This is a post-peer-review, pre-copyedit version of this article. The final authenticated version is available online at:
https://doi.org/10.1109/REW53955.2021.00020

Evaluating Unsupervised Text Embeddings on Software User Feedback

Peter Devine, Yun Sing Koh, and Kelly Blincoe
The University of Auckland
Auckland, New Zealand
pdev438@aucklanduni.ac.nz, {y.koh, k.blincoe}@auckland.ac.nz

***Abstract*—User feedback on software products has been shown to be useful for development and can be exceedingly abundant online. Many approaches have been developed to elicit requirements in different ways from this large volume of feedback, including the use of unsupervised clustering, underpinned by text embeddings. Methods for embedding text can vary significantly within the literature, highlighting the lack of a consensus as to which approaches are best able to cluster user feedback into requirements relevant groups. This work proposes a methodology for comparing text embeddings of user feedback using existing labelled datasets. Using 7 diverse datasets from the literature, we apply this methodology to evaluate both established text embedding techniques from the user feedback analysis literature (including topic modelling and word embeddings) as well as text embeddings from state of the art deep text embedding models. Results demonstrate that text embeddings produced by state of the art models, most notably the Universal Sentence Encoder (USE), group feedback with similar requirements relevant characteristics together better than other evaluated techniques across all seven datasets. These results can help researchers select appropriate embedding techniques when developing future unsupervised clustering approaches within user feedback analysis.**

I. Introduction

Explicit online user feedback of software products can be found in many places including in app store reviews, social media posts, and forum posts. Such feedback has been shown to contain information that is useful for requirements engineering (such as bug reports, feature requests, and other data) on many platforms, including Tweets on Twitter [23], reviews on app stores [11], [36], forum posts [47], subreddit posts from Reddit [5], and reviews on Steam [34]. This feedback can be potentially very high volume, with more than 4,000 app reviews being given to the Facebook app in a single day on the Apple App Store [39].

Thus, efforts have been made to automate the analysis of this feedback, such that developers can effectively harness key information for requirements engineering [23] [26] [16] [49] [21]. These efforts have generally tended to come in two forms, *classification* and *clustering*. While the vast majority of classification approaches rely on supervised machine learning, clustering approaches vary more in their methodology, with topic modelling and word frequency-based traditional clustering both popular [33]. Such techniques are underpinned by an embedding of text from natural language into vectors, such that similarity, and thus cluster membership, can be calculated [2]. However, there have been no studies evaluating embedding approaches on their ability to group user feedback by requirements-relevant categories.

State-of-the-art deep learning-based text embedding techniques (such as BERT [15] and USE [10]) have recently been evaluated within the field of requirements engineering for their use in matching similar requirements documents [14] and in matching requirements documents to related source code [35]. These models have not yet been employed for use within publicly published user feedback analysis tools within the literature, but their promise within adjacent fields such as requirement documentation-software matching suggest their adoption in the near future. However, many deep embedding models exist, with there being more than 8,000 available models on the Huggingface model repository at the time of writing[1].

Due to the widespread use of unsupervised clustering techniques in distinguishing semantically separate feedback, there is a clear need to understand which text embedding methods are most appropriate to underpin this clustering.

This work was motivated by one overall research question: **Which text embedding methods are best able to group similar user feedback together?** In order to answer this, we evaluate the ability of both state-of-the-art and established techniques on their ability to match user feedback with requirements related traits in an unsupervised way.

This work has two main contributions. Firstly, we propose a novel approach for evaluating unsupervised text embeddings by using an information retrieval (IR) ranking evaluation approach on class-labelled user feedback datasets from the literature. Secondly, we apply this methodology on 24 text embedding methods, representing 4 broad categories of embedding methods, to find which embedding is best able to group user feedback on the basis of requirements related characteristics. These results show that deep pre-trained embedding models (particularly Google's Universal Sentence Encoder) out-perform all other evaluated methods of text embedding for grouping user feedback, and thus shows that these deep models are appropriate for use in future user feedback clustering tools.

II. Related work

This section details both the state of the art in user feedback embedding methods and the use of deep model embeddings

[1]https://huggingface.co/models

in the field of requirements engineering more broadly.

A. Current requirements extraction methods for user feedback

Many approaches for extracting requirements from user feedback have been developed in the literature. Examples include tools such as SURF [16], CLAP [49], SIMBA [38], and MERIT [21], as well as models without tools associated with them, such as the deep classifier created by Stanik et al. [45].

Approaches like SURF [16], CLAP [49], and the Stanik et al. deep classifier [45] utilise classification approaches, whereby feedback is classified as one or more categories. These categories can be general (e.g., "bug", "feature request") or specific (e.g. "GUI", "Security"). While SURF and CLAP both translate text into features that a machine learning classifier can interpret using word-frequency based methods (namely n-gram extraction), the Stanik et al. classifier uses pre-trained word embeddings based on fastText embedding weights [28]. Classification allows developers to only have to read the types of feedback that are of interest to them. However, these supervised models often fail to generalise from the data domain that they have been trained on to user feedback from other platforms, such as from app reviews to forum posts [47]. These models also only classify data into categories on which they have already been trained.

