A FrameNet-based Approach for Annotating Natural Language Descriptions of Software Requirements

Waad Alhoshan, Riza Batista-Navarro, Liping Zhao
School of Computer Science, University of Manchester, United Kingdom
waad.alhoshan@postgrad.manchester.ac.uk
{riza.batista, liping.zhao}@manchester.ac.uk

Abstract

As most software requirements are written in natural language, they are unstructured and do not adhere to any formalism. Processing them automatically—within the context of software requirements engineering tasks—thus becomes difficult for machines. As a step towards adding structure to requirements documents, we exploited frames in FrameNet and applied them to the semantic annotation of software descriptions. This was carried out through an approach based on automated lexical unit matching, manual validation and harmonisation. As a result, we produced a novel corpus of requirements documents containing software descriptions which have been assigned a total of 242 unique semantic frames overall. Our evaluation of the resulting annotations shows substantial agreement between our two annotators, encouraging us to pursue finer-grained semantic annotation as part of future work.

Keywords: Semantic Frames, FrameNet, Corpus Annotation, Software Requirements, Requirements Engineering

1. Introduction

Software requirements play a pivotal role in all system design phases. Requirements are generally written in natural language, and therefore are unstructured (Ferrari et al., 2017a). This however presents a challenge to Requirements Engineering (RE) tasks, e.g. requirements analysis, which often necessitate the organisation and management of requirements in a systematic manner (Dick et al, 2017). While certain RE tasks (e.g., modelling) could benefit from automated analysis, this can only be facilitated if some structure is applied to the otherwise unstructured natural language requirements contained in software descriptions (Ferrari et al., 2017b).

One way by which we can add structure to software descriptions written in natural language is by attaching machine-readable semantic metadata that captures meaning. In documents from the general and scientific domains, this often corresponds to named entities, e.g., proper names of persons, places, diseases or chemical compounds. Software descriptions however do not allude to such proper names as often and instead mention generic if not abstract concepts (e.g., account creation, file deletion) and the participants involved (e.g., user, system). As shown in early work by Belkhouche and Kozma (1993) and Rolland and Priox (1992), capturing meaning contained in requirements can be approached by using *semantic frames*: coherent structured representations of concepts (Petruck, 1997). These representations are based on the theory of *frame semantics* proposed by Fillmore (1977) whose work formed the basis of FrameNet, an online computational lexicon that catalogues detailed information on semantic frames[1] (Baker et al., 1998). For every frame it contains, FrameNet specifies the following: frame title, definition, frame elements (i.e., participants) and lexical units, i.e., words that evoke the frame. The concept of creation, for example, is encoded in FrameNet as a frame entitled *Creating*, with frame elements pertaining to *Creator*, *Created_entity* and *Beneficiary* (among many others). Importantly, lexical units that signify the concept is also provided, each of which is represented as a combination of their lemmatised form and part-of-speech (POS) tag (e.g., *assemble.v*, *create.v* where *v* stands for verb). Such a frame can then be applied on a piece of text (such as in Example 1) to represent, in a structured manner, the creation idea that is being conveyed. Containing over 1,200 such frames, FrameNet has become an invaluable resource to the NLP research community.

Example 1:
[The system] Creator [generates] Creating_lexical unit [records of user activities] Created_entity [each time] Frequency [the user logs into the system] Cause.

Recent studies in RE have explored the application of FrameNet frames to software requirements acquisition and analysis. For example, Jha and Mahmoud (2017) employed semantic frames (automatically extracted by the SEMAFOR semantic role labeller[2]) as features in training machine learning-based models for categorising user reviews of mobile applications. Meanwhile, Kundi and Chitchyan (2017) proposed a technique for gathering requirements that employed FrameNet frames as the basis of linguistic patterns for generating use cases at the early stages of RE. They specifically made use of the *Agriculture* frame to demonstrate their approach.

We consider FrameNet as a rich repository of semantic metadata that can be added to requirements documents in order to add structure to them. In this work, we seek to employ FrameNet as the basis of a scheme for capturing the meaning of software descriptions. To this end, we adopt FrameNet semantic frames in annotating software requirements in a corpus of documents written in natural language. To the best of our knowledge, our work is the first attempt to investigate FrameNet as a means for annotating meaning within requirements documents. In this way, we are enriching them with semantic metadata and hence incorporating structure into them. As a result, we have produced and made publicly available a resource for the perusal of other members of the research

[1] https://framenet.icsi.berkeley.edu

[2] http://www.cs.cmu.edu/~ark/SEMAFOR/

community: the FrameNet-annotated FN-REQ[3] corpus of natural language requirements documents.

