SRXCRM: Discovering Association Rules Between System Requirements and Product Specifications

Vasco Leitão, Ibéria Medeiros

LASIGE, Faculdade de Ciências, Universidade de Lisboa, Portugal

Abstract

Industrial products integrate highly configurable safety-critical systems which must be intensively tested before being delivered to customers. This process is highly time-consuming and may require associations between product features and requirements demanded by customers. Machine Learning (ML) has proven to help engineers in this task, through automation of associations between features and requirements, where the latter are prioritized first. However, ML application can be more difficult when requirements are written in natural language (NL), and if it does not exist a ground truth dataset with them. This work presents SRXCRM, a Natural Language Processing-based model able to extract and associate components from product design specifications and customer requirements, written in NL, of safety-critical systems. The model has a Weight Association Rule Mining framework that defines associations between components, generating visualizations that can help engineers in prioritization of the most impactful features. Preliminary results of the use of SRXCRM show that it can extract such associations and visualizations.

Keywords

Requirement engineering, requirement prioritization, software testing, natural language processing, noun phrase chunking, association rule mining,

1. Introduction

Safety-critical systems (e.g., automotive, railway systems) have been widely integrated into industrial products. Being critical, it is of the utmost importance to test them properly and intensively. Testing is an essential part of the requirement engineering process to better understand the system behavior and to detect failures. Quality testing aims for the reduction of cycle times while avoiding human intervention. This is a hard task, as products can have several configurations, features and corresponding *customer requirements* (CRs), and the available time providers have to deliver them to customers is not enough to test them properly. Software Product Line Engineering (SPLE) tries to tackle this task. Software Product Lines (SPL) are sets of software-intensive systems that share common series of product line features, which allow to derive an individual product from reusable features of the product line [1]. CRs need to be analysed by engineers familiar with the system in order to detect discrepancies with the product. Such is made in a manual process where all features of a product and the existing

In: F.B. Aydemir, C. Gralha, S. Abualhaija, T. Breaux, M. Daneva, N. Ernst, A. Ferrari, X. Franch, S. Ghanavati, E. Groen, R. Guizzardi, J. Guo, A. Herrmann, J. Horkoff, P. Mennig, E. Paja, A. Perini, N. Seyff, A. Susi, A. Vogelsang (eds.): Joint Proceedings of REFSQ-2021 Workshops, OpenRE, Posters and Tools Track, and Doctoral Symposium, Essen, Germany, 12-04-2021

vleitao@lasige.di.fc.ul.pt (V. Leitão); imedeiros@di.fc.ul.pt (I. Medeiros)

project requirements on paper must be correlated. This process is prone to errors, since CRs are described in natural language (NL) or a restricted language and lack any use of formal writing techniques [2]. Accuracy is also dependent on the engineer's experience. Natural Language Processing (NLP) can be applied to support linguistic tasks presented in the SPLE field [3], while *requirement prioritization* can identify relevant requirements to associate with a system, reducing testing costs.

In this paper, we tackle a SPLE scenario focused on identification of requirement similarities and variance. For that, we use NLP techniques over CRs and *product design specifications* (DSs) from a Propulsion and Control railway subsystem (PCS), both written in NL and belong to two different, but related, domains. The paper presents SRXCRM, a NLP-based model for CR prioritization, linking them to DSs in an automated manner. The model contains a *Weight Association Rule Mining* (WARM) framework to process both CRs and DSs for extraction of relevant information, obtaining association rules within each domain and their visualizations. Preliminary results show that the model can extract such associations and visualizations.

The contributions of the paper are: 1) a grammar able to interpret DSs and CRs written in NL; 2) the SRXCRM model capable of extracting components and association rules from DSs and CRs, linking the latter to the former; 3) the *WARM* framework to extract association rules, considering the weights of components; 4) the knowledge graph of the extracted rules and showing their associations; 5) preliminary results of SRXCRM on processing of DSs and CRs.

2. Related Work

There is a great amount of research focused in adapting NLP models to named entity recognition (NER). Passos et al. [4] seek to use a variation of the Skip-Gram [5] model that can extract information from lexicons in order to improve representation. Foley et al. [6] explored NER in limited data as a retrieval task through user interaction. In these models, expert users identify entities in a set of CRF-ranked sentences, generating training data that is suitable for NER.

Some adaptations to specific domains have been made lately as well, in order to reduce user annotation and manual training. Soderland et al. [7] classified domain arguments from elements extracted in extensive noun-phrases, given a small training set. They also proposed an algorithm for rule extraction that attempts to evaluate generalization of relations between distinct classes, although most of the relations extracted are verb-centered.