A different approach to classification can be seen in the SIMBA tool that was proposed by Oehri and Guzman [38]. SIMBA uses a word aligner from the literature [46] to rank user feedback such that similar feedback is ranked higher in order to aid developers in describing requirements (e.g. showing all feedback that describes a specific bug or feature). This ranking was done across four languages and four feedback platforms, evaluating reviews from Google Play Store and the Apple App Store, Tweets, and Facebook comments. This matching yielded high correlations of 0.78 and 0.79 to labelled similarity oracles for monolingual and multilingual datasets respectively, indicating its usefulness in matching alike requirements across languages and platforms. While this approach is useful for ranking a corpus against a single query document, mutual distances need to be calculated for all pairs of feedback for clustering approaches, and applying the world aligner to all possible pairwise combinations would be computationally intensive, making scaling to large datasets challenging.

Other tools have been proposed that use unsupervised clustering to break user feedback down into organically defined groups of semantically linked information (e.g. CLAP [49] and MERIT [21]). MERIT's approach allows developers to see the main themes of what their users are saying, as defined by a few key pieces of feedback per theme, and their relative changes over time. CLAP is designed to allow developers to go beyond simple classification and see not only all their feedback which is classed as a "bug" or "feature request", but feedback grouped into smaller clusters which describe individual bugs or features. MERIT uses a bi-term topic modelling approach to quantify user feedback, while CLAP employs an n-gram word frequency-based approach to embed text before clustering. These methods are unsupervised, meaning that they do not require a labelled dataset for training, and so can be used by developers in novel domains immediately. The benefits of unsupervised techniques mean that a developer can apply them anywhere without preparation or training data, reducing the overhead of deployment. Both approaches rely on text embeddings, which do not explicitly encode concepts of feedback type (bug types, feature types, etc.) into them. Little work exists in measuring which embedding technique is thus most suitable for disambiguating various types of user feedback available.

Indeed, there exists no consensus as to which unsupervised technique is most appropriate for grouping user feedback, as is evidenced by the sheer variety of clustering approaches within the literature [33]. Due to the fact that the embedding of text into machine readable values is pivotal to the success of clustering techniques, we aim to determine which technique is most appropriate for grouping user feedback, and whether the techniques used within the literature can be improved upon.

B. Embeddings from deep text embedding models and their use in requirement engineering

The state of the art in text embeddings in the field of NLP are currently generated by transformer based pre-trained text embedding models, as is evidenced by their dominance in the Semantic Text Similarity Benchmark (STS-b)[2]. Included in these are Sentence-BERT (SBERT) [43] and the Universal Sentence Encoder (USE) [10]. While SBERT has been trained to explicitly encode semantically similar texts close together, USE has been trained on a variety of NLP tasks, such that it is able to generate general text embeddings that would be useful for many different applications. LaBSE [18] is a cross-lingual embedding model based upon the BERT model, and is trained to embed semantically similar texts in different languages closely within embedding space.

The usefulness of these models stems from their architecture and their pre-training. The transformer [48] based architecture means that these models can interpret a piece of text as not just a bag of words, but as a bi-directional sequence of words, giving these models the ability to capture more intricate details from a given piece of text. Pre-training on masses of text means that these models also contain a modelling of language which manifests itself within the embedding.

Recent studies within the field of requirements engineering have evaluated some deep embedding models. Araujo et al. evaluated the word frequency bag-of-words (BOW) and term frequency inverse document frequency (TF-IDF) text embedding techniques against several deep pre-trained embedding models including BERT on their performance as inputs into a supervised machine learning classification model [6]. This evaluation was done on a labelled dataset of app reviews from Maalej et al. [36]. They found that classification F1 scores

[2]https://paperswithcode.com/sota/semantic-textual-similarity-on-sts-benchmark

were higher when using embeddings from deep pre-trained embedding models as input to a classifier compared to the word frequency embeddings.

Abbas et al. investigated the ability of several unsupervised text embedding techniques including TF-IDF and BERT to determine similarity between requirements documentation and source code [1]. This research found that TF-IDF had the best performance in linking similar requirements to source code. Lin et al. also investigated the ability of BERT models to link requirements documentation to source code, but trained several models on this linking task instead of using unsupervised embeddings [35]. Das et al. investigated the ability of several supervised and unsupervised deep embedding models, including USE and BERT, on their ability to match similar requirements documents [14]. Out of the results of the unsupervised models reported in this work, the USE model was found to work best.

While these models have been evaluated on general NLP benchmarks, supervised user feedback classification, and unsupervised requirements documentation matching, an evaluation of these models for the purposes of unsupervised user feedback clustering has not been carried out. This work seeks to compare the previously established approaches for embedding user feedback to these state-of-the-art text embeddings in order to identify which techniques are most effective at collecting user feedback into requirements relevant groups.