The rest of this paper is organised as follows. Section 2 describes our methods for collecting software requirements documents and annotating them based on the semantic frames contained in FrameNet. In Section 3, we present and analyse results of our annotation. Lastly, we present our conclusions and plans for future work in Section 4.

2. Methodology

In this section, we present the methods we carried out in order to construct a corpus of documents containing sentences of software requirements, and to subsequently annotate them according to FrameNet.

2.1 Document Selection

Our goal is to gather a document set consisting of different types of software requirements. As a preliminary step, we formed a Google search query containing keywords such as "software description", "natural language requirements" and "software requirements specification". Furthermore, we employed snowball sampling and found additional requirements from various sources such as web blogs, research articles (together with their corresponding datasets), lecture materials and industrial/commercial documents. This step resulted in the collection of 34 requirements documents varying in length. The NLTK tool[4] for sentence boundary detection was then applied on the 34 documents. After manually verifying the results, a total of 1,148 sentences[5] were obtained (corresponding to 21,012 tokens).

2.2 Annotation Procedure

The annotation was carried out in a semi-automatic manner. This was facilitated by the two main steps described as follows.

2.2.1 Evoking Frames by Lexical Unit Matching

With the intention of making the annotation process more efficient, we developed a simple method for automatically matching words in the software descriptions in our corpus against lexical units contained in FrameNet, in order to evoke candidate semantic frames. The tokens contained in the requirements documents were lemmatised and assigned part-of-speech (POS) tags using NLTK. For every description, we attempt to match each token (together with its lemma and POS tag) against lexical units in FrameNet, via the application programming interface (API) available in NLTK[6]. We note that only particular types of FrameNet lexical units were considered by this matching method, namely: all verbs and any expressions pertaining to time (e.g., "beforehand"), condition (e.g., "in case", "otherwise"), additional action (e.g., "further), inclusion (e.g., "inclusive"), exclusion (e.g., "excluding"), contradiction (e.g., "nevertheless"), causation (e.g., "because of") and purpose (e.g., "in order"). The selection of these types was informed by our observations on the linguistic styles often used in writing software requirements. Through this process, we were able to evoke candidate semantic frames that denote the meaning of the requirements in our documents.

2.2.2 Validation

Deciding which FrameNet semantic frames capture the meaning expressed in software descriptions was performed manually in order to maximise accuracy. For this task, we employed two annotators. The first annotator (Annotator A) is a requirements engineer with five years of experience in the IT industry. The second annotator (Annotator B) is one of the authors of this paper and is a PhD candidate whose study is focussed on the use of NLP techniques to support RE tasks.

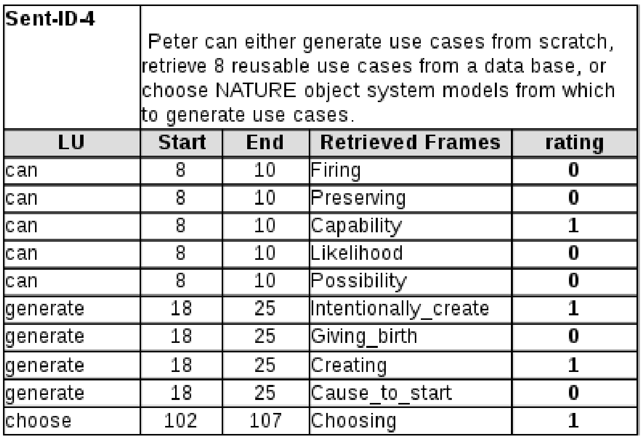
Provided with candidate frames obtained in the previous step, the annotators were asked to confirm whether they capture the meaning of a given software description or not. This validation process was carried out in accordance with the guidelines we developed which drew inspiration from the FrameNet annotation scheme proposed by (Baker, 2017). Over a four-week period, both annotators were trained in applying these guidelines on the annotation of a set of software descriptions from documents other than those in our corpus. Afterwards, the entire corpus of 34 documents—together with the candidate semantic frames retrieved in the previous step—was presented to each of Annotators A and B for annotation. We provide Table 1 to show an example of the details that are presented to an annotator and the kind of judgement that he/she is expected to provide. At the top row of the table is a sample software description. The first column (LU) lists the lexical units matched by the method described in Section 2.2.1. The second and third columns (Start and End) indicate the location of the corresponding lexical unit in terms of character offsets—useful information in cases where a lexical unit appears multiple times within a description. The fourth column (Retrieved Frames) lists the titles of the frames linked with the matched lexical units and are thus considered as candidate frames for annotating the given description. The annotator indicates in the last column his/her judgement on whether a candidate frame applies to the software description (rating = 1) or not (rating = 0). Both annotators completed this task for all 1,148 software descriptions in our corpus.