For requirement modelling, Mu et al. proposed EFRF [8], a framework focused on extracting functional requirements from software requirement specifications, analyzing their linguistic structure. However, they did not specify how the model deals with domain-specific terms.

Schlutter et al. [9] proposed a pipeline for concept extraction from requirement documents, being these represented in graphs that help engineers in knowledge detection and querying. However, the pipeline lacks quality criteria to support validation. We look for validation in communication with domain experts, to understand if term extraction ensures completeness.

Abbas et al. [10] defined a process focused in requirement reuse, using existing CRs in order to recommend features to implement new, unseen CRs. Their work is based on TFIDF or Doc2Vec techniques, followed by embedding clustering to aid recommendation. Despite the good results in accuracy and requirement identification, the lack of understanding of its algorithm from

expert users decreased the methodology acceptance. Our model aims on creating a system based on chunking and Association Rule Mining frameworks [11], which provides easy-to-follow results about component relationships. We propose a model with an Information Extraction (IE) pipeline similar to [12], but we look for different approaches in prioritization.

3. Data

3.1. Raw Dataset

The raw dataset is source material from a PCS, constituted by a set of DSs and CRs, both written in NL and not preprocessed before being handed to us. DSs and CRs follow different domains, i.e, the former follows a low level domain, while the latter fits on a high level domain.

The material was received in form of a document that consists in three main elements: *object identifiers*, DSs, and CRs that may be associated with the DSs. This document follows a hierarchy of sections, such as *Conventions* or *System Overview*. Through exploratory data analysis, we identified 526 instances of DSs, each one identified by a unique *object identifier*, and where 85 of them are related to one or more CRs. Overall, we identify 164 CRs.

A DSs is formally described as an conceptualization of a PCS standard product feature, describing its implementation in NL. It comprises various interfaces and functions, which are some of the components that compose a product feature.

On the other hand, a CR is defined as a concrete description of product feature to be deployed to a customer. It consists in a revision of a product feature in order to adapt the product feature to the needs of a specific customer of the system. It complements the standard feature description integrated in the component specification with new information defined by a customer.

Each DS is identified by an *object identifier*, a code that does not give any information about the system, although we will use it to map DSs. In addition, a DS may be linked to various CRs; and CRs may be linked to several DSs. CRs associated with the same DS are independent.

3.2. Preprocessing

The preprocessing pipeline is focused on formatting the NL sentences of each DS and CR in *noun phrase chunks*, the latter being the main inputs to association rule extraction.

First, we perform *non-specification identification* over DSs. DSs related to section, subsection, and category titles are identified and discarded, since these do not bring any value to evaluation. Afterwards, *tokenization* is done, to split text into *tokens* and determine which these corresponding to words or punctuation marks. Those corresponding to *stopwords* are removed.

The final task of the process is the *Part-of-Speech Tagging*, categorizing tokens by its gramatical tag in each CR and DS, using the Averaged Perceptron Model [13] with the Penn Treebank Tagset [14]. These tags are used as input to the *Main Chunking* phase, described in Section 4.1.

4. The SRXCRM Model

This section is focused on presenting SRXCRM *(System Requirement eXtraction with Chunking and Rule Mining)*, a NLP-based model able to extract *main chunks* for both DSs and CRs, creating

for each domain association rules that map different components. These will be then used to map new CRs to the DSs of the subsystem. SRXCRM also creates visualizations that help engineers identify the most relevant features in each domain.

Figure 1 depicts the proposed model, comprising three main phases: *Preprocessing* (described in Section 3.2), *Main Chunking*, and *Rule and Knowledge Extraction*. The whole process requires a raw dataset of DSs and CRs as input in order to generate, for each domain, the knowledge graph and the association rules that represent the relations between relevant components of the PCS. Rules are established after a *Noun Phrase Chunking* refinement process and a *Weighted Association Rule Mining* (WARM) task. Both processes are detailed in the next sections.

4.1. Main Chunking

The *main chunking* phase aims to recognize components of PCS and extract them from DSs and CRs. For that, the phase receives a set of DSs and CRs, with their sentences already tagged under the *preprocessing* phase, performs the *Noun Phrase Chunking* task over each sentence, and then refines the results to obtain the most relevant chunks, that we refer as *main chunks*.