III. Method

In order to evaluate which text embedding methods are most appropriate for use in user feedback analysis tools, seven datasets from the literature were used to evaluate both established and state-of-the-art text embedding techniques. The following sections detail the datasets used for evaluation, how the evaluation was done, what embedding methods were tested, what the distance metrics used in the evaluation were, and the baselines from which embedding performance was measured.

A. Datasets

Seven class labelled datasets from the literature were used to evaluate text embeddings. These datasets were taken from the works of Chen et al. [11], Guzman et al. [22], Maleej et al. [36], Ciurumelea et al. [13], Scalabrino et al. [44], Williams et al. [50], and Tizard et al. [47]. These datasets were chosen due to their public availability, as well as their diversity in feedback platforms (online user feedback from Apple App Store, Google Play Store, Twitter, and forum posts are all included), number of apps, and label set. This heterogeneity between datasets, in both the type of feedback and the linguistic character of the feedback they contain, was prioritised in order to evaluate the ability of the embeddings to distinguish between multiple different types of feedback in multiple contexts. Thus, these heterogeneous datasets can help determine if a suitable embedding method can be applied across different contexts.

Feedback was considered on an app-level for these datasets, meaning that all feedback on one app from a dataset was embedded, and these embeddings were analysed in relation to each other. These datasets were also cleaned for evaluation. Apps with less than or equal to 10 pieces of feedback were excluded so as to focus on measuring the performance of embeddings on apps with large amounts of feedback - where manual reading of all feedback is less feasible. Apps were also only included if they contained feedback with different labels (i.e. all feedback for a given app did not have the same label) so as to evaluate embedding techniques' ability to group distinct feedback. By applying these constraints on the datasets, 10 apps (40 pieces of feedback) were excluded from the Ciurumelea et al. dataset [13], 458 apps (973 pieces of feedback) from the Maalej et al. dataset [36], and 1 app (5 pieces of feedback) from the Scalabrino at al. dataset [44]. There was no feedback excluded from the other 4 datasets.

Post-cleaning metrics and information for each of these datasets can be seen in Table I.

B. Embeddings

The types of text embeddings that were evaluated can be broadly collected into 4 groups: topic modelling, averaged word embeddings, word frequency embeddings, and pre-trained text embedding model embeddings. These groups of embeddings were chosen due to their use within the user feedback literature as unsupervised text embeddings (topic modelling, averaged word embeddings, word frequency embeddings), or due to their proven ability in matching similarity of other requirements related text (pre-trained deep model embeddings) [1] [14].

Topic modelling embeddings are generated by applying the entire corpus of feedback for one app to the model to generate topic distributions for the app. Topic models are trained on an unlabelled text corpus, where they separate words or n-grams into a given number of topics using term co-occurrence in such a way that terms that often co-occur are more likely to share the same topic. Therefore, every topic is characterised by its probability distribution over terms within the corpus, and thus so too are documents characterised over topics by their constituent terms. We use these probability distributions as the encoding for a given piece of feedback.

The three topic modelling approaches that were tested were LDA [9], BTM [51] and Gibbs sampling algorithm for a Dirichlet Mixture Model (GSDMM) [52]. LDA was chosen due to its abundance within the literature [24] [26] [37] [31], BTM was chosen due to its proven good performance in modelling shorter text which user feedback often is [23] [12], and GSDMM was chosen due to its good performance and taxonomic difference in kind as a short text modeller compared to BTM [42]. These models were also chosen due to their simple implementations in Python, from the Gensim[3], Biterm[4]

[3]https://radimrehurek.com/gensim/

[4]https://github.com/markoarnauto/biterm

TABLE I
DETAILS OF THE SEVEN CLASS-LABELLED USER FEEDBACK DATASETS USED FOR EVALUATING TEXT EMBEDDING METHODS

	Source	**Feedback platform**	**Number of apps**	**Label set**	**Dataset size**
A	Chen et al.[11]	Google Play Store reviews	4	Informative, Non-informative	11,340
B	Ciurumlelea et al.[13]	Google Play Store reviews	17	Resources, Usage, Compatibility, Pricing, Protection, Other	1,538
C	Guzman et al.[24]	Google Play Store reviews	7	Bug report, Noise, Usage scenario, Praise, Complaint, Feature shortcoming, Feature strength, User request	4,401
D	Maalej et al.[36]	Google Play Store reviews, Apple App Store reviews	24	Bug, Feature, User experience, Rating	488
E	Scalabrino et al.[44]	Google Play Store reviews	13	Feature, Performance, Usability, Security, Energy, Bug	702
F	Tizard et al.[47]	Forum posts	3 (2 apps, 3 forums)	User setup, Question on application, Requesting more information, Feature request, Non-informative, Malfunction confirmation, Question on background, Help seeking, Attempted solution, Application usage, Praise for application, Acknowledgement of problem resolution, Agreeing with the feature request, Agreeing with the problem, Limitation confirmation, Application guidance, Dispraise for application, Apparent bug, Other	3,654
G	Williams et al.[50]	Twitter posts	10	Feature, Bug, Other	3,654

and GSDMM[5] packages for LDA, BTM, and GSDMM respectively.

