Appendix

1 Background

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

1.1 CrowdRE Dataset

The CrowdRE 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. 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%

1.2 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 (sub-words). 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.

2 Pretrained Model Version and Embeddings Dimension

The 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 2.

Table 2: 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 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.

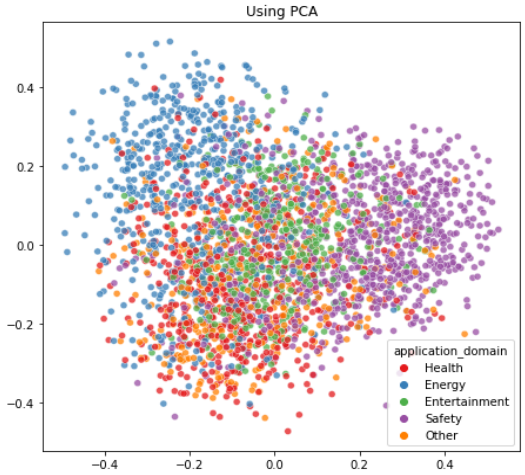

Figure 1: Visualising SRoBERTa embeddings

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.