Using k-means for redundancy and inconsistency detection: application to industrial requirements

Manel Mezghani, Juyeon Kang Choi, Florence Sèdes

Using k-Means for Redundancy and Inconsistency Detection: Application to Industrial Requirements

Manel Mezghani[1,2(✉)], Juyeon Kang[2], and Florence Sèdes[1]

[1] IRIT, University of Toulouse, CNRS, INPT, UPS, UT1, UT2J, Toulouse, France
mezghani.manel@gmail.com, florence.sedes@irit.fr
[2] Prometil, 52 Rue Jacques Babinet, 31100 Toulouse, France
j.kang@semiosapp.com

Abstract. Requirements are usually "hand-written" and suffers from several problems like redundancy and inconsistency. These problems 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. We propose in this paper to automatically handle redundancy and inconsistency issues in a classification approach. The main contribution of this paper is the use of k-means algorithm for redundancy and inconsistency detection in a new context, which is Requirements Engineering context. Also, we introduce a preprocessing step based on the Natural Language Processing techniques in order to see the impact of this latter to the k-means results. We use Part-Of-Speech (POS) tagging and noun chunking in order to detect technical business terms associated with 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 preprocessing.

Keywords: K-means · Requirements engineering · POS tagging
Redundancy · Inconsistency

1 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

[1] http://www.standishgroup.com.
[2] They studied 50,000 projects around the world, ranging from tiny enhancements to massive systems re-engineering implementations.

https://doi.org/10.1007/978-3-319-91947-8_52

Requirements Engineering (RE) and 70% of them come from the difficulties of understanding implicit requirements. All these errors do not lead to the 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 errors were discovered during the requirements phase of a project [5,6].

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 [1]. 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 focus in this paper on two critical issues in writing high quality requirements that can generate fatal errors in a product development stage: redundancy and inconsistency. We tackle these problems in terms of similarity between the requirements since more than two similar requirements can be classified as redundant or inconsistent. 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 (i.e. [7]) Naives Bayes classifier [8] and text classification algorithms [9]. Data classification approaches could be data clustering through algorithm such as K-means. This latter is studied in different contexts due to its efficiency [10]. However, in Requirements Engineering (RE) 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 RE context. Also, we introduce a preprocessing step based on the 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 with the requirements documents.

This paper is structured as follows: Sect. 2 presents related works on the redundancy and inconsistency detection, preprocessing approaches and k-means. Section 3 presents our clustering approach and the associated results. In Sect. 4, we discuss the obtained results and in Sect. 5, conclude and give some future research directions.

2 Related Works

In this section, we first present related works associated with 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.

2.1 Redundancy and Inconsistency Detection

Researches on redundancy detection began by traditional bag-of-words (BOW), TF-IDF frequency matrix, and n-gram language modeling [11]. Then, researchers like Juergens et al. [12] use ConQAT to identity copy-and-paste reuses in requirements specifications. Falessi *et al.* [13] 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.* [14] extend the work presented in [13] 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 [15] by proposing a framework of a patterns-based k-means requirements clustering, 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. Frenay et al. [16] present a survey of techniques treating data quality such as inconsistency. They present different machine learning approaches and their impact on the results. Dermeval et al. [17] present a survey about how using ontologies in RE activities both in industry and academy is beneficial, specially for reducing ambiguity, inconsistency and incompleteness of requirements.

2.2 Preprocessing

Some researches introduce preprocessing steps in requirements analysis context. According to [18], 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 POS tagger [19], (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 [20] by performing standard NLP techniques such as removing English stop words and striping off the newsgroup related meta-data (including noisy headers, footers and quotes) and normalized bag-of-words (BOW).

2.3 K-Means

K-means clustering is a 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. 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, silhouette and Inertia) or numerical value (e.g. statistic gap). The 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 [21].

Recently, some studies have introduced k-means in requirements classification tasks. Notably, [18] 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) in order to classify requirements into functional and non-functional requirements.

3 Clustering Approach

We apply the k-means algorithm already detailed in Sect. 2.3 on datasets detailed below. We present in this section the validation approach and the comparative results obtained with and without the preprocessing step.

3.1 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 domain expert in order that the expert evaluates the relevance of the generated clusters. A cluster may contains one or more requirement(s). 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 requirements), which means that we discard clusters with partially relevant requirements. We consider only clusters with more than one requirement.
- "Average" validation (AV): we calculate the average of relevant requirements per cluster.