To create the topic models, three configurations were used for the topic number hyperparameter, 5 topics, 13 topics, and 50 topics. The 5 and 50 parameters were chosen as an order of magnitude difference between each other, while 13 was chosen due to the findings by Gao et al. that that configuration was most suitable for the BTM model on app review data [21]. Other hyperparameters (e.g. α and β) were all set to their default from their Python implementation.

Averaged word embeddings were generated by calculating word embedding for each word in a given piece of feedback and averaging these together to get a single embedding for every piece of feedback. The word embeddings used were GloVe [40] (both the 6 billion token and 840 billion token variants), ExtVec [29], a modified version of word2vec from Levy & Goldberg [30], and the unsupervised smoothed inverse frequency (USIF) method from Kawin Ethayarajh [17]. GloVe, ExtVec, and Levy & Goldberg's word2vec were chosen due to their large vocabulary size (roughly 400,000, 250,000, and 175,000 words, respectively), their adoption within the NLP literature [3] [7] [25], and their availability in the SentenceTransformers Python package[6]. USIF was selected due to its relatively high performance on semantic similarity tasks and due to the fact that it does not need hyperparamater tuning to function, thus making it suitable for unsupervised embedding [17].

The **word frequency embeddings** tested were bag-of-words (BOW) and term frequency inverse document frequency (TF-IDF) [27] due to their use within the requirements elicitation literature [49] [4] [8]. Approaches that both kept and removed stop words [19] from the text were tested, using the English stop-word list from the NLTK package[7]. With the stop words removed model, a uni- and bi-gram model was also tested for both BOW and TF-IDF, with a minimum term frequency for words within the corpus set at 2.

Pre-trained deep model embeddings were generated from three different text embedding models, SBERT, USE, and LaBSE. These models were chosen due to their wide use within the literature [32] [41], and their high performance on the Semantic Text Similarity Benchmark. Moreover, USE was shown to be good at classifying similar requirements documents [14]. LaBSE was included due to the fact that it is trained to semantically embed text cross-lingually for over 100 languages, thus adding to the diversity in pre-training regimes of deep models tested. Two variants of SBERT available from SentenceTransformers were used, nli-bert-large (S-BERT) and nli-roberta-large (S-RoBERTa), as well as the SentenceTransformers implentation of LaBSE. The large version of USE available on the Tensorflow Hub[8] was used.

A **random baseline** was used to contextualise the performance of the above embedding techniques. This random

[5]https://github.com/rwalk/gsdmm
[6]https://www.sbert.net/
[7]https://www.nltk.org/
[8]https://tfhub.dev/google/universal-sentence-encoder-large/5

embedding was an array of 50 dimensional vectors of random numbers between 0 and 1 for each piece of feedback generated by the NumPy random package[9].

C. Distance Metrics

Due to the variety in the types of output from the embedding models, different distance metrics were used to compare semantic relatedness from the outputs. For the pre-trained embedding models and averaged word embedding models, the cosine distance of their outputs was used in the similarity ranking calculation, while the Jensen-Shannon distance was used for topic models and word frequency methods, which were found to give the highest MRR and mean NDCG scores for each method. Because cosine similarity was used to train some of the pre-trained embedding models (such as LaBSE and S-BERT), it matches with intuition that this distance metric is appropriate for the pre-trained models. Moreover, as topic models output a probability distribution over topics for each document, it is again intuitive that measuring distance between two distributions with the Jensen-Shannon metric (a metric for expressing the distance between two probability distributions) is the best performing. The Jensen Shannon distance has been used in user feedback literature in the past by Gao et al. [20], but to compare term distributions over topics, instead of topic distributions over documents.

D. Evaluation approach

Following the approaches of Lin et al. [35], we treated this similarity matching problem as an information retrieval (IR) task. For each piece of feedback, we hold that feedback as a query and every other piece of feedback as possible documents to match with this query. We considered the "document" feedback a match if it shared a label with the query feedback, where the labels are taken from the seven ground truth evaluation datasets.

