Appendix

In this online appendix, we provide supplementary material and detailed information to support the findings presented in our paper **Classification of Crowd-Based Software Requirements via Unsupervised Learning**. This includes explanation about fundamental concepts discussed in the paper, additional tables and figures with explanation. Readers are encouraged to refer to this appendix for a comprehensive view of experimental setup, analysis and results interpretation. We believe that the contents of this appendix will enrich the discussion on the topic addressed in our paper.

1 Background

This section discusses some fundamental concepts that are useful in understanding the content of the paper.

1.1 CrowdRE Dataset

The CrowdRE[1] dataset was gathered during a study [1] conducted for investigating the impact of human personality and creativity potential in crowd-based RE. It includes the personality traits and creative potential of crowd workers, requirements identified for a smart home application, and the creativity ratings assigned to these requirements. A two-stage sequential crowd-based RE process has been used where workers in one stage examine the requirements generated in the previous stage and contribute additional requirements. To reduce the information overload, selection strategies have been used to decide which requirements are presented to the workers in the subsequent stages. An empirical investigation was conducted on Amazon Mechanical Turk to explore the process of creating requirements by 300 workers, and then having 300 additional workers to evaluate the creativity (in terms of novelty and usefulness) of those requirements.

This dataset contains 2966 requirements for smart home application domain. Each entry in the dataset has 6 attributes, i.e., role, feature, benefit, application domain, tags and date-time of creation. Since we are interested in the classification of these requirements using clustering techniques, we only consider the feature and benefit attributes for our experiments. In Table 2, some examples from the dataset are presented. An example requirement obtained after merging the feature and benefit attributes is as follows: *my smart home to be able to order delivery food by simple voice command, I can prepare dinner easily after a long day at work.* The descriptive statistics of the CrowdRE dataset is shown in Table 1.

Table 1: Descriptive statistics of CrowdRE dataset

Application Domain	No. of requirements	% of total requirements
Health	593	20.0%
Energy	626	21.1%
Entertainment	471	15.9%
Safety	892	30.1%
Others	384	12.9%

[1]https://crowdre.github.io/murukannaiah-smarthome-requirements-dataset/

Table 2: Examples from CrowdRE dataset

Requirement	Feature	Benefit	Application Domain
R1	my smart home to be able to order delivery food by simple voice command	I can prepare dinner easily after a long day at work	Health
R2	my smart home to have a certain television show start on my TV when I arrive home	I can immediately jump into watching it	Entertainment
R3	my smart home to shut the lights off for me in rooms with nobody in them	I can save on electricity on my bills	Energy
R4	my smart home to have video camera's outside my home	I always know who is at my door before checking	Safety
R5	my smart home to open my garage door when it senses my vehicle arriving outside of it	I don't have to worry about carrying a remote control opener	Other

1.2 Clustering

Clustering is the grouping of objects such that objects within the same group share more similarities with each other than with the objects in the other groups. We only focus on hard clustering algorithms for our experiments where each data point belongs to only one cluster. In clustering, unsupervised learning is employed to group unorganized data based on *similarities, patterns or differences* without any prior training on the data. In this paper, we have used the following two clustering algorithms: *K-means* [2] and *Hierarchical Agglomerative Clustering (HAC)* [3].

1.2.1 K-means

K-means clustering partitions the data points into K clusters. The objective of the K-means algorithm is to minimize the total squared error. This involves clustering data points in such a way that the distance between points in the same cluster is minimized while the distance between points belonging to different clusters is maximized. The algorithm employs the Expectation-Maximization approach and begins with K (randomly selected) centroids. Each data point is assigned to the nearest centroid based on the distance between the point and the centroids, and the new centroids are calculated by averaging the data points in the cluster. The process continues until there is no change in the assignment of data points to the clusters.

1.2.2 Hierarchical Agglomerative Clustering (HAC)

HAC is a clustering algorithm that works in a bottom-up manner. Initially, the algorithm considers each data point as a separate cluster. As the algorithm progresses iteratively, it merges different clusters with the goal of minimizing a specific linkage criterion. The outcome of this iterative merging process is a *dendrogram* which is a tree structure that represents the data points and their respective clusters. Dendrogram can be used to determine the number of clusters present in the HAC clustering. For merging the clusters, ward linkage criteria is used as it calculates the similarity between two clusters by taking all pairs of points and computing their similarities (i.e., the sum of the square of distances) and taking an average of similarities.

1.3 Text Vectorization

Text Vectorization is the process of converting text into numerical representation. There are many methods to convert textual data into numerical vectors. In this paper we have used three types of models, namely, word frequency based, word embeddings based and transformer based models to convert text into numerical vectors.