Noun Phrase Chunking. In order to retrieve components, we have created a *chunk grammar* able to recognize and extract noun phrase chunks (*chunks* for short), i.e., components, over the tagged sentences. The grammar, shown in Table 1, comprises four rules (expressed as regular expressions) that define tag sequences that can be associated with the presence of components.

Main Chunking Refinement. After chunk extraction and manual analysis, we realized that several chunks derive from others. A chunk A *derives* from a chunk B if B is a substring of A, and B occurs more frequently than A. With the aim of getting the most precise results, the *main chunk* concept was introduced, which denotes a chunk with frequency above a threshold t. For A and B, B would be a *main chunk*. Therefore, after retrieving the chunks, we refine the whole set to extract the *main chunks* and, for each one, the set of their *derived chunks*.

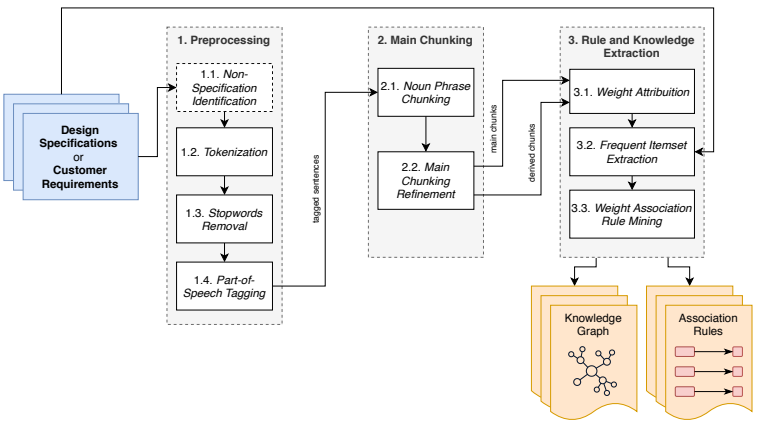

Figure 1: Overview of the SRXCRM model.

Table 1
The rules defined in the chunk grammar.

Rule	Description	*#ch*	HR (%)	PR (%)	NR (%)
{<JJ>+<NNP>+}	At least one adjective (JJ) followed by at least by a proper noun (NNP)	108	65,7	16,7	17,6
{<NN>*<NNP>+<NN>*}	A noun (NN) (if exists) followed by at least a proper noun (NNP), followed by another noun (NN) (if exists)				
{<JJ>+<NN>+}	At least one adjective (JJ) followed by at least a noun (NN)				
{<NN><NN>}	Two consecutive nouns (NN)				

#ch: Number of extracted *main chunks* *HR(%)*: Percentage of *highly relevant main chunks* extracted
PR(%): Percentage of *possibly relevant main chunks* extracted *NR(%)*: Percentage of *not relevant main chunks* extracted

4.2. Rule and Knowledge Extraction

The *Rule and Knowledge Extraction* phase receives as input the main chunks, their derived chunks, the DSs or CRs that have been processed, outputting the association rules discovered between the main chunks and the graph representing the connections between them. This phase employs the *Weight Association Rule Mining* (WARM) process, which evolves from the ARM methodology, with the weight of each chunk that occurs in the dataset.

The WARM process is boosted by the *Weight Attribution* and *Frequent Itemset Extraction* tasks that, respectively, calculates the weight of participation for each chunk and extracts itemsets where they occur within the dataset. Afterwards, the rules are extracted by WARM.

Weight Attribution. We look to define *weights* for each main chunk. For that, first, each chunk *c* (either a main chunk mc or a derived chunk dc) will have an associated *uniqueness value* (UV), which depends on the number of main chunks from where it derives (Equation 1). Secondly, the *weight* (W) of each main chunk mc is determined as being the sum of the UV values of its derived chunks (Equation 2). In order to avoid a great discrepancy between values, we apply min-max normalization to *[0,1]* range over the resulting *weight* array of main chunks. Since Equation 3 uses a product of the $W(mc)$, we perform value smoothing by adding a thousandth unit to the weights to avoid giving zero-value for them, and so to avoid null results in Equation 3. Similarly to zero-value weights, we also apply smoothing over unit weights by subtracting a thousandth unit as well, in order to keep uniformity in the task.

$$UV(c) = \frac{1}{\#(related\ main\ chunks)_c} \quad (1) \qquad W(mc) = \sum_{i=1}^{N} UV(dc_i) \quad (2)$$

