approach used to evaluate the relevance of our results. In Section V, we present the datasets used to evaluate our approach and the results obtained by applying our clustering approach. In Section VI, we discuss the associated results. In Section VII, we conclude and give some future research directions.

[1] $http://www.standishgroup.com$

[2] They studied 50,000 projects around the world, ranging from tiny enhancements to massive systems re-engineering implementations.

DOI : 10.1109/RE.2018.00037

II. Related works

In this Section, we first present related works associated to redundancy and inconsistency detection in specifications documents or technical documents. Second, we give some researches focusing on text preprocessing in requirements engineering context. Finally, we focus on approaches using k-means clustering in the latter context.

A. Redundancy and inconsistency detection

Researches on redundancy detection began by traditional bag-of-words (BOW), TF-IDF frequency matrix, and n-gram language modeling [12] [13]. Then, researchers like Juergens et al. [14] use ConQAT to identity copy-and-paste reuses in requirements specifications. Falessi et al. [15] detect similar content using information retrieval methods such as Latent Semantic Analysis. They compare NLP techniques on a given dataset to correctly identify equivalent requirements. Rago et al. [16] extend the work presented in [15] specifically for use cases. Their tool, ReqAlign, combines several text processing techniques such as a use case-aware classifier and a customized algorithm for sequence alignment.

Inconsistency is analyzed in [17] by proposing the framework of a patterns-based unsupervised requirements clustering (based on k-means algorithm), called PBURC, which makes use of machine-learning methods for requirements validation. This approach aims to overcome data inconsistencies and effectively determine appropriate requirements clusters for optimal definition of software development sprints. Dermeval et al., [18] present a survey about how using ontologies in RE activities both in industry and academy, is beneficial, especially for reducing ambiguity, inconsistency and incompleteness of requirements.

Ambiguity is usually related to redundancy/inconsistency. Recently, Sabriye et al., [19] are using POS tagging in order to detect ambiguity in software requirements specification. Shah et al., [20] present a survey of the currently available tools for ambiguity resolution. According to this study, the presented approaches are classified as automated and semi-automated and they use NLP tools such as extracting requirements from the document, tag the requirements sentence and find duplicate requirements.

B. Preprocessing

Some researches introduce preprocessing steps in requirements analysis context. According to [21], the preprocessing helps reducing the inconsistency of requirements specifications by leveraging rich sentence features and latent co-occurrence relations. It is applied through i) a Part-Of-Speech tagger [22], ii) an entity tagging through a supervised training data, iii) a temporal tagging through a rule-based temporal tagger and iv) co-occurrence counts and regular expressions. This preprocessing approach improved the performance of an existing classification method.

Preprocessing data for redundancy detection is used in [23] by performing standard NLP techniques such as removing English stop words and striping off the newsgroup related metadata (including noisy headers, footers and quotes). The Joint Neural Network for redundancy detection approach in [23] also uses normalized bag-of-words (BOW) as a preprocessing approach. The normalized BOW generates a global uni-gram based dictionary mapping. With the presence of the uni-gram indexer, the authors could readily remove low frequency terms and lengthy snippets.

C. k-means

k-means clustering is a popular type of unsupervised learning approach, which is used on unlabeled data (i.e., data without defined categories or groups). The goal of this algorithm is to cluster the data into k groups (k number of groups).

Classifying requirements is an important task in requirements engineering. Recently, some studies introduce k-means in requirements classification tasks. Notably, [21] applies different approaches such as i) topic modeling using Latent Dirichlet Allocation (LDA) and Biterm Topic Model (BTM) and ii) clustering using k-means, Hierarchical approach and Hybrid (k-means and hierarchical) to classify requirements into functional (FR) and non-functional requirements (NFR). k-means algorithm shows its efficiency in this work.