1.3.1 Word Frequency Based

Word frequency based models like bag-of-words (BOW) and term frequency inverse document frequency (TF-IDF) [4] have been used for our experiments. BOW model generates a vector of a document by considering the frequency of its words in the corpus. TF-IDF model calculates a word's significance in a document by combining its frequency within the document and its frequency across all the other documents. Both unigrams and bigrams were included in the vocabulary. Vectors generated through frequency-based models are sparse in nature and do not capture the semantics of words in the corpus.

1.3.2 Word Embeddings Based

Word embeddings [5] are a powerful approach for language analysis and have been extensively used in the IR and text mining communities. In this paper, three word embedding based models, i.e., Word2vec [6], FastText [7], and Glove [8] have been used for producing an embedding vector associated with each word in the corpus. These models convert words into high-dimensional numeric vectors, which can retain the syntactic and semantic relationships between the words in the vector space. Note that embeddings of similar words would be close in the vector space. Word2vec and Glove involve feeding individual words into the neural network, whereas FastText breaks words into several n-grams (subwords). For example, the tri-grams for the word *apple* are *app*, *ppl*, and *ple*. The embedding vector for the word *apple* will be the sum of all these n-grams. In short, FastText is able to properly represent rare words since it is highly likely that some of their n-grams also appear in other words. Note that both Word2vec and Glove fail to provide any vector representation for words that are not in the corpus. Both the *Pre-trained* and the *Self-trained* (on the unlabeled CrowdRE corpus with only feature and benefit attributes) variants of these models have been used. The number of epochs was taken as 30, and a window size of 5 (i.e., 5 words before and after the target word were considered as context) was taken as parameters for training the word embeddings based models. To obtain the sentence embeddings from these word vectors, the following two approaches were used: *(1) Average of word vectors of a sentence (2) TF-IDF weighted average of word vectors of the sentence.* Embeddings generated from the word embeddings based models are context independent i.e., if the same word is used in different contexts in different parts of a document it would still have the same vector. Finally, a generalization of the Word2vec method called Doc2Vec [9] has also been used in our experiments. Doc2Vec is used to create a numeric representation of a document/sentence regardless of its length.

1.3.3 Transformer Based

Bidirectional Encoder Representations from Transformers (BERT) [10] is a deep learning model in which every output element is connected to every input element, and the weightings between them are dynamically calculated based upon their connection (attention). BERT can be used to extract contextual word embeddings where these embeddings alter depending on the context in which a word is used. Pre-trained versions of Universal Sentence Encoder (USE) [11], Sentence-BERT (SBERT) [12], Language-Agnostic BERT Sentence Embedding (LaBSE) [13] and Sentence-Robustly Optimized BERT Approach (S-RoBERTa) [14] were used in our experiments to extract sentence embeddings. Embeddings generated using transformer based models are close in the embeddings space for sentences which are semantically similar.

1.4 Evaluation Metrics

- **Marco Precision:** It measures the proportion of true positive predictions among all positive predictions made by the model. Macro averaged precision is calculated by taking the average of precision scores for each class, without considering the class imbalance.

$$\text{Precision}_{\text{macro}} = \frac{1}{n}\sum_{i=1}^{n}\frac{TP_i}{TP_i + FP_i}$$

- **Marco Recall:** It measures the proportion of true positive predictions among all actual positive instances in a class. Macro averaged recall is calculated by taking the average of recall scores for each class, without considering the class imbalance.

$$\text{Recall}_{\text{macro}} = \frac{1}{n}\sum_{i=1}^{n}\frac{TP_i}{TP_i + FN_i}$$

- **Macro F1 score:** It is the harmonic mean of the macro averaged precision and recall.

$$\text{F1-score}_{\text{macro}} = \frac{2 \times \text{Precision}_{\text{macro}} \times \text{Recall}_{\text{macro}}}{\text{Precision}_{\text{macro}} + \text{Recall}_{\text{macro}}}$$

2 Pretrained Model Version and Embeddings Dimension

The model versions and vector dimensions of pre-trained models used in our experimentation such as Word2vec, FastText, Glove, USE, S-RoBERTa, SBERT and LaBSE have been mentioned in Table 3.

Table 3: Pre-Trained models used

Model	Version	Embedding Dimension
Word2vec	google-news-300	300
FastText	wiki-news-subwords-300	300
Glove	wiki-gigaword-300	300
USE	universal-sentence-encoder	512
S-RoBERTa	all-distilroberta-v1	768
SBERT	paraphrase-MiniLM-L6-v2	768
LaBSE	sentence-transformers/LaBSE	768

3 MRR and NDCG of Different Embeding Models

The Mean Reciprocal Rank (MRR) and Normalized Discounted Cumulative Gain (NDCG) scores of different embedding strategies used to convert textual requirements into numerical vectors are shown in Table 4.