Frequent Itemset Extraction. It is created a binary occurrence dataset that sets associations between the DSs or CRs and the occurrence of main chunks. Each DS or CR is encoded as a transaction that contains the occurrences of main chunks. Transactions are obtained following the reasoning described by Agrawal et al. [11]. Given a transaction T and a main chunk mc: if $mc \in T$, $T[mc] = 1$, otherwise $T[mc] = 0$. For example, if we have 15 main chunks and a specification *S* contains 4 main chunks, an instance of the binary dataset will be composed of 15 elements, where 4 of them will have their values equal to 1. Afterwards, a further analysis is

made to recognize if there are DSs or CRs that do not contain any main chunk, i.e., instances of the dataset with all values equal to 0. These are viewed as irrelevant, and therefore are removed.

After refining, *frequent itemset extraction* is performed over the binary dataset, invoking the *Apriori* [15] algorithm to extract itemsets with *support* above a manually defined threshold. These are required to build association rules that express relations between components.

Weight Association Rule Mining (WARM). Following the reasoning of Tao et al. [16], we evolved the standard ARM with the inclusion of the *weights* of chunks for better using the derived chunks of each main chunk. Also, the standard metrics of *support*, *confidence*, and *lift* [11] were adjusted, resulting new ones. WARM is described in the following pipeline: (1) For each itemset $\{mc_1, ..., mc_n\}$ (it for short) from the previous task, it is defined its *Itemset Transaction Weight* (ITW) (Equation 3) reflecting the aggregated weight of all main chunks it contains. (2) *Weighted support* (WS) is calculated through Equation 4. We also make adjustments in the number of transactions, since they depend directly on the WS (Table 2, column 4). (3) Representing a rule as $A \implies B$, where $A = \{mc_1, ..., mc_n\}$, $B = \{mc_m\}$ and $A \cap B = \emptyset$, *weighted confidence* (WC) and *weighted lift* (WL) are obtained from Equations 5 and 6.

$$ITW(\{mc_1, ..., mc_n\}) = \prod_{i=1}^{n} W(mc_i) \quad (3) \qquad WS(it) = \frac{T_{it} \times ITW(it)}{\#(transactions)} \quad (4)$$

$$WC(A \implies B) = \frac{WS(A \cup B)}{WS(A)} \quad (5) \qquad WL(A \implies B) = \frac{WS(A \cup B)}{WS(A) \times WS(B)} \quad (6)$$

The extracted rules follow a minimum WC threshold and are ordered first by WC values. In case of a tie, rules are ordered by WS and then by WL. The knowledge graph that represents the relations between components (i.e., main chunks) is also created, where a graph *node* corresponds to a main chunk that occurs in the set of rules, including notation with its amount of derived chunks, and a *directional edge* corresponds to a rule between two main chunks. As two main chunks might occur in several rules together, the highest possible confidence is represented in the edge for a combination of the two main chunks with coloring methods. Figure 2 shows an example of a knowledge graph containing the association rules between components extracted from DSs, for a minimum WC threshold of 10.0%.

5. Experimental Results

In this section, we evaluate SRXCRM over the raw PCS dataset. First, we give a description of the configuration of its main phases, and then we present the preliminary results obtained.

SRXCRM is implemented with different packages. For preprocessing and noun phrase chunking, we used the NLTK [17] toolkit, namely *tokenization* and *POS tagging*. To conduct *frequent itemset* and *rule extraction*, we used MLxtend [18]. Graphs are obtained with Graphviz [19].

Preprocessing. From preprocessing we were able to extract 2074 sentences from the 228 identified DSs, and 1076 sentences from 164 CRs. The *stopword removal* task involved the English stopword corpus defined by NLTK [17], although we removed the word *not* from the list, since we noticed it had an impact on the integrity of the chunks extracted.

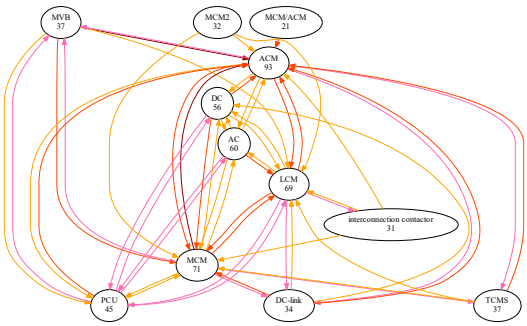

Figure 2: Knowledge graph extracted from 228 DSs representing component relations. If $WC > 80\%$, a red-colored edge is returned. If $80\% \geq WC > 50\%$, an orange-red edge is returned. If $50\% \geq WC > 20\%$, an orange-colored edge is returned. If $20\% \geq WC > 10\%$, a pink-colored edge is returned.