III. Clustering Approach

The main steps of our approach are shown in Figure 1. Given an industrial specification, we extract first the requirement file containing only requirements to analyze using a predefined function in SEMIOS software. Second, we apply a pretreatment step in order to eliminate which we call noisy requirements. Third, we use a POS tagging in order to extract business terms. Last, we apply a k-means clustering algorithm. We detail and explain these steps in the sections below.

A. Requirements quality analysis: SEMIOS

SEMIOS[3] is a software for detecting errors in specifications from the conception phase. The core semantic engine of this tool is based on NLP techniques and works directly with RE domains tools like IBM DOORS, IBM Doors Next Generation, MS Word, MS excel, etc. It aims to control specifications quality and reduce management cost. The clustering approach that we propose in this work will allow to overcome a shortcoming in the current version of SEMIOS like possibility to analyze requirements coming from different specifications. The result of this work will be integrated as a new functionality.

B. Pretreatment

Redundancy has negative effects on document maintenance, but it also eases readability, if relevant information/requirements are repeated in the respective context.

In this section, we explain how we filter noisy requirements. We consider some requirements as noisy when they are written exactly in same words and found in different chapters. According to our RE expert, these identical requirements are in most cases non-redundant, therefore should be discarded. Keeping all the specifications for analysis leads us to consider

[3]http://www.semiosapp.com/

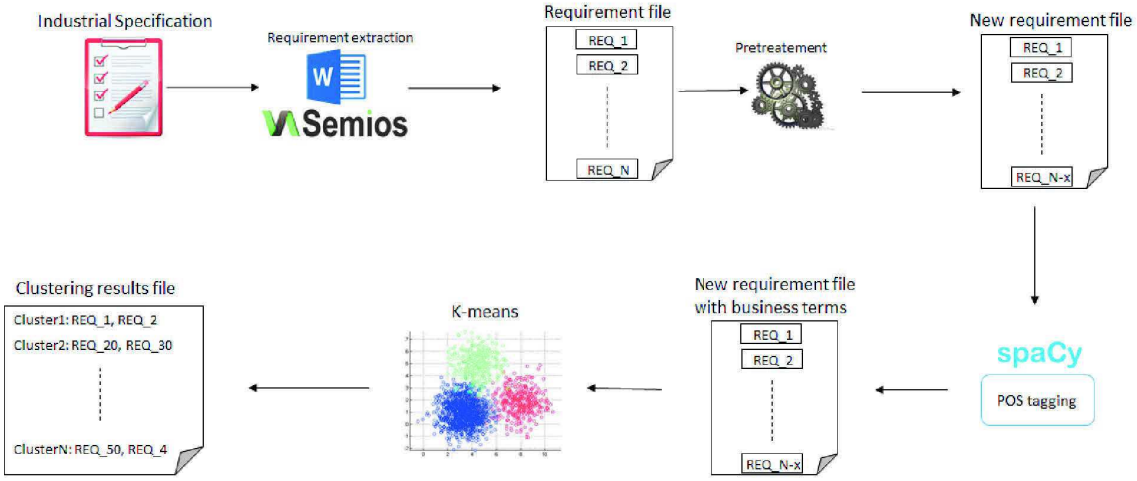

Fig. 1. Clustering approach overview

all the requirements and may impact the clustering results. That is the reason why we proceed to eliminate the obvious false positive elements in the specifications before clustering.

As described in Figure 1, we use the software SEMIOS to extract only requirements sentences from specifications documents which contain normally many other elements than requirements like table of contents, introduction, remarks, conclusion, annexes, etc.

We explain the preprocessing step of creating a new requirements file without identical requirements belonging to the different chapter in the Algorithm 1. Let us assume that a requirement file $ReqFile = \{Req_1, ..., Req_\bullet ..., Req_n\}$ where $\bullet \in \{1, n\}$ and n is the number of requirements in the $ReqFile$. $Req_\bullet$ is defined with a unique ID and also a path describing its position according to chapters information. $Req_\bullet.chapter()$ return a chapter information associated to $Req_\bullet$.