To evaluate each embedding method, we employed the standard metrics of mean reciprocal rank (MRR) and the mean of the normalised discounted cumulative gain (NDCG) metrics. MRR is calculated by ranking all feedback in relation to one piece of query feedback, f_i. All feedback except f_i are ranked from 1 to $N-1$ for a dataset with N pieces of feedback, based on the distance of their embeddings to f_i, with the closest feedback being ranked highest. Then, the rank of the first piece of feedback in the rankings to share at least one class label (e.g. "bug", "feature request") with f_i is found. This ranking is defined as $FirstRank_i$, and its reciprocal is then calculated. This process is repeated for every piece of feedback and the mean of these values is the MRR score. This value has a maximum at 1, which represents perfect ranking. The equation for MRR can be seen in Equation 1.

$$MRR = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{FirstRank_i} \tag{1}$$

[9]https://numpy.org/doc/stable/reference/random/index.html

Similarly, NDCG also ranks all feedback based on embedding distance relative to f_i. Firstly, the discounted cumulative gain (DCG) is calculated for f_i by the sum of gains for each ranked piece of feedback within the ranking (Equation 2). The gain for a piece of feedback at rank j in relation to f_i is calculated using rel_j, which is 1 if this feedback shares labels with f_i, and 0 otherwise. This gain is then the fraction of $\frac{2^{rel_j}-1}{log_2(j+1)}$, which effectively scores similar feedback with a high ranking highly, and dissimilar feedback as 0. The ideal DCG (IDCG) is calculated by calculating the DCG if the feedback was ranked perfectly (i.e. all feedback that share labels with f_i are ranked higher than those that do not), as can be seen in Equation 3. Finally, the NDCG is the ratio of the DCG to IDCG, which served to give a normalised measure with a maximum at 1, representing perfect ranking. This is then calculated for all feedback, before taking the mean of these values as the mean NDCG (Equation 4).

$$DCG_i = \sum_{j=1}^{p_i}\frac{2^{rel_j}-1}{log_2(j+1)} \tag{2}$$

$$IDCG_i = \sum_{k=1}^{p_i^{ideal}}\frac{2^{rel_k}-1}{log_2(k+1)} \tag{3}$$

$$MeanNDCG = \frac{1}{N}\sum_{i=1}^{N}\frac{DCG_i}{IDCG_i} \tag{4}$$

These metrics were calculated for all feedback within one app within each dataset. Since each dataset has varying numbers of apps, and each app has a varying number of pieces of feedback, these metrics were averaged across all apps within a dataset, and these averages are reported.

IV. Evaluation

The MRR and mean NDCG over the seven evaluation datasets can be seen in Table II and Table III, respectively. Table II shows that the MRR of the four deep text embedding models evaluated are on average higher than all other embedding methods in the evaluation, while table III shows that the NDCG of these models is also higher than all other approaches. We can see that deep embedding models perform better at grouping user feedback into requirement relevant groups over all datasets compared to established techniques based on averaged word embeddings, word frequency methods, and topic modelling. Overall, we can see that USE performs best on the largest number of datasets (4 out of 7) across both metrics. Of particular note is this model's performance across datasets of user feedback from varied platforms. USE has the highest MRR for 3 of the 5 app review datasets, is in the top three for the other 2 review datasets. It also has the highest MRR for the Twitter user feedback dataset, and is second highest for the forum user feedback dataset. Similarly for NDCG, USE performs best on the Twitter and forum user feedback datasets, and is in the top three for all other datasets. In summary, USE is best

TABLE II
MRR OF ALL EMBEDDING MODELS OVER ALL EVALUATION DATASETS. BOLDED VALUES ARE THE HIGHEST BETWEEN EMBEDDING METHODS ON THE GIVEN DATASET. DATASET LETTERS REFER TO DATASETS FROM: A - CHEN ET AL. (2014) [11], B - CIURUMLELEA ET AL. (2017) [13], C - GUZMAN ET AL. (2014) [24], D - MAALEJ ET AL. (2016) [36], E - SCALABRINO ET AL. (2017) [44], F - TIZARD ET AL. (2019) [47], G - WILLIAMS ET AL. (2017) [50]