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

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. i $\in \{1, k\}$ is the value of k and $precision(c_i)$ is defined as:

$$precision(c_i) = \frac{Number\,Of\,Relevant\,Requirements}{Total\,Number\,Of\,Requirements} \tag{2}$$

3.2 Classification Results

In this section, we present the obtained results of applying the k-means algorithm. In order to test our approach, we extracted requirements from 22 industrial specifications (∼2000 pages). From this, we constructed three different

datasets (corpus1, corpus2 and corpus3) explained below. 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, telecommunication, finance, energy. 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 38 requirements fully redundant according to our expert,
- Corpus2: dataset that contains 42 requirements fully inconsistent according to our expert,
- Corpus3: dataset that contains 337 requirements randomly chosen with no a priori information of redundancy and inconsistency,

We choose to apply the statistic gap approach which allows to obtain a numerical value reflecting the coherence of the clusters. Table 1 shows, for each dataset, the number of requirements, the best value of k (according to the statistic gap), the results of the "strict" validation (SV) and the associated number of relevant clusters, and the results of the "average" validation (AV) and the associated number of relevant clusters.

Table 1. Results: best value of K, validation results and the associated number of relevant clusters for each dataset

Dataset	Best value of K	SV (Nb. of relevant clusters)	AV (Nb. of relevant clusters)
Corpus1	30	100% (8)	100% (8)
Corpus2	17	100% (15)	100% (15)
Corpus3	26	22% (4)	30.96% (18)

In Corpus1 and Corpus2 dataset, the result is very interesting since we know the characteristics of these datasets, which are fully redundant/fully inconsistent and then the clustering approach is appropriate to this kind of datasets. In Corpus3 dataset, we have less good results, but this is explained by the nature of this dataset which is not fully redundant or inconsistent. So, we do not know in reality, how much redundancy or inconsistency has this dataset. From this dataset, very close to a real industrial dataset, we can conclude that the clustering approach has detected redundancy/inconsistency, but we do not know if it had detected the whole redundancy/inconsistency information.

3.3 Classification Results with the Preprocessing Step

For the preprocessing step, we use the POS tagging and Noun chunking from spaCy[3] as a popular tool in natural language processing field. spaCy is a free

[3] https://spacy.io/.

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 part of a noun chunk[4]. In 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 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*), etc. So, we apply the k-means algorithm on the same datasets with the identified technical terms in order to see the impact of this preprocessing on the results. Table 2 summarizes the different results obtained from the same experiments presented in Sect. 3.1.

Table 2. Results with preprocessing: best value of K, validation results and the associated number of relevant clusters for each dataset

Dataset	Best value of K	SV (Nb. of relevant clusters)	AV (Nb. of relevant clusters)
Corpus1	28	100% (10)	100% (10)
Corpus2	24	92.85% (13)	92.85% (13)
Corpus3	36	22.22% (6)	39.20% (27)

In Corpus1 dataset, the POS tagging has shown its efficiency to improve redundancy/inconsistency detection results with two more relevant clusters than the results without preprocessing. In Corpus2 dataset, we obtain two less relevant clusters than the clustering without preprocessing. In this case, the preprocessing has shown its inefficiency to improve inconsistency detection results. In Corpus3 dataset, we have the same relevant value of the strict validation comparing to the Table 1. However, the number of relevant clusters is higher. For the average validation, we clearly see an improvement of the percentage of relevant clusters and also the total number of relevant clusters. The preprocessing has improved the rate of the redundancy/inconsistency detection.

4 Discussion

The k-means results are given to our domain 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. For the results without preprocessing, the results are as follows: for to Corpus1, our expert assume that 23 (instead of 30) is the best value of k with 100% of relevance (for SV) and with 13 relevant clusters (instead of 8).

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

For Corpus2, our expert assume that 18 (instead of 17) is the best value of k with 100% of relevance (for SV) and with 16 relevant clusters (instead of 15). For the results with preprocessing, the results are as follows: for to Corpus1, our expert assume that 23 (instead of 28) is the best value of k with 100% of relevance (for SV) and with 14 relevant clusters (instead of 13). For Corpus2, our expert assume that 25 (instead of 24) is the best value of k with 100% of relevance (for SV) and with 15 relevant clusters (instead of 13). In our case, the statistic gap did not found the best k value for our domain.

5 Conclusion

In this paper, we used k-means algorithm for redundancy and inconsistency detection in RE context. We used POS tagging and noun chunking in order to detect technical business terms associated to the requirements documents that we analyze. This approach is tested on real industrial datasets with different characteristics of redundancy and/or inconsistency. According to Corpus1 (redundant) and Corpus2 (inconsistent), k-means provides very relevant results. Preprocessing has improved the rate of redundancy detection but not the rate of the inconsistency detection. According to Corpus3, the results show the importance of the preprocessing step to improve the clustering results in terms of precision and also the number of detected clusters. Even with high quality results on Corpus1 and Corpus2, we are not able yet to differentiate redundancy or inconsistency in very similar clusters in Corpus3. So, we plan to apply another clustering approach based on semantic features. After improvements, this work will be integrated in the industrial tool: *Semios for requirements*[5].