The pretreatment is detailed as follows:

Algorithm 1 Create new requirements file

Require: $i, j \geq 0, ReqFile, Req_\bullet.chapter(), RegFile.length()$
Ensure: $NewReqFile$
  for $i = 0$; $i < RegFile.length()$; i++ **do**
    for $j = i + 1$; $j < RegFile.length() - 1$; j++ **do**
      if $Req_i = Req_j and Req_i.chapter() \neq Req_j.chapter()$ **then**
        $NewReqFile \leftarrow Req_i$
      end if
    end for
  end for

We remind that this algorithm do not aim to guarantee uniqueness of the requirements, but remove identical requirements belonging to the different chapters. So, we may still have the same requirements in the same chapters in the new requirement file.

C. POS tagging

For the preprocessing step, we use the Part-Of-Speech (POS) tagging and Noun chunking from SpaCy[4] as a popular tool in natural language processing field. SpaCy is a free open-source library featuring state-of-the-art speed and accuracy and a powerful Python API.

After applying this tagging approach, we proceed to detect technical terms according to some combination of tags. According to our RE expert, technical business terms are often expressed in open or hyphenated compound words (e.g. *high speed*, *safety-critical*) and we observe that they are always parts of a noun chunk[5]. For this paper, we first extracted all noun chunks from our Corpus1, then observed the syntactic patterns inside noun chunks referring to POStags, obtained by SpaCy. The most used 13 combination patterns[6] in business terms are selected and validated in collaboration with our RE expert: for example, noun-noun (e.g. *runway overrun*), adjective-noun (e.g. *normal mode*), proper noun-noun (e.g.*BSP data*), adjective-adjective-noun (e.g. *amber visual indication*), noun-noun-noun (e.g. *output voltage value*).

Once the business terms are detected according to the previous pattern, they will be integrated to the main document as one word instead of several words. For example, "runway overrun" will be written in the new file as "runway_overrun".

D. k-means algorithm

Principle: The k-means algorithm is used to partition a given set of observations into a predefined amount of k clusters. The algorithm as described by [24] starts with a

[4]https://spacy.io/

[5]A noun chunk is a noun plus the words describing the noun.

[6]We give here all patterns used in this work: noun-noun, adjective-noun, proper noun-noun, adjective-adjective-noun, noun-noun-noun, number-noun, proper noun-proper noun, proper noun-punctuation-proper noun, number-proper noun, proper noun-proper noun-proper noun, proper noun-number-noun, proper noun-noun-noun, adjective-noun-noun.

random set of k center-points (μ). During each update step, all observations x are assigned to their nearest center-point (see Equation 1). In the standard algorithm, only one assignment to one center is possible. If multiple centers have the same distance to the observation, a random one would be chosen.

$$S_i^{(t)} = \left\{ x_p : \left\| x_p - \mu_i^{(t)} \right\|^2 \leq \left\| x_p - \mu_j^{(t)} \right\|^2 \ \forall j, 1 \leq j \leq k \right\} \tag{1}$$

Afterwards, the center-points are repositioned by calculating the centroid of the assigned observations to the respective center-points.

$$\mu_i^{(t+1)} = \frac{1}{|S_i^{(t)}|} \sum_{x_j \in S_i^{(t)}} x_j \tag{2}$$

The update process reoccurs until all observations remain at the assigned center-points and therefore the center-points would not be updated anymore.

This means that the k-means algorithm tries to optimize the objective function 3. As there is only a finite number of possible assignments for the amount of centroids and observations available and each iteration has to result in better solution, the algorithm always ends in a local minimum.

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} ||x_n - \mu_k||^2 \tag{3}$$

r_{nk} =1 if $x_n \in S_k$, 0 otherwise.