[3] Read as "fine req"

[4] http://www.nltk.org/api/nltk.html

[5] Identified based on sentence delimiters such as the full stop. Not all of these however are sentences in the strict sense; some are phrases. They all however pertain to software descriptions, thus we use "descriptions" rather than "sentences" in the rest of this paper.

[6] http://www.nltk.org/howto/framenet.html



Sent-ID-4	Peter can either generate use cases from scratch, retrieve 8 reusable use cases from a data base, or choose NATURE object system models from which to generate use cases.			
LU	**Start**	**End**	**Retrieved Frames**	**rating**
can	8	10	Firing	**0**
can	8	10	Preserving	**0**
can	8	10	Capability	**1**
can	8	10	Likelihood	**0**
can	8	10	Possibility	**0**
generate	18	25	Intentionally_create	**1**
generate	18	25	Giving_birth	**0**
generate	18	25	Creating	**1**
generate	18	25	Cause_to_start	**0**
choose	102	107	Choosing	**1**

Table 1. A sample software description from the corpus. An annotator is presented with the automatically matched lexical units, their character offset locations and the titles of the frames linked with them. He/she then indicates whether the frames apply to the requirements (rating = 1) or not (rating = 0). (NB: The second instance of "generate" is also presented to the annotator but excluded here for brevity.)

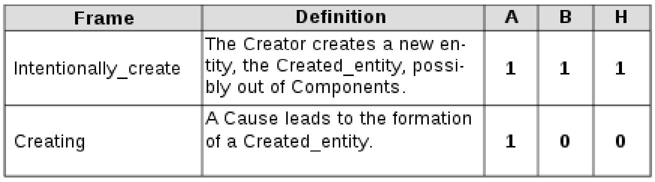
3. Results and Discussion

In this section, we discuss the results of the methodology described above by providing details on inter-annotator agreement and reasons behind annotator discrepancies. We then describe additional steps that were taken in order to prepare the corpus for publication. After presenting attributes of the resulting corpus in terms of annotation frequencies, we discuss a few suggestions on how our proposed annotation method can be useful to members of the research community within the context of RE tasks.

3.1 Inter-annotator Agreement

In order to assess the consistency of annotations between our two annotators, we evaluated inter-annotator agreement based on Cohen's kappa coefficient (McHugh, 2012) as well as the harmonic mean of recall and precision, i.e., F-score. We obtained "substantial" agreement[7] according to Cohen's kappa (72.81%). Furthermore, after determining the number of true positives, false positives and false negatives (by treating the annotations from Annotator B as gold standard and those from Annotator A as response) and micro-averaging over all the documents in our corpus, we obtained an F-score of 80.89%. These results indicate that there is a more than satisfactory level of consistency between our two annotators, implying that their annotations can be considered as highly reliable.

Nevertheless, we investigated the reasons of discrepancy between our two annotators. We found that these are mostly due to close semantic relationships between certain semantic frames. FrameNet, for example, contains a *Creating* and an *Intentionally_create* frame, both of which would be retrieved by our automated lexical unit matching method—and thus presented to an annotator—for a description containing the word "generate" as a verb. As these two frames have similar lexical units and are linked by hyponymy (where *Intentionally_create* has *Creating* as its parent frame), Annotator A could select one frame while Annotator B might select the other (or both, as shown in the example in Table 2). Aiming to produce annotations that are of the highest quality as possible, we resolved these discrepancies, as described in the next section, prior to publishing the annotated corpus.



Frame	**Definition**	**A**	**B**	**H**
Intentionally_create	The Creator creates a new entity, the Created_entity, possibly out of Components.	**1**	**1**	**1**
Creating	A Cause leads to the formation of a Created_entity.	**1**	**0**	**0**

Table 2. A case where Annotator A's judgements on which frames apply to the the word "generate" (in the software description in Table 1), are in disagreement with those of Annotator B. This can be attributed to the hyponymic relationship between the *Intentionally_create* and *Creating* frames. The last column is for recording the results of harmonisation (H).

