A clustering approach for detecting defects in technical documents

Manel Mezghani, Juyeon Kang Choi, Florence Sèdes

A clustering approach for detecting defects in technical documents

Manel Mezghani[1], Juyeon Kang[1], Florence Sèdes[2]

[1] Prometil, 52 Rue Jacques Babinet, 31100 Toulouse, France
{m.mezghanni, j.kang}@prometil.com
[2] 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 pre-processing step based on the Natural Language Processing (NLP) techniques to see 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. We experiment this approach on real industrial datasets. The results show the efficiency of the k-means clustering algorithm especially with the pre-processing.

1. Introduction

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 (Hull, 2011). Writing clearly all required elements without ambiguities (Berry, 2003) in the specifications is an essential task before passing to the development stage (Galin2003, Bourque2004). According to the 2015 Chaos report[1] by the Standish Group, only 29% of projects were successful[2], 50% of the challenged projects are related to the defects (Alshazly, 2014) from the Requirements Engineering (RE) and 70% of them come from the difficulties of understanding implicit requirements. All these defects do not lead to the failure but generate useless information. It is well known that the costs to fix defects increase much more after that the product is built than it would if the requirements defects were discovered during the requirements phase of a project (Glas, 2002; Stecklein, 2004).

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 many language errors (lexical, grammatical, style), and a lack of overall concordance, or redundancy and inconsistency in the underlying meaning of requirements. 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 defects in a product development stage: redundancy and inconsistency. We tackle these problems in terms of similarity between requirements since more than two similar requirements can be classified as redundant or inconsistent requirements.

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

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 (Winkler, 2016), Naives Bayes classifier (Knauss, 2012), text classification algorithms (Ott, 2013). Data classification approaches could be data clustering through algorithm such as K-means. This latter is studied in different contexts due to its efficiency (Jain, 2010). 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 pre-processing step based on the Natural Language Processing (NLP) techniques 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 2, we present related works on the redundancy and inconsistency detection through artificial intelligence approach and especially k-means technique. In section 3, we present the datasets used for the experimental part. In section 4, we present our clustering approach. In section 5, we discuss the associated results. In section 6, we conclude and give some future research directions.

2. 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 pre-processing 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 modelling (Allan, 2000) (Brown, 1992). Then, researchers like Juergens et al. (Juergens, 2010) use ConQAT to identity copy-and-paste reuses in requirements specifications. Falessi et al. (Falessi, 2013) 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. (Rago, 2016) extend the work presented in (Falessi, 2013) 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 (Belsis, 2014) 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., (Dermeval, 2016) 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.

2.2 Pre-processing

Some researches introduce pre-processing steps in requirements analysis context. According to (Abad, 2017), the pre-processing 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 (Klein, 2003), 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 pre-processing approach improved the performance of an existing classification method.

Pre-processing data for redundancy detection is used in (Fu, 2017) 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). The Joint Neural Network for redundancy detection approach in (Fu, 2017) also uses normalized bag-of-words (BOW) as a pre-processing approach. The normalized BOW generates a global unigram based dictionary mapping. With the presence of the uni-gram indexer, the authors could readily remove low frequency terms and lengthy snippets.

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 (k number of groups).

Classifying requirements is an important task in requirements engineering. Recently, some studies introduce k-means in requirements classification tasks. Notably, (Abad, 2017) applies different approaches such as i) topic modelling 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).

3. Datasets

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. These 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,

The expert in this work means requirements engineer with more than 15 years of experiences (industrial and academic).

4. Clustering Approach

In this section, we present the basics of k-means clustering algorithm and the results of our approach. We also analyze the impact of the pre-processing step on the results.

4.1 K-means algorithm

The k-means algorithm is used to partition a given set of observations into a predefined amount of *k* clusters. 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, silhouette and Inertia or numerical value (e.g. statistic gap (Mohajer, 2010)). 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.
- 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 (Mohajer, 2010).

4.2 Determining the best number of K

We apply the k-means algorithm on the datasets already detailed in section 3. To choose the best similarity metric, we tested different similarity metrics such as Euclidean distance, TF-IDF, JACCARD, Correlation and Dice. K-means is applied using the Euclidean distance as similarity metric since we had best results comparing to other similarity metrics according to our expert.

We first begin by determining the best number of K by calculating the inertia in figure 2.

According to figure 2, determining visually the number of k cannot always be unambiguously identified. We can estimate the number of K between 25 and 30. 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.