Example set: In order to illustrate the k-means algorithm, we give an example of clustered requirements. Let's assume that we have a list of 11 requirements as follows :

1) *If the estimated stopping distance is greater than the available distance and the system is engaged, the system shall detect a vehicle traveling with no throttle.*
2) *If a vehicle traveling with no throttle is detected and the system is engaged, the system shall decelerate the vehicle.*
3) *Digital state : (switch conversion) states are calculated using input voltage.*
4) *Digital state : (switch conversion) states are calculated for frequency external inputs using input voltage.*
5) *Application data can be freely defined in remaining space by the customer.*
6) *Application data can be freely defined by the customer.*
7) *Application data can be freely defined in the process document by the customer.*
8) *The approval shall be stamped in conformance with 6.7.4, and recorded.*
9) *The approval shall be marked in conformance with 6.7.4, and recorded.*
10) *The system shall withstand acceleration up to 150m.s-2.*
11) *In climb-out phase, the system shall withstand an acceleration up to 150m.s-2.*

k-means algorithm will cluster this list into a set of k fixed number of clusters. Let's assume that k=5, the result of the algorithm will be as follows:

- Cluster 1:
  1) *If the estimated stopping distance is greater than the available distance and the system is engaged, the system shall detect a vehicle traveling with no throttle.*
  2) *If a vehicle traveling with no throttle is detected and the system is engaged, the system shall decelerate the vehicle.*
- Cluster 2:
  1) *Digital state : (switch conversion) states are calculated using input voltage.*
  2) *Digital state : (switch conversion) states are calculated for frequency external inputs using input voltage.*
- Cluster 3:
  1) *Application data can be freely defined in remaining space by the customer.*
  2) *Application data can be freely defined by the customer.*
  3) *Application data can be freely defined in the process document by the customer.*
- Cluster 4:
  1) *The approval shall be stamped in conformance with 6.7.4, and recorded.*
  2) *The approval shall be marked in conformance with 6.7.4, and recorded*
- Cluster 5:
  1) *The system shall withstand acceleration up to 150m.s-2*
  2) *In climb-out phase, the system shall withstand an acceleration up to 150m.s-2*

The algorithm cluster the requirements according to their similarities. So, each cluster contains the most similar requirements. A cluster may contain one or more requirements depending on the dataset. We have shown in this example only the case of 2 requirements per cluster.

Challenges of k-means algorithm: k-means algorithm is a very popular approach due to its efficiency. However, it needs a predefined value of K as an input, which is the main issue about using this algorithm. Some researchers focus on this issue and present solutions based on the graphical (e.g. elbow approach[7], silhouette[8] and Inertia[9]) or numerical value (e.g. statistic gap [25]). We use in this paper the following solutions to calculate the value of k:

- Inertia: calculated as the sum of squared distance for each point to its closest centroid, i.e., its assigned cluster. It can be recognized as a measure of how internally coherent clusters are. The k-means algorithm aims to choose centroids that minimize the inertia.
- Statistic gap: calculates a goodness of clustering measure. The statistic gap standardizes the graph of $log(W_k)$, where W_k is the within-cluster dispersion, by comparing it to its expectation under an appropriate null reference distribution of the data [25].

IV. Validation approach

Since we use an unsupervised clustering approach, we do not have any ground truth about the redundancy and/or the inconsistency of the requirements. So, we give the results related to the best value of k to our RE expert in order that the expert evaluates the relevance of the generated clusters. A cluster may contain one or more requirement(s).

[7] $https://en.wikipedia.org/wiki/Elbow\_method\_(clustering)$
[8] $https://en.wikipedia.org/wiki/Silhouette\_(clustering)$
[9] $http://scikit-learn.org/stable/modules/clustering.html$

For a given k value, the validation is done according to two methods:

- "Strict" validation (SV): we assume that a relevant cluster contains 100% correct requirements (fully redundant or incoherent requirements), which means that we discard clusters with partially relevant requirements. Also, we consider only clusters with more than one requirement. For example, let's assume that we have cluster1 with 4 requirements, Cluster1={requirement1, requirement2, requirement3, requirement4}. Cluster1 is considered relevant only if all the 4 requirements are redundant/incoherent. Otherwise, it is considered as non relevant.
- "Average" validation (AV): we calculate the average of relevant requirements per cluster.