Main Chunking. Noun Phrase Chunking uses the POS tagged sentences and the grammar as input. For each sentence, the grammar parser is ran during 10 epochs and extracts all chunks and respective frequencies. Chunks with $frequency \geq 4$ (the threshold we defined) were considered main chunks. In DS dataset, 8.04% of chunks were signalized as main chunks.

We decided to use PCS human expert validation to evaluate each grammar rule, based on the relevancy of each extracted main chunk. To do so, for each main chunk, we asked if it was *highly relevant* (recognized as a PCS component), *possibly relevant* (PCS state) or *not relevant*. This process was done in two iterations. In the first iteration, with an initial grammar we defined, 89.6% (69) of the 77 extracted main chunks from DSs were considered *highly or possibly relevant* by PCS experts. Afterwards, we adjusted the grammar adding a new rule to capture new interesting chunk patterns (see Table 1), and we rerun the model. In this second iteration, from the 108 main chunks extracted, 82.4% (89) were considered *highly or possibly relevant* by PCS experts. These results are described in Table 1, columns 3-6. Though there is a reduction on the ratio of relevant chunks with the new rule, the grammar is useful because it extracted 16 new *highly relevant* main chunks that are helpful to decision makers. After applied the model over CRs, we verified that 38% of *highly relevant main chunks* extracted in DSs occur in CRs.

Rule and Knowledge Extraction. We compared WARM with ARM. For ARM we extracted frequent itemsets without using the *weight* parameter to understand which patterns would be extracted without the impact of the derived chunks. Frequent itemsets from DSs were extracted with a minimum support threshold of 5 transactions ($\approx$ 1.9%), and then rules were extracted for a minimum confidence threshold of 50% (Table 2, rows 2–6). We manage to extract 428 itemsets (columns 2–4) and 459 rules (columns 5–8) from DSs, many with the highest possible confidence ($C = 1$), which did not allow to get conclusions about *main chunk* relations. Next, to better understand the model behaviour we retrieved the rules with WARM (Table 2, rows 7–11). We reduced the support threshold to 0.5% to capture a significant set of rules, keeping the same minimum confidence threshold, retrieving 65 *weighted* itemsets (columns 2–4) and 102 *weighted* rules (columns 5–8). These resulting rules are better related to the context of the

Table 2
Itemsets and rules extracted in DSs with ARM and WARM.

F	Itemsets	S_i	#T	Association Rules	S_r	C	L
ARM	(MCM)	0.221	57.0	(MCM1) $\Longrightarrow$ (MCM2)	0.07	1.00	9.92
	(AC)	0.186	48.0	(MCM, MCM/ACM) $\Longrightarrow$ (ACM)	0.05	1.00	5.38
	...	...	...	...	...	...	...
	(effort reference, Max)	0.019	5.0	(LCM, MVB) $\Longrightarrow$ (MCM)	0.02	0.50	2.26
	(TCMS, MCM, PCU, MVB)	0.019	5.0	(TCMS, LCM) $\Longrightarrow$ (MCM)	0.02	0.50	2.26
WARM	(ACM)	0.091	23.52	(MCM, MVB) $\Longrightarrow$ (ACM)	0.01	0.83	9.13
	(MCM)	0.084	21.55	(MCM/ACM) $\Longrightarrow$ (ACM)	0.01	0.80	8.77
	...	...	...	...	...	...	...
	(AC, DC, LCM)	0.005	1.31	(LCM) $\Longrightarrow$ (connection contactor)	0.01	0.10	5.58
	(ACM, MCM, TCMS)	0.005	1.30	(LCM) $\Longrightarrow$ (AC, ACM)	0.01	0.10	8.69

F: Framework S_i: Itemset Support *#T*: Number of Transactions S_r: Rule Support *C*: Confidence *L*: Lift

problem due to the fact that weight is dependent of derived chunks and their UV.

The knowledge graph depicted in Figure 2 shows the connections between the 12 components that are recognized from the association rules from WARM, for a minimum weighted confidence threshold of 10.0%. The graph can give a comprehensive overview of the relations between the components, to help engineers understand how these interact inside the processed domains. For CRs we applied the same process. From 123 main chunks discovered, 238 rules were extracted, with the graph representing relations between 22 main chunks.