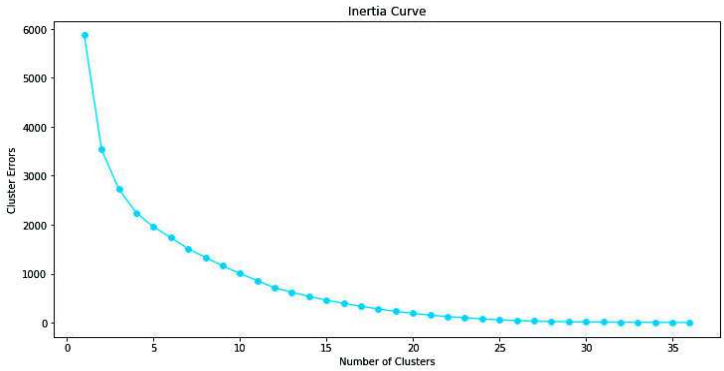

Fig. 2. Inertia curve for Corpus1 dataset

We apply the statistic gap to our datasets and the best number of k is presented in Table 1.

4.3 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 contain 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 or incoherent requirements), which means that we discard clusters with partially relevant requirements. Also, we consider only clusters with more than one requirement.
- "Average" validation (AV): we calculate the average of relevant requirements per cluster.

$$AV_k = (\Sigma i=1, k\ precision(C_i))/\ k' \quad (1)$$

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 ∈{1, k} is the value of k and $precision(c_i)$ is defined as:

$$precision(C_i) = NumberofRelevantRequirements/TotalNumber\text{-}ofRequirements \quad (2)$$

4.5 Classification results with the pre-processing step

For the pre-processing step, we use the Part-Of-Speech (POS) tagging and Noun chunking from SpaCy[3] 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[4]. 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 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*), proer_noun-noun (e.g. *BSP data*), adjective-adjective-noun (e.g. *amber visual indication*), noun-noun-noun (e.g. *output voltage value*).

So, we apply the k-means algorithm on dataset containing technical terms to see the impact of this pre-processing on the results. Table 2 summarizes the different results obtained from the same experiments presented in 4.3.

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

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

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)

Table 2. Results with pre-processing: 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, we have 100% of relevant clusters and 10 relevant clusters for which the number of requirements is greater than 1 in both validations. The clustering has detected clusters with the right redundancy information but two more relevant clusters than the clustering without preprocessing. In this case, the tagging has shown its efficiency to improve redundancy detection results.

In Corpus2 dataset, we have 92.85% of relevant clusters and 13 relevant clusters which their number of requirements is greater than 1 in both validations. The clustering has detected clusters with the right inconsistency information but 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 the total number of relevant clusters. The preprocessing has improved the rate of the redundancy/inconsistency detection.

5. 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 pre-processing, 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). For Corpus2, our expert assumes 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 pre-processing, 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 assumes 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).

The nature of the dataset is very important to define the best value of k. In fact, if the dataset contains too many identical requirements, the k-means algorithm tends to cluster these requirements together and then very similar requirements will be discarded from these clusters and putted in other clusters. This is the case for Corpus3 where we found repetitions of many requirements. We should then take into consideration this characteristic to analyze more efficiently specifications.

Also, we experiment this clustering approach on large dataset (not mentioned above) with ~900 requirements. This type of dataset is very noisy with many identical requirements belonging to different chapters or sections. The results on this dataset are not satisfying. Taking into consideration the information about document hierarchy could help us to analyze different sub-documents and then shorter datasets to find the best clusters.

6. Conclusion

In this paper, we proposed an automatic approach for redundancy and inconsistency detection in requirements engineering context. 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 pre-processing step based on the NLP pre-processing to see the impact of this latter to the k-means results. We used Part-Of-Speech (POS) tagging and noun chunking 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. According to Corpus1 (redundant) and Corpus2 (inconsistent), k-means provides very relevant results by providing only clusters (with more than one requirement) with relevant information. Pre-processing 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 pre-processing step to improve the clustering results in terms of precision and the number of detected clusters.

Even with high quality results for Corpus1 and Corpus2, we are not able yet to differentiate redundancy or inconsistency in very similar clusters in Corpus3. To overcome this shortcoming, we plan to apply another clustering approach on similar clusters. This new clustering will be based on semantic features. Also, we plan to eliminate identical requirements belonging to the same chapter before applying clustering to improve clustering. 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

Dermeval, D. (2016). Applications of ontologies in requirements engineering: a systematic review of the literature. *Requirements Engineering*, *21*(4), 405-437.