$$AV_k = \frac{\sum_{i=1}^{k} precision(c_i)}{k'} \tag{4}$$

where AV_k is the average validation for a given value of k. k is the number of clusters. k' is the number of clusters which their number of requirements is >1. The precision a cluster c_i is defined as:

$$precision(c_i) = \frac{NumberOfRelevantRequirements}{TotalNumberOfRequirements} \tag{5}$$

The *NumberOfRelevantRequirements* is the sum of all relevant (redundant) requirements within a cluster c_i. For example, let's assume that we have cluster with 4 requirements Cluster1={requirement1, requirement2, requirement3, requirement4} and only requirement1 and requirement2 are redundant/incoherent and the other two requirements are not redundant/incoherent. Then this cluster is 50% relevant.

V. Experimentation Results

In this section, we present in Section V-A the datasets used in the experiment. In Section V-B we explain how to determinate the best k value used in our clustering approach. In Section V-C we present the result of our approach.

A. Datasets

In order to test our approach, we extracted list of requirements from 2 industrial specifications. For confidentiality issues, we are not allowed to reveal the identity of the companies. The main features considered to validate our datasets are: 1) texts following various kinds of business style and format guidelines imposed by companies, 2) texts coming from various industrial areas: aeronautic, automobile, spatial. Theses datasets enable us to analyze different types of redundancy and inconsistency in terms of frequency and context. We present characteristics of these datasets (written in English) as follows:

- Corpus1: dataset that contains ∼360 pages and ∼913 requirements with no a priori information of redundancy and inconsistency,
- Corpus2: dataset that contains ∼111 pages and ∼326 requirements with no a priori information of redundancy and inconsistency.

These datasets generally have a low rate of redundancy and inconsistency according to RE expert. However, detecting this problem is very significant in this case of industrial requirements.

B. Determining the best number of K

From the datasets already detailed in Section V-A, we extract a new requirement file for each dataset by applying the algorithm explained in Section III-B and then the POS tagging explained in Section III-C.

The new number of requirements of each new requirement file are mentioned in Table I. For Corpus1, we will analyze 902 requirements (instead of 913) and for Corpus2, we will analyze 280 requirements (instead of 326). The pretreatment steps deduces the size of the datasets but they still significantly important compared to the rate of redundancy/inconsistency a priori existent.

We apply then k-means algorithm on each new requirement file using the Euclidean distance as similarity metric since we had best results comparing to other similarity metrics such as TF-IDF, JACCARD, Correlation and Dice according to our expert.

We determinate in this Section the best number of K by calculating the inertia (the clusters errors) of new requirement file of Corpus1 and Corpus 2 in Figure 2 and Figure 3 respectively. Since the inertia is reflecting the clusters errors, we should choose the minimum value to determine the value of k.

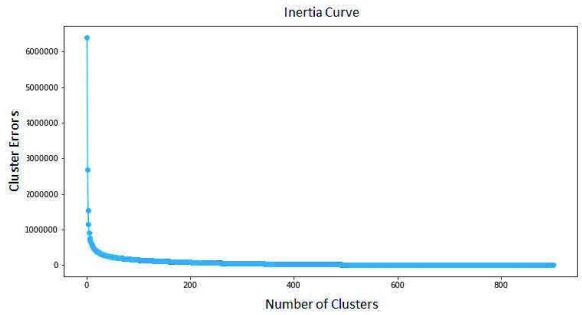

Fig. 2. Inertia curve for Corpus1 dataset

According to Figure 2 and Figure 3, determining visually the number of k cannot always be unambiguously identified. In order to leverage this ambiguity, we choose to apply the statistic gap approach which allows to obtain a numerical value reflecting the coherence of the clusters. We apply the statistic gap to our datasets and the best number of k for Corpus1 is 38 and for Corpus2 is 42. These values are coherent according to the two previous figures, since the curves become almost stable starting from these values.