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 en LanguEs Naturelles à leurs Analyses Automatiques) project.

References

1. Hull, E., Jackson, K., Dick, J.: Requirements Engineering. Springer-Verlag, London (2011)
2. Daniel, M., Berry, E.K., Krieger, M.M.: From Contract Drafting to Software Specification: Linguistic Sources of Ambiguity (2003)
3. Galin, D.: Software Quality Assurance: From Theory to Implementation (2003)
4. Bourque, P.: Guide to the Software Engineering Body of Knowledge (SWEBOK Guide) (2004)
5. Glas, R.L.: Facts and Fallacies of Software Engineering. Addison-Wesley Professional, Reading (2002)
6. Stecklein, J.M., Dabney, J., Dick, B., Haskins, B., Lovell, R., Moroney, G.: Error cost escalation through the project life cycle. In: Proceedings of the 14th Annual International Symposium, Toulouse, France (2004)

[5] http://www.semiosapp.com/index.php?lang=en.

7. Winkler, J., Vogelsang, A.: Automatic classification of requirements based on convolutional neural networks. In: 2016 IEEE 24th International Requirements Engineering Conference Workshops (REW), pp. 39–45, September 2016
8. Knauss, E., Damian, D., Poo-Caamao, G., Cleland-Huang, J.: Detecting and classifying patterns of requirements clarifications. In: 2012 20th IEEE International Requirements Engineering Conference (RE), pp. 251–260, September 2012
9. Ott, D.: Automatic requirement categorization of large natural language specifications at mercedes-benz for review improvements. In: Doerr, J., Opdahl, A.L. (eds.) REFSQ 2013. LNCS, vol. 7830, pp. 50–64. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-37422-7_4
10. Jain, A.K.: Data clustering: 50 years beyond k-means. Pattern Recogn. Lett. **31**(8), 651–666 (2010). Award winning papers from the 19th International Conference on Pattern Recognition (ICPR)
11. Allan, J., Lavrenko, V., Malin, D., Swan, R.: Detections, bounds, and timelines: umass and tdt-3. In: Proceedings of Topic Detection and Tracking Workshop (TDT-3), Vienna, VA, pp. 167–174 (2000)
12. Juergens, E., Deissenboeck, F., Feilkas, M., Hummel, B., Schaetz, B., Wagner, S., Domann, C., Streit, J.: Can clone detection support quality assessments of requirements specifications? In: Proceedings of the 32Nd ACM/IEEE International Conference on Software Engineering, vol. 2. ICSE 2010, New York, USA, pp. 79–88. ACM (2010)
13. Falessi, D., Cantone, G., Canfora, G.: Empirical principles and an industrial case study in retrieving equivalent requirements via natural language processing techniques. IEEE Trans. Softw. Eng. **39**(1), 18–44 (2013)
14. Rago, A., Marcos, C., Diaz-Pace, J.A.: Identifying duplicate functionality in textual use cases by aligning semantic actions. Softw. Syst. Model. **15**(2), 579–603 (2016)
15. Belsis, P., Koutoumanos, A., Sgouropoulou, C.: Pburc: a patterns-based, unsupervised requirements clustering framework for distributed agile software development. Requir. Eng. **19**(2), 213–225 (2014)
16. Frenay, B., Verleysen, M.: Classification in the presence of label noise: a survey. IEEE Trans. Neural Netw. Learn. Syst. **25**(5), 845–869 (2014)
17. Dermeval, D., Vilela, J., Bittencourt, I.I., Castro, J., Isotani, S., Brito, P., Silva, A.: Applications of ontologies in requirements engineering: a systematic review of the literature. Requir. Eng. **21**(4), 405–437 (2016)
18. Abad, Z.S.H., Karras, O., Ghazi, P., Glinz, M., Ruhe, G., Schneider, K.: What works better? a study of classifying requirements. In: 2017 IEEE 25th International Requirements Engineering Conference (RE), pp. 496–501, September 2017
19. Klein, D., Manning, C.D.: Accurate unlexicalized parsing. In: Proceedings of the 41st Annual Meeting on Association for Computational Linguistics, vol. 1. ACL 2003, Stroudsburg, PA, USA, pp. 423–430. Association for Computational Linguistics (2003)
20. Fu, X., Ch'ng, E., Aickelin, U., See, S.: CRNN: a joint neural network for redundancy detection. In: 2017 IEEE International Conference on Smart Computing (SMARTCOMP), pp. 1–8, May 2017
21. Mohajer, M., Englmeier, K.H., Schmid, V.J.: A comparison of gap statistic definitions with and without logarithm function (2010)