Frenay, B. & Verleysen, M. (2014). Classification in the Presence of Label Noise: A Survey. IEEE *Transactions on Neural Networks and Learning Systems*, *25*(5), 845-869.

Belsis, P., Koutoumanos, A. & Sgouropoulou, C. (2014). PBURC: a patterns-based, unsupervised requirements clustering framework for distributed agile software development. *Requirements Engineering*, *19*(2), 213-225.

Anil, K. Jain (2010). Data clustering: 50 years beyond K-means. *Pattern Recognition Letters*, *31*(8), 651-666.

Rago, A., Marcos, C. & Diaz-Pace, J.A. (2016). Identifying duplicate functionality in textual use cases by aligning semantic actions. *Software & Systems Modeling*, *15*(2), 579-603.

Falessi, D., Cantone, G. & Canfora, G. (2013). Empirical Principles and an Industrial Case Study in Retrieving Equivalent Requirements via Natural Language Processing Techniques. *IEEE Trans. Softw. Eng.*, *39*(1), 18-44.

Juergens, E., Deissenboeck, F., Feilkas, M., Hummel, B., Schaetz, B., Wagner, S., Domann, C. & Streit, J. (2010). Can Clone Detection Support Quality Assessments of Requirements Specifications?. *Proceedings of the 32Nd ACM/IEEE International Conference on Software Engineering*,Volume 2, 79-88.

Fu, X., Ch'ng, E., Aickelin, U. & See, S. (2017). 2017 CRNN: A Joint Neural Network for Redundancy Detection. *IEEE International Conference on Smart Computing (SMARTCOMP)*, 1-8.

Abad, Z.S.H., Karras, O., Ghazi, P., Glinz, M., Ruhe, G. & Schneider, K. (2017). What Works Better? A Study of Classifying Requirements. *IEEE 25th International Requirements Engineering Conference (RE)*, 496-501.

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

Klein, D. & Manning, C.D. (2003). Accurate Unlexicalized Parsing. In: *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, Volume 1, 423-430.

MacQueen, J. (1967). Some method for classification and analysis fo multivariante observations. L.M. Le Cam, J. Neyman, J. (eds.), *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, Volume 1, University of California Press, 281-297.

Winkler, J. & Vogelsang, A. (2016). Automatic Classification of Requirements Based on Convolutional Neural Networks. *IEEE 24th International Requirements Engineering Conference Workshops (REW)*, 39-45.

Ott, D. (2013). Automatic Requirement Categorization of Large Natural Language Specifications at Mercedes-Benz for Review Improvements. *Requirements Engineering: Foundation for Software Quality: 19th International Working Conference, REFSQ 2013*, Essen, 50-64.

Knauss, E., Damian, D., Poo-Caamaño, G. & Cleland-Huang, J. (2012). Detecting and classifying patterns of requirements clarifications. In: *20th IEEE International Requirements Engineering Conference (RE)*, 251-260.

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

Brown, P.F., deSouza, P.V., Mercer, R.L., Watson, T.J., Pietra, V.J.D. and Lai, J.C. (1992). Class-Based n-gram Models of Natural Language. *Computational Linguistics*, *18*(4), 467-480.

Mohajer, M., Englmeier, K.-H. & Schmid, V.J. (2010). A comparison of Gap statistic definitions with and without logarithm function, Vol 96, http://nbn-resolving.de/urn/resolver.pl?urn=nbn:de:bvb:19-epub-11920-3.

Hull, E., Jackson, K. & Dick, J. (2011). *Requirements Engineering*. Springer-Verlag London.

Glas, R.L. (2002). *Facts and Fallacies of Software Engineering*. Addison-Wesley Professional.

Stecklein, J.M., Dabney, J., Dick, B., Haskins, B., Lovell, R. & Moroney, G. (2004). Error Cost Escalation Through the Project Life Cycle. *Proceedings of the 14th Annual International Symposium*, Toulouse, France.

Galin, D. (2003). *Software Quality Assurance: From Theory to Implementation*. Pearson.

Bourque, P. (2004). *Guide to the Software Engineering Body of Knowledge (SWEBOK) Guide*. IEEE Computer Society.

Berry, D.M., Kamsties, E. & Krieger, M.M. (2003). *From Contract Drafting to Software Specification: Linguistic Sources of Ambiguity*. https://fr.scribd.com/document/76672102/Ambiguity-Handbook.

Alshazly, A.A., Elfatatry, A.M., Abougabal, M.S., (2014). *Detecting defects in software requirements specification*, Alexandria Engineering Journal, *53*(3), 513-527.