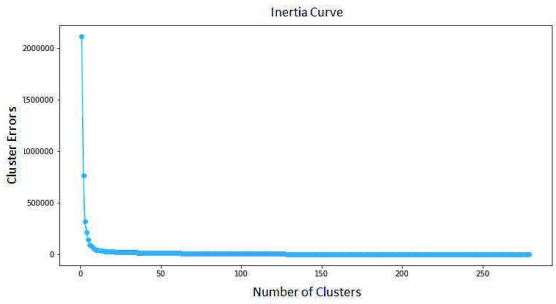

Fig. 3. Inertia curve for Corpus2 dataset

C. Classification results of our approach

We apply the k-means algorithm on files already preprocessed and POS tagged. Table I shows the new number of requirements after applying the pretreatment step, and summarizes the different results obtained on Corpus1 and Corpus2 datasets. In Table I, we present different best values of k according to the static gap previously detailed and also a new value related to a percentage of errors. This latter aims to fix a value according to the ”usual” percentage of errors in requirements files and it is provided by our RE expert. In our case, the RE expert estimates this value about 20%.

We note that in Corpus1, among 721 clusters, only 115 clusters contain more than one requirement. In Corpus2, among 224 clusters, only 25 clusters contain more than one requirement. As explained before, the clusters with only one requirement are discarded from the validation calculation process.

According to the best value of k, we can clearly see that the statistic gap is not appropriate to Corpus1 and Corpus2 since we did not had many relevant clusters. However, the best k value based on RE expert, provides much better results in both SV and AV. In these industrial documents, the statistic gap is not appropriate to the domain and can not be used in large corpus. Redundancy and inconsistency are detected in Corpus1 and Corpus2 with a better relevancy for k value based on RE expert. So, a better number of relevant clusters is found compared to the statistic gap.

VI. Discussion

The k-means results are given to our RE expert to judge the best value of k from his/her own domain-based expertise. We found a difference between the generated k value (according to the statistic gap) and the best value according to our expert. After several experiments, it seems that the statistic gap generates in most cases very low number of k value independently of the corpus size. For example, in Corpus 1, when we analyze clusters of k=38, value obtained by the statistic gap, among 38 clusters, 24 clusters contain more than 10 requirements in a cluster and there was also found clusters containing more than 100 requirements. Our RE expert observes that almost of them do not show the similarity between them.

Otherwise, the decision based on the errors rate (20%) as best k value shows significantly improved results. We also experimented on Corpus1 and Corpus2 varying potential errors rate from 15% to 30%. The best SV and AV obtained are only 49.72% and 52%, respectively.

VII. Conclusion

In this paper, we proposed an automatic approach for redundancy and inconsistency detection in requirements engineering context that will be integrated to SEMIOS software. This approach is based on an artificial intelligence technique and more precisely unsupervised machine learning algorithm, k-means. This approach is tested on real industrial datasets with different characteristics of redundancy and/or inconsistency. Also, we introduced the preprocessing step based on the NLP techniques in order to see the impact of this latter to the k-means results. We used Part-Of-Speech (POS) tagging and noun chunking in order to detect technical business terms associated to the requirements documents that we analyze.

k-means algorithm is tested according to the best k value generated by the statistic gap method and also by a value defined by our RE expert. The statistic gap did not provide relevant results and it is not appropriate to our two corpus. According to the best k value provided by our expert, k-means provides very relevant results by generating only clusters (with more than one requirement) with relevant information.

Our approach is applicable to every textual requirements. So, this work is domain-independent and may be applied to every type of requirements written in Natural Language and without any a priori knowledge.