Table 4: MRR and NDCG of embedding models over CrowdRE Dataset

Model	**Model Type**	**MRR**	**NDCG**
S-RoBERTa	Transformer Model	**0.738**	**0.771**
SBERT	Transformer Model	**0.731**	**0.770**
Word2Vec(Self Trained)	Word Embedding	**0.725**	**0.769**
USE	Transformer Model	0.722	0.771
Word2Vec(Pre Trained)	Word Embedding	0.722	0.770
LaBSE	Transformer Model	0.720	0.770
Tf-IDF Word2Vec(Self Trained)	Word Embedding	0.712	0.770
Tf-IDF Word2Vec(Pre Trained)	Word Embedding	0.712	0.770
Glove(Pre Trained)	Word Embedding	0.708	0.769
FastText(Pre Trained)	Word Embedding	0.704	0.769
TF-IDF(1,2-gram)	Word Frequency	0.699	0.768
TF-IDF Glove(Pre Trained)	Word Embedding	0.697	0.769
TF-IDF Fasttext(Pre Trained)	Word Embedding	0.696	0.769
Bag of Words(1,2-gram)	Word Frequency	0.687	0.769
TF-IDF Fasttext(Self Trained)	Word Embedding	0.680	0.768
FastText(Self Trained)	Word Embedding	0.674	0.767
Doc2Vec	Word Embedding	0.669	0.766
TF-IDF Glove(Self Trained)	Word Embedding	0.589	0.768
Glove(Self Trained)	Word Embedding	0.580	0.767

4 i-nary clustering

We have conducted experiments involving both binary and multi-class classification, e.g., tertiary, quaternary, and quinary clustering approaches. For binary clustering, we considered all possible combinations of two sectors from the list of sectors. For e.g., in [**Energy,Entertainment**] we only took the requirements belonging to these two sectors, vectorized them, applied clustering, and then performed topic modeling on the resulting clusters. Subsequently, we conducted two types of experiments i.e. manual and automated labeling for assigning labels to the clusters obtained thereby assigning label to each requirement. Now, with the labels assigned to each requirement and ground truth available (as every requirement already has been classified into one of the application domain (i.e. sectors) in the CrowdRE dataset. We have used these labels (sectors) for our experiments as ground truth), we calculated precision, recall, and F1-score.

Similarly, for tertiary, quaternary, and quinary clustering, we considered all possible combinations of three, four, and five sectors, respectively. We only took the requirements belonging to these combinations of sectors and applied the aforementioned steps to determine the label for each requirement and reported the precision, recall, and F1-score in the paper.

To perform i-nary clustering if no prior ground truth exists, we can use same embedding technique (e.g., SRoBERTa, SBERT, or Word2vec) to create vectors if the application domain of new use case is very similar to CrowdRE. And if the domain is different from the one used in the paper it is advisable that we can reevaluate different embeddings for new use case. Other components of our approach like clustering algorithms may also be required to be reevaluated if the domain is significantly different from the one discussed in this paper.

5 Automated Labelling

In Table 5, we present the results for the experiments where K-means and HAC clustering algorithms have been combined with the self trained Word2vec embeddings for automated labelling.

6 Visualizing CrowdRE Requirements

In Figure 1, we present the visualization of all 2966 CrowdRE requirements, which were vectorized using the SRoBERTa model. It can be observed from the figure that there is a significant overlap between the clusters representing the Health (red) and the Other (orange) sectors.

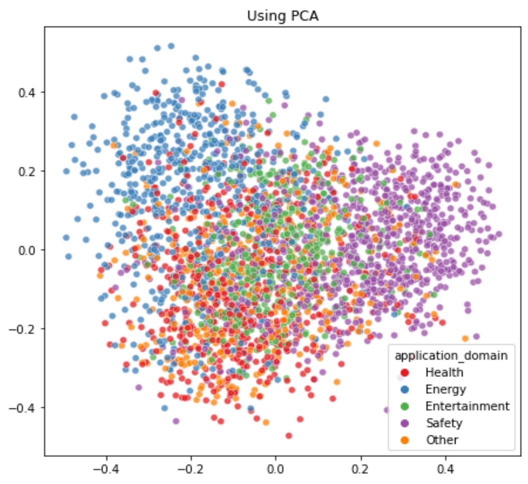

Figure 1: Visualising SRoBERTa embeddings

Table 5: K-means and HAC clustering on Word2vec(Self Trained) embeddings (Automatic Labelling)