6. Conclusions

The paper presented SRXCRM, a NLP-based model able to extract, associate, and rank NL requirements from highly configurable systems, resorting to Noun Phrase Chunking and Weight Association Rule Mining. The model was validated given domain expert analysis where almost 90% of the processed noun phrase chunks can be considered relevant to system engineers. As next steps, we intend to perform a sentence similarity task that uses `<DS, CR, match value>` tuples as input, where the `match value` is based on the main chunks and rules contained in each DS and CR. This will require better processing of CRs since these have a broader domain.

Acknowledgments

This work was partially supported by the ITEA3 European through the XIVT project (I3C4-17039/FEDER-039238), and national funds through FCT with reference to LASIGE Research Unit (UIDB/00408/2020 and UIDP/00408/2020). The authors would also like to thank Bombardier Transportation AB, for their continued support.

References

[1] F. J. v. d. Linden, K. Schmid, E. Rommes, Software Product Lines in Action: The Best Industrial Practice in Product Line Engineering, Springer-Verlag, Berlin, Heidelberg, 2007.

[2] M. Kassab, C. Neill, P. Laplante, State of practice in requirements engineering: contemporary data, Innovations in Systems and Software Engineering 10 (2014) 235–241.

[3] L. Zhao, W. Alhoshan, A. Ferrari, K. J. Letsholo, M. A. Ajagbe, E.-V. Chioasca, R. T. Batista-Navarro, Natural language processing (nlp) for requirements engineering: A systematic mapping study, 2020. arXiv:2004.01099.

[4] A. Passos, V. Kumar, A. McCallum, Lexicon infused phrase embeddings for named entity resolution, in: Proceedings of the Eighteenth Conference on Computational Natural Language Learning, Association for Computational Linguistics, Ann Arbor, Michigan, 2014, pp. 78–86.

[5] T. Mikolov, G. Corrado, K. Chen, J. Dean, Efficient estimation of word representations in vector space, 2013, pp. 1–12.

[6] J. Foley, S. M. Sarwar, J. Allan, Named Entity Recognition with Extremely Limited Data (2018) 2–7.

[7] S. Soderland, B. Roof, B. Qin, S. Xu, Mausam, O. Etzioni, Adapting open information extraction to domain-specific relations, AI Magazine 31 (2010) 93–102.

[8] Y. Mu, Y. Wang, J. Guo, Extracting software functional requirements from free text documents, in: 2009 International Conference on Information and Multimedia Technology, 2009, pp. 194–198. doi:10.1109/ICIMT.2009.47.

[9] A. Schlutter, A. Vogelsang, Knowledge representation of requirements documents using natural language processing, RWTH, 2018.

[10] M. Abbas, M. Saadatmand, E. P. Enoiu, D. Sundmark, C. Lindskog, Automated reuse recommendation of product line assets based on natural language requirements, in: 19th International Conference on Software and Systems Reuse, 2020.

[11] R. Agrawal, T. Imieliński, A. Swami, Mining association rules between sets of items in large databases, SIGMOD Rec. 22 (1993) 207–216. doi:10.1145/170036.170072.

[12] R. Sonbol, G. Rebdawi, N. Ghneim, Towards a Semantic Representation for Functional Software Requirements (2020) 1–7. doi:10.1109/AIRE51212.2020.00007.

[13] J. Votrubec, Morphological tagging based on averaged perceptron, WDS'06 proceedings of contributed papers (2006) 191–195.

[14] Alphabetical list of part-of-speech tags used in the penn treebank project, 2003. URL: https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html.

[15] R. Agrawal, R. Srikant, Fast algorithms for mining association rules in large databases, in: Proceedings of the 20th International Conference on Very Large Data Bases, VLDB '94, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1994, p. 487–499.

[16] F. Tao, F. Murtagh, M. Farid, Weighted association rule mining using weighted support and significance framework, Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2003) 661–666. doi:10.1145/956750.956836.

[17] S. Bird, E. Klein, E. Loper, Natural language processing with Python: analyzing text with the natural language toolkit, " O'Reilly Media, Inc.", 2009.

[18] S. Raschka, Mlxtend: Providing machine learning and data science utilities and extensions to python's scientific computing stack, Journal of Open Source Software 3 (2018) 638.

[19] J. Ellson, E. R. Gansner, E. Koutsofios, S. C. North, G. Woodhull, Graphviz and Dynagraph — Static and Dynamic Graph Drawing Tools, Springer Berlin Heidelberg, Berlin, Heidelberg, 2004, pp. 127–148.