		Dataset							
	Model type	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**Mean**
USE	**Deep model**	**0.941**	0.897	**0.865**	0.872	**0.831**	0.665	**0.817**	**0.841**
S-RoBERTa	**Deep model**	0.919	0.892	0.859	**0.888**	0.787	0.642	0.782	0.824
S-BERT	**Deep model**	0.910	0.889	0.847	0.885	0.781	0.636	0.770	0.817
LaBSE	**Deep model**	0.920	**0.899**	0.828	0.853	0.772	**0.674**	0.764	0.816
TF-IDF	**Word frequency**	0.902	0.879	0.783	0.823	0.779	0.601	0.758	0.789
USIF	**Word embeddings**	0.905	0.886	0.816	0.838	0.758	0.582	0.727	0.787
BOW	**Word frequency**	0.889	0.878	0.770	0.812	0.780	0.594	0.740	0.780
GloVe (840B, 300D)	**Word embeddings**	0.891	0.873	0.807	0.834	0.758	0.582	0.713	0.780
GloVe (6B, 300D)	**Word embeddings**	0.891	0.879	0.808	0.818	0.773	0.556	0.710	0.776
TF-IDF (Stopwords removed)	**Word frequency**	0.895	0.865	0.758	0.812	0.788	0.561	0.742	0.775
Komninos	**Word embeddings**	0.892	0.887	0.812	0.799	0.758	0.559	0.715	0.775
1,2-gram TF-IDF (Stopwords removed)	**Word frequency**	0.889	0.869	0.758	0.825	0.757	0.574	0.744	0.774
1,2-gram BOW (Stopwords removed)	**Word frequency**	0.882	0.868	0.747	0.824	0.767	0.560	0.732	0.769
Levy	**Word embeddings**	0.890	0.876	0.815	0.785	0.749	0.563	0.706	0.769
BOW (Stopwords removed)	**Word frequency**	0.887	0.863	0.738	0.814	0.779	0.549	0.729	0.766
Biterm (T=50)	**Topic modelling**	0.857	0.835	0.722	0.817	0.741	0.487	0.679	0.734
GSDMM (T=50)	**Topic modelling**	0.861	0.803	0.736	0.766	0.736	0.497	0.686	0.726
GSDMM (T=13)	**Topic modelling**	0.863	0.791	0.749	0.777	0.743	0.472	0.679	0.725
GSDMM (T=5)	**Topic modelling**	0.850	0.823	0.735	0.796	0.696	0.451	0.675	0.718
Biterm (T=13)	**Topic modelling**	0.842	0.840	0.705	0.808	0.701	0.456	0.652	0.715
Biterm (T=5)	**Topic modelling**	0.818	0.815	0.652	0.789	0.688	0.450	0.626	0.691
LDA (T=50)	**Topic modelling**	0.793	0.798	0.632	0.753	0.664	0.449	0.642	0.676
LDA (T=5)	**Topic modelling**	0.787	0.786	0.642	0.785	0.634	0.439	0.632	0.672
LDA (T=13)	**Topic modelling**	0.796	0.764	0.640	0.771	0.636	0.426	0.609	0.663
Random baseline	**Random**	0.730	0.753	0.555	0.774	0.636	0.354	0.590	0.627

able to group requirements-relevant user feedback from the test datasets out of all embedding methods evaluated.

LaBSE also performs best on two datasets for the MRR ranking metric, and is fourth on average for both metrics. S-RoBERTA performs best on 1 dataset for MRR, and three for NDCG, and is overall second in the average ranking for both. More broadly, we can see that the deep models perform better across all datasets compared to other methods, with all four deep embedding models tested making up the top four of the embedding techniques on average for both MRR and NDCG metrics.

In comparison, the word frequency embedding TF-IDF and the word embedding-based USIF methods are the next best text embeddings in our evaluation. With the results from deep models excluded, TF-IDF performed best on 3 out of 7 datasets for MRR and 2 out of 7 for NDCG, while USIF also performed best on 3 out of 7 datasets for MRR and 2 out of 7 for NDCG.

In contrast, the topic modelling methods of GSDMM, BTM, and LDA underperform these methods. Overall, we can see that topic modelling approaches that are suited to shorter text (such as the Biterm model and GSDMM) perform better over both metrics for all datasets compared to the LDA model.

V. DISCUSSION

A. Reflection on results

As can be seen from section IV, deep text embedding models are better at embedding feedback from our test datasets into groups that are relevant to requirements compared to other evaluated techniques. Therefore, these embedding methods stand to aid many of the future tools for extracting requirement information from user feedback, or improve on existing tools that already leverage text embeddings, such as CLAP[49]. Of particular note is the superior performance across a wide range of heterogeneous datasets. This result suggests that deep neural networks such as USE are an effective embedding method for user feedback from multiple sources, and so could potentially be used in tools that analyse user feedback from multiple platforms for a single app.

The relatively strong performance of LaBSE is another notable result, because this model is trained to produce cross lingual embeddings where sentences of different languages but equivalent semantic meaning are mapped into the same embedding space. Due to the multilingual nature of its training, it could potentially perform similarly on other languages.

While deep models achieve better results in our evaluation compared to other methods, they require being run on a GPU to embed them at maximum efficiency. Out of the models which do not benefit from being run on a GPU, TF-IDF and USIF performed best. Therefore, in situations where using a deep model is not feasible, TF-IDF or USIF could also be considered.

Within the topic model results, we can see that the strength of approaches that are more suited to short texts perform better in our evaluation, which validates the recent widespread adoption of the Biterm model over LDA within the user feedback field when topic modelling. Overall, embeddings derived

TABLE III

NDCG OF ALL EMBEDDING MODELS OVER ALL EVALUATION DATASETS. BOLDED VALUES ARE THE HIGHEST BETWEEN EMBEDDING METHODS ON THE GIVEN DATASET. DATASET LETTERS REFER TO DATASETS FROM: A - CHEN ET AL. (2014) [11], B - CIURUMLELEA ET AL. (2017) [13], C - GUZMAN ET AL. (2014) [24], D - MAALEJ ET AL. (2016) [36], E - SCALABRINO ET AL. (2017) [44], F - TIZARD ET AL. (2019) [47], G - WILLIAMS ET AL. (2017) [50]