3.2 Preparation of the Final Corpus

In order to produce the final set of annotations, we harmonised the judgements provided by our two annotators, addressing the primary cause of discrepancies discussed in the previous section. From the set of semantic frames for which the annotators were in disagreement, the following instances were revisited by Annotator B: (1) where the FrameNet frame that she selected as being most relevant to a description is semantically related to the one selected by Annotator A; and (2) where multiple—presumably semantically related—frames were selected for a word in a description. Annotator B reviewed information pertinent to the frames in question, e.g., the definitions and descriptions provided in FrameNet, examples of annotations in the FrameNet corpus[8], as well as the judgements provided by Annotator A. In cases where she is convinced that Annotator A's judgements were more correct, she modified her own annotations; otherwise, she kept her original judgements. She also ensured that only one frame is assigned to a given word (i.e., the matched lexical unit), choosing the one that best captures the meaning of a description (as she understands it), while also reviewing the definitions and examples that are available in FrameNet. The outcome of this process formed the basis of the final set of annotations in our corpus.

3.3 Frequency Analysis

After harmonisation of manually provided judgements, we performed frequency analysis over the final set of annotations, the results of which are presented in Table 3. Alongside these we also provide the frequency of annotations resulting from our automated lexical unit matching method, as the reader might be interested in seeing how much improvement was obtained after manual validation and harmonisation. As one can expect, the automated method for matching lexical units introduced a considerable amount of noise. Firstly, the matching of

[7] As stipulated in Landis and Koch (1977)

[8] Refer to Language Resource Reference

tokens (with their lemmatised forms and POS tags) against FrameNet lexical units does not have perfect accuracy as the POS tagger that we utilised was assigning the wrong POS tag to tokens in a few cases. Secondly, for a given word from a description, e.g., "generate", our method would have retrieved all frames that are associated with the "generate" lexical unit regardless of the sense (e.g., *Intentionally_create*, *Giving_birth*, *Creating*, *Cause_to_start*). This would have resulted in a significant number of false positives, i.e., frames that are irrelevant to a given software description. These issues were however rectified during manual validation and subsequently, during harmonisation.

In our final set of annotations, only frames with rating = 1 (after manual validation and harmonisation) were included. We can observe from Table 3 that out of the 408 semantic frames retrieved through automated lexical unit matching, 166 (40.7%) were eliminated during manual validation and harmonisation, and thus were not included in the final set. There was also a significant drop in terms of the average number of frames assigned to each software description (from 8.82 per description to only 2.21).

	Automated lexical unit matching	**Final set of annotations**
Total number of unique frames	408	242
Total number of unique lexical units	372	340
Average number of frames per software description	8.82	2.21

Table 3. Frequency analysis over the final set of annotations in the FN-REQ corpus. For comparison, we also provide the frequency of annotations obtained through automated lexical unit matching (prior to manual validation and harmonisation).

Our corpus can be considered as densely annotated, with semantic frames assigned to 88.4% of the total number of descriptions (1,015 out of 1,148). Annotations were encoded in a standoff manner, i.e., separately from the documents that were annotated. While the requirements documents were stored following an extended version of the schema proposed by (Ferrari et al., 2017), the annotations were encoded according to the FrameNet format (Baker, 2017).

3.4 Potential Applications

The utilisation of frames in FrameNet to attach semantic metadata to software descriptions—as demonstrated in this work—could potentially facilitate the (partial) automation of certain requirements engineering tasks. For instance, similarities between requirements statements written in natural language can be automatically detected or measured on the basis of the semantic frames assigned to each of them. This in turn can enable *traceability*, i.e., establishing relationships or groupings between requirements and effectively, the software systems they pertain to (Zogaan et al., 2017). Additionally, attaching semantic metadata derived from FrameNet to requirements statements makes them machine-readable and hence more searchable. A software engineer developing requirements for a new system can thus find existing requirements of relevance in a more efficient and systematic manner. In this way, the *reusability* of existing requirements can be enhanced, hence avoiding unnecessary duplication of efforts (Alonso-Rorís et al., 2016).

4. Conclusion and Future Work

In this work, we demonstrated how semantic frames can be applied to the annotation of software descriptions. Along the way, we produced FN-REQ corpus, which we have made publicly available, together with other associated resources (e.g., annotation guidelines, the script that automates matching of FrameNet lexical units), at https://data.mendeley.com/datasets/s7gcp54wbv/1 .