Our experiments on the clustering approach applied in this work show encouraging results as a first step for detecting redundancy and inconsistency, and some directions for the future works. First, we plan to investigate on how to automatically obtain the optimal value of k for specifications documents. In parallel, we will evaluate more diverse corpus and analyze clustering results applying the best value of k based on the errors rate. It will allow us to confirm the accuracy of applying 20% as usual errors rate. Second, our expert only evaluated the content of the proposed clusters. We plan to evaluate whether the requirements placed by k-means in different clusters are really redundant/inconsistent. Third, in order to improve the clustering results, we will introduce semantic approach on lexical level using Word2Vec, for example. Last, even with high quality results for Corpus1 and Corpus2, we are not able yet to differentiate redundancy or inconsistency in very similar clusters. To overcome this shortcoming, we plan to apply another clustering approach on similar clusters. This new clustering will be based on semantic features.

Acknowledgements

This work is financially supported by the Occitanie region of France in the framework of CLE (Contrat de recherche Laboratoires-Entreprises)-ELENAA (des Exigences

TABLE I
RESULTS: NEW NB. OF REQUIREMENTS, BEST VALUE OF K, VALIDATION RESULTS AND THE ASSOCIATED NUMBER OF RELEVANT CLUSTERS FOR EACH DATASET

Dataset	New nb. of req.	Best value of K		SV (Nb. of relevant clusters)	AV (Nb. of relevant clusters)
		Based on statistic gap	Based on RE expert		
Corpus1	902	38	–	11.11% (4)	26.17% (31)
		–	721	48.69% (56)	51.31% (64)
Corpus2	280	42	–	8.33% (2)	19.54% (11)
		–	224	76% (19)	76% (19)

en LanguEs Naturelles à leurs Analyses Automatiques) project. We would like to thank Audrey Speronel from Prometil for her contribution to generate adapted files from Semios software.

REFERENCES

[1] K. J. Elizabeth Hull and J. Dick, *Requirements Engineering*. Springer-Verlag London, 2011.

[2] E. K. Daniel M. Berry and M. M. Krieger, *From Contract Drafting to Software Specification: Linguistic Sources of Ambiguity*, 2003.

[3] D. Galin, *Software Quality Assurance: From Theory to Implementation*, 2003.

[4] P. Bourque, *Guide to the Software Engineering Body of Knowledge (SWEBOK Guide*, 2004.

[5] A. A. Alshazly, A. M. Elfatatry, and M. S. Abougabal, "Detecting defects in software requirements specification," *Alexandria Engineering Journal*, vol. 53, no. 3, pp. 513 – 527, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1110016814000568

[6] R. L. Glas, *Facts and Fallacies of Software Engineering*. Addison-Wesley Professional, 2002.

[7] B. D. B. H. R. L. Jonette M. Stecklein, Jim Dabney and G. Moroney, "Error cost escalation through the project life cycle," in *Proceedings of the 14th Annual International Symposium*, Toulouse, France, 2004.

[8] J. Winkler and A. Vogelsang, "Automatic classification of requirements based on convolutional neural networks," in *2016 IEEE 24th International Requirements Engineering Conference Workshops (REW)*, Sept 2016, pp. 39–45.

[9] E. Knauss, D. Damian, G. Poo-Caamao, and J. Cleland-Huang, "Detecting and classifying patterns of requirements clarifications," in *2012 20th IEEE International Requirements Engineering Conference (RE)*, Sept 2012, pp. 251–260.

[10] D. Ott, *Automatic Requirement Categorization of Large Natural Language Specifications at Mercedes-Benz for Review Improvements*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 50–64. [Online]. Available: https://doi.org/10.1007/978-3-642-37422-7_4

[11] A. K. Jain, "Data clustering: 50 years beyond k-means," *Pattern Recognition Letters*, vol. 31, no. 8, pp. 651 – 666, 2010, award winning papers from the 19th International Conference on Pattern Recognition (ICPR). [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167865509002323

[12] J. Allan, V. Lavrenko, D. Malin, and R. Swan, "Detections, bounds, and timelines: Umass and tdt-3," in *Proceedings of Topic Detection and Tracking Workshop (TDT-3)*. Vienna, VA, 2000, pp. 167–174.