		K-means clustering			HAC clustering		
		Precision	Recall	F1-score	Precision	Recall	F1-score
Binary Clustering	Energy,Entertainment	**0.904**	**0.899**	**0.901**	0.832	0.838	0.833
	Energy,Safety	0.874	0.882	0.877	**0.855**	**0.844**	**0.848**
	Entertainment,Safety	0.856	0.875	0.864	0.874	0.733	0.756
	Health,Entertainment	0.803	0.800	0.801	0.779	0.771	0.773
	Health,Safety	0.825	0.838	0.827	0.736	0.742	0.718
	Health,Energy	0.799	0.799	0.798	0.763	0.759	0.757
	Energy,Other	0.803	0.821	0.800	0.748	0.759	0.750
	Entertainment,Other	0.785	0.773	0.758	0.766	0.761	0.763
	Safety,Other	0.664	0.689	0.619	0.638	0.664	0.627
	Health,Other	0.522	0.523	0.521	0.542	0.542	0.542
Tertiary Clustering	Energy,Entertainment,Safety	0.812	0.822	0.815	0.816	0.723	0.744
	Health,Entertainment,Safety	0.741	0.750	0.741	0.701	0.671	0.664
	Health,Energy,Entertainment	0.761	0.749	0.753	0.740	0.655	0.663
	Health,Energy,Safety	0.750	0.751	0.747	0.666	0.645	0.631
	Energy,Entertainment,Other	0.744	0.728	0.721	0.752	0.642	0.616
	Energy,Safety,Other	0.707	0.706	0.687	0.630	0.626	0.613
	Entertainment,Safety,Other	0.664	0.671	0.648	0.706	0.596	0.582
	Health,Safety,Other	0.415	0.522	0.462	0.497	0.507	0.486
	Health,Energy,Other	0.428	0.525	0.458	0.563	0.552	0.552
	Health,Entertainment,Other	0.563	0.556	0.558	0.408	0.520	0.450
Quaternary Clustering	Health,Energy,Entertainment,Safety	0.710	0.708	0.706	0.661	0.596	0.602
	Energy,Entertainment,Safety,Other	0.669	0.660	0.652	0.670	0.570	0.569
	Health,Energy,Safety,Other	0.463	0.540	0.498	0.422	0.502	0.457
	Health,Entertainment,Safety,Other	0.474	0.558	0.504	0.582	0.479	0.485
	Health,Energy,Entertainment,Other	0.490	0.549	0.504	0.432	0.524	0.471
Quinary Clustering	Health,Energy,Entertainment,Safety,Other	0.514	0.551	0.517	0.558	0.466	0.471

7 Usage Scenarios

Since obtaining a large amount of high quality labeled data is extremely difficult in the context of CrowdRE, our approach may be used to provide assistance to the requirements engineers and/or business analysts in carrying out the task of classifying crowd-based requirements into different sectors or application domains. For instance, this technique would be useful in a situation where the labeled data is either not available, e.g., new sectors/domains, or the availability is very limited. Our technique may significantly reduce the amount of time and effort required by the analysts, and if completely automated would also be able to control the subjectivity involved in this process. Another interesting usage scenario would be to combine our approach with ZSL or few-shot learning (FSL) techniques where LLMs are deployed, and use the collective intelligence of these approaches to solve the sector classification problem. Since LLMs suffer from consistency, accuracy, bias, and interpretability related issues, combining LLMs with our approach would make sense if it helps in improving the performance and generating more interpretable (and consistent) results. Finally, it would also be interesting to check if our approach can be used for other requirements related classification problems, e.g., functional vs non-functional, requirements vs non-requirements, and errors vs features etc.

References

[1] Pradeep K Murukannaiah, Nirav Ajmeri, and Munindar P Singh. Acquiring creative requirements from the crowd: Understanding the influences of personality and creative potential in crowd re. In *2016 IEEE 24th International Requirements Engineering Conference (RE)*, pages 176–185. IEEE, 2016.

[2] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297. University of California Press, 1967.

[3] Lior Rokach and Oded Maimon. Clustering methods. In *The Data Mining and Knowledge Discovery Handbook*, pages 321–352. Springer, 2005.

[4] Thorsten Joachims. A probabilistic analysis of the rocchio algorithm with tfidf for text categorization. Technical report, Carnegie-mellon univ pittsburgh pa dept of computer science, 1996.

[5] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26, 2013.

[6] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[7] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the association for computational linguistics*, 5:135–146, 2017.

[8] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[9] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188–1196. PMLR, 2014.

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[11] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*, 2018.

[12] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*, 2019.

[13] Fangxiaoyu Feng, Yinfei Yang, Daniel Cer, Naveen Arivazhagan, and Wei Wang. Language-agnostic bert sentence embedding. *arXiv preprint arXiv:2007.01852*, 2020.

[14] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.