As we were progressing with the manual annotation process described in this work, both annotators observed that there are words in some descriptions which to them clearly pertain to software requirements, but however cannot be assigned any of the frames in FrameNet. For example, it is now typical for software requirements to mention the process of logging into a system, often signified by the verb "log" (as in Example 1 in Section 1). However, none of the frames in the most recent version of FrameNet conveys this concept. This is not a surprise as FrameNet is a general vocabulary and was not designed to cater to specific domains. However, for our purposes of supporting requirements engineering tasks as part of downstream applications, it is worth investigating how many of such requirements in our corpus are currently not covered by FrameNet, in order to assess if there is scope for extending it through the proposal of new additional frames. This is part of our ongoing work. Furthermore, we are in the process of extending our FN-REQ corpus with more requirements documents, while we also carry out finer-grained annotation of software descriptions by labelling frame elements as well. In our future work, we shall exploit the corpus in the context of RE tasks, specifically in detecting traceability and reusability of software requirements.

Acknowledgement

We thank Mohammed Homaid for the time and effort he spent on the annotation of our corpus. We are also grateful for our reviewers whose comments and feedback helped improve the paper. Waad Alhoshan’s PhD is sponsored by the Al-Imam Muhammad Ibn Saud University..

References

Alonso-Rorís, V. M., Álvarez-Sabucedo, L., Santos-Gago, J. M., & Ramos-Merino, M. (2016). Towards a cost-effective and reusable traceability system. A semantic approach. *Computers in Industry*, (83):1-11).

Baker, C. F., Fillmore, C. J., & Lowe, J. B. (1998, August). The berkeley framenet project. In *Proceedings of the 17th international conference on Computational linguistics-Volume 1* (pp. 86-90). Association for Computational Linguistics.

Baker, C. F. (2017). FrameNet: Frame Semantic Annotation in Practice. In *Handbook of Linguistic Annotation* (pp. 771-811). Springer, Dordrecht.

Belkhouche, B., & Kozma, J. (1993). Semantic case analysis of informal requirements. In *Proceedings of the 4th Workshop on the Next Generation of CASE Tools* (pp. 163-181).

Hull, E., Jackson, K., & Dick, J. (2010). Management Aspects of Requirements Engineering. In *Requirements Engineering* (pp. 159-180). Springer London.

Ferrari, A., DellOrletta, F., Esuli, A., Gervasi, V., & Gnesi, S. (2017a). Natural Language Requirements Processing: A 4D Vision. *IEEE Software*, *34*(6), (pp. 28-35), IEEE.

Ferrari, A., Spagnolo, G. O., & Gnesi, S. (2017). PURE: A Dataset of Public Requirements Documents. In *Requirements Engineering Conference (RE), 2017 IEEE 25th International* (pp. 502-505). IEEE.

Fillmore, C. J. (1977). Scenes-and-frames semantics. *Linguistic structures processing*, *59*, 55-88.

Jha, N., & Mahmoud, A. (2017). Mining user requirements from application store reviews using frame semantics. In *International Working Conference on Requirements Engineering: Foundation for Software Quality*(pp. 273-287). Springer, Cham.

Kundi, M., & Chitchyan, R. (2017). Use Case Elicitation with FrameNet Frames. In *2017 IEEE 25th International Requirements Engineering Conference Workshops (REW)* (pp. 224-231). IEEE.

McHugh, M. L. (2012). Interrater reliability: the kappa statistic. *Biochemia Medica: Casopis Hrvatskoga Drustva Medicinskih Biokemicara / HDMB*, (pp. 276-282).

Petruck, M. R. L. (1997). Frame semantics. Handbook of Pragmatics (Vol. 12, pp. 1-13). Amsterdam: John Benjamins Publishing Company.

Rolland, C., & Proix, C. (1992). A natural language approach for Requirements Engineering. In Advanced Information Systems Engineering (pp. 257-277). Springer, Berlin, Heidelberg.

Landis, J. R., & Koch, G. G. (1977). An application of hierarchical kappa-type statistics in the assessment of majority agreement among multiple observers. *Biometrics*, (pp. 363-374).

Zogaan, W., Sharma, P., Mirahkorli, M., & Arnaoudova, V. (2017). Datasets from Fifteen Years of Automated Requirements Traceability Research: Current State, Characteristics, and Quality. In *Proceedings of the 25th International Requirements Engineering Conference* (pp. 110-121). IEEE.

Language Resource Reference

FrameNet. (2017). The FrameNet project, distributed via International Computer Science Institute in Berkeley, 1.7, URL: https://goo.gl/Nbuqvd