		Dataset							
	Model type	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**Mean**
USE	**Deep model**	0.941	**0.904**	0.871	0.886	**0.836**	**0.744**	**0.855**	**0.862**
S-RoBERTa	**Deep model**	**0.944**	0.894	**0.888**	**0.907**	0.812	0.732	0.842	0.860
S-BERT	**Deep model**	0.942	0.894	0.883	0.901	0.804	0.728	0.836	0.855
LaBSE	**Deep model**	0.932	0.903	0.853	0.875	0.797	0.738	0.832	0.847
USIF	**Word embeddings**	0.928	0.892	0.844	0.856	0.791	0.697	0.815	0.832
GloVe (840B, 300D)	**Word embeddings**	0.923	0.888	0.842	0.852	0.785	0.706	0.811	0.830
TF-IDF	**Word frequency**	0.922	0.894	0.831	0.844	0.791	0.698	0.821	0.829
GloVe (6B, 300D)	**Word embeddings**	0.923	0.888	0.843	0.847	0.794	0.695	0.811	0.829
Komninos	**Word embeddings**	0.925	0.893	0.843	0.840	0.789	0.692	0.813	0.828
Levy	**Word embeddings**	0.924	0.890	0.843	0.839	0.785	0.693	0.811	0.827
BOW	**Word frequency**	0.921	0.894	0.826	0.839	0.791	0.695	0.819	0.826
TF-IDF (Stopwords removed)	**Word frequency**	0.920	0.886	0.824	0.838	0.800	0.685	0.819	0.825
1,2-gram TF-IDF (Stopwords removed)	**Word frequency**	0.920	0.888	0.823	0.839	0.791	0.686	0.820	0.824
BOW (Stopwords removed)	**Word frequency**	0.919	0.884	0.819	0.841	0.795	0.684	0.817	0.823
1,2-gram BOW (Stopwords removed)	**Word frequency**	0.919	0.889	0.820	0.838	0.791	0.684	0.818	0.823
GSDMM (T=5)	**Topic modelling**	0.933	0.863	0.834	0.839	0.769	0.670	0.810	0.817
Biterm (T=50)	**Topic modelling**	0.925	0.877	0.811	0.843	0.783	0.667	0.808	0.816
GSDMM (T=50)	**Topic modelling**	0.929	0.853	0.825	0.833	0.780	0.678	0.811	0.815
GSDMM (T=13)	**Topic modelling**	0.930	0.849	0.829	0.832	0.780	0.667	0.810	0.814
Biterm (T=13)	**Topic modelling**	0.925	0.875	0.805	0.841	0.773	0.665	0.797	0.812
Biterm (T=5)	**Topic modelling**	0.924	0.866	0.792	0.835	0.760	0.662	0.792	0.804
LDA (T=5)	**Topic modelling**	0.910	0.844	0.784	0.827	0.737	0.660	0.792	0.794
LDA (T=50)	**Topic modelling**	0.910	0.845	0.776	0.828	0.741	0.654	0.792	0.792
LDA (T=13)	**Topic modelling**	0.910	0.835	0.776	0.832	0.733	0.652	0.789	0.790
Random baseline	**Random**	0.903	0.838	0.768	0.821	0.738	0.641	0.784	0.785

from topic modelling performed relatively poorly compared to other techniques in our evaluation. This is because topic modelling approaches are most commonly used to generate an overview of topics within a corpus, rather than to categorize individual pieces of feedback. Therefore, we can see that topic modelling may not be appropriate for this particular task compared to other state of the art and established text embedding techniques.

B. Future work

With this work, we evaluated whether similar individual pieces of feedback could be embedded close to each other based on coarse labels. These labels group requirement-related user feedback at a high level (e.g. bug reports and feature requests). Future work could examine whether these embeddings can also be used to group requirement-related information into more fine-grained groupings (e.g. all user feedback related to a specific software feature or even a specific requirement).

Future work could also implement the approach used by CLAP, where user feedback is first classified into broad requirements relevant categories (E.g. "Bug", "Feature request") before being clustered using one of the evaluated techniques in order to better segment the data. Such work could evaluate how preliminary classification of user feedback affects the ability of embedding methods to group related feedback together.

C. Threats to validity

A threat to validity of this work is whether our coverage of feedback platforms and label sets in our test datasets gives a useful representation of user feedback and its associated requirements relevant characteristics. This threat was minimised by evaluating on as a wide a range of datasets as is publicly available in the field of software user feedback analysis, with a variety of different feedback platforms and label sets included. However, as new datasets describing user feedback (such as those describing feedback on a new platform or with a new label set) are created and become publicly available, it is for future work to add these datasets to the evaluation of embedding techniques.