[13] P. F. Brown, P. V. deSouza, R. L. Mercer, T. J. Watson, V. J. D. Pietra, and J. C. Lai, "Class-based n-gram models of natural language," *Computational Linguistics*, vol. 18, no. 4, pp. 467–480, 1992. [Online]. Available: http://www.aclweb.org/anthology/J92-4003

[14] E. Juergens, F. Deissenboeck, M. Feilkas, B. Hummel, B. Schaetz, S. Wagner, C. Domann, and J. Streit, "Can clone detection support quality assessments of requirements specifications?" in *Proceedings of the 32Nd ACM/IEEE International Conference on Software Engineering - Volume 2*, ser. ICSE '10. New York, NY, USA: ACM, 2010, pp. 79–88. [Online]. Available: http://doi.acm.org/10.1145/1810295.1810308

[15] D. Falessi, G. Cantone, and G. Canfora, "Empirical principles and an industrial case study in retrieving equivalent requirements via natural language processing techniques," *IEEE Trans. Softw. Eng.*, vol. 39, no. 1, pp. 18–44, Jan. 2013. [Online]. Available: http://dx.doi.org/10.1109/TSE.2011.122

[16] A. Rago, C. Marcos, and J. A. Diaz-Pace, "Identifying duplicate functionality in textual use cases by aligning semantic actions," *Software & Systems Modeling*, vol. 15, no. 2, pp. 579–603, May 2016. [Online]. Available: https://doi.org/10.1007/s10270-014-0431-3

[17] P. Belsis, A. Koutoumanos, and C. Sgouropoulou, "Pburc: a patterns-based, unsupervised requirements clustering framework for distributed agile software development," *Requirements Engineering*, vol. 19, no. 2, pp. 213–225, Jun 2014. [Online]. Available: https://doi.org/10.1007/s00766-013-0172-9

[18] D. Dermeval, J. Vilela, I. I. Bittencourt, J. Castro, S. Isotani, P. Brito, and A. Silva, "Applications of ontologies in requirements engineering: a systematic review of the literature," *Requirements Engineering*, vol. 21, no. 4, pp. 405–437, Nov 2016. [Online]. Available: https://doi.org/10.1007/s00766-015-0222-6

[19] A. O. J. Sabriye and W. M. N. W. Zainon, "A framework for detecting ambiguity in software requirement specification," in *2017 8th International Conference on Information Technology (ICIT)*, May 2017, pp. 209–213.

[20] U. S. Shah and D. C. Jinwala, "Resolving ambiguities in natural language software requirements: A comprehensive survey," *SIGSOFT Softw. Eng. Notes*, vol. 40, no. 5, pp. 1–7, Sep. 2015. [Online]. Available: http://doi.acm.org/10.1145/2815021.2815032

[21] Z. S. H. Abad, O. Karras, P. Ghazi, M. Glinz, G. Ruhe, and K. Schneider, "What works better? a study of classifying requirements," in *2017 IEEE 25th International Requirements Engineering Conference (RE)*, Sept 2017, pp. 496–501.

[22] D. Klein and C. D. Manning, "Accurate unlexicalized parsing," in *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, ser. ACL '03. Stroudsburg, PA, USA: Association for Computational Linguistics, 2003, pp. 423–430. [Online]. Available: https://doi.org/10.3115/1075096.1075150

[23] X. Fu, E. Ch'ng, U. Aickelin, and S. See, "Crnn: A joint neural network for redundancy detection," in *2017 IEEE International Conference on Smart Computing (SMARTCOMP)*, May 2017, pp. 1–8.

[24] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability - Vol. 1*, L. M. Le Cam and J. Neyman, Eds. University of California Press, Berkeley, CA, USA, 1967, pp. 281–297.

[25] M. Mohajer, K.-H. Englmeier, and V. J. Schmid, "A comparison of gap statistic definitions with and without logarithm function," 2010. [Online]. Available: http://nbn-resolving.de/urn/resolver.pl?urn=nbn:de:bvb:19-epub-11920-3