Another threat is whether the comprehensiveness of the embedding methods evaluated effectively represents the methods available to embed text for clustering. Again, as wide a set of available embedding methods as possible was chosen for evaluation. However, this is not an exhaustive list of every possible embedding method, and as new embedding methods become available, it is for future work to evaluate these embeddings.

Varied distance measures were used for different embedding methods. This makes for an imperfect comparison between embedding methods, and thus is a potential threat to validity. We used varied distance measures due to the varying suitability of these measures for different embedding types (see Section III-C for more), and the highest overall performing metric was used for each embedding type. This was done to mitigate threats to validity when comparing a diverse set of text embeddings.

Further, all of the datasets used for evaluation within our

evaluation are in English. This is due to the relative abundance of English language datasets compared to other languages in the field of analysing user feedback. Therefore, the results of our evaluation can only be assumed to extend to that of English user feedback. However, many of the techniques tested, such as word frequency and topic modelling methods are language agnostic, meaning that they could be tested on datasets of other languages if available. Moreover, the USE model has a cross-lingual embedding version[10], and LaBSE is trained to produce cross-lingual embeddings. Therefore, these models could potentially be useful for embedding user feedback in other languages. Future work could apply these models to user feedback written in other languages to examine the generalisability of our results.

VI. Conclusion

We present an evaluation of text embedding techniques on software user feedback in order to determine which techniques are most appropriate for use in feedback clustering and ranking tasks that exist within the literature. This evaluation was done over 7 datasets from the literature and on 4 different types of embeddings. While existing embedding techniques such as TF-IDF performed well, the performance of pre-trained deep embedding models such as USE exceeded all existing techniques from the literature.

The demonstrated superior performance of these pre-trained models across multiple domains of user feedback can inform the construction of future tools, particularly tools which seek to cluster or rank feedback. The best performing embedding techniques could also be paired with a use-case appropriate text classifier for potentially even better user feedback analysis tools.

The replication package for the experiments within this paper can be found at https://doi.org/10.5281/zenodo.5183351.

References

[1] Muhammad Abbas, Alessio Ferrari, Anas Shatnawi, and Eduard Paul. Is requirements similarity a good proxy for software similarity? an empirical investigation in industry. In *The 27th International Working Conference on Requirements Engineering: Foundation for Software Quality*, 2021.

[2] Charu C Aggarwal and ChengXiang Zhai. A survey of text clustering algorithms. In *Mining text data*, pages 77–128. Springer, 2012.

[3] Alan Akbik, Duncan Blythe, and Roland Vollgraf. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th international conference on computational linguistics*, pages 1638–1649, 2018.

[4] Afnan A Al-Subaihin, Federica Sarro, Sue Black, Licia Capra, Mark Harman, Yue Jia, and Yuanyuan Zhang. Clustering mobile apps based on mined textual features. In *Proceedings of the 10th ACM/IEEE international symposium on empirical software engineering and measurement*, pages 1–10, 2016.

[5] Javed Ali Khan, Lin Liu, Lijie Wen, and Raian Ali. Conceptualising, extracting and analysing requirements arguments in users' forums: The crowdre-arg framework. *Journal of Software: Evolution and Process*, 32(12):e2309, 2020.

[6] Adailton Araujo, Marcos Golo, Breno Viana, Felipe Sanches, Roseli Romero, and Ricardo Marcacini. From bag-of-words to pre-trained neural language models: Improving automatic classification of app reviews for requirements engineering. In *Anais do XVII Encontro Nacional de Inteligência Artificial e Computacional*, pages 378–389. SBC, 2020.

[10]https://tfhub.dev/google/universal-sentence-encoder-xling-many/1

[7] Sanjeev Arora, Yingyu Liang, and Tengyu Ma. A simple but tough-to-beat baseline for sentence embeddings. 2016.

[8] Noor Hasrina Bakar, Zarinah M Kasirun, Norsaremah Salleh, and AHA Halim. Extracting software features from online reviews to demonstrate requirements reuse in software engineering. In *Proceedings of the International Conference on Computing & Informatics*, pages 184–190, 2017.

[9] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.

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

[11] Ning Chen, Jialiu Lin, Steven CH Hoi, Xiaokui Xiao, and Boshen Zhang. Ar-miner: mining informative reviews for developers from mobile app marketplace. In *Proceedings of the 36th international conference on software engineering*, pages 767–778, 2014.

[12] Tse-Hsun Chen, Stephen W Thomas, and Ahmed E Hassan. A survey on the use of topic models when mining software repositories. *Empirical Software Engineering*, 21(5):1843–1919, 2016.

[13] Adelina Ciurumelea, Andreas Schaufelbühl, Sebastiano Panichella, and Harald C Gall. Analyzing reviews and code of mobile apps for better release planning. In *2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pages 91–102. IEEE, 2017.

[14] Souvick Das, Novarun Deb, Agostino Cortesi, and Nabendu Chaki. Sentence embedding models for similarity detection of software requirements. *SN Computer Science*, 2(2):1–11, 2021.

[15] 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.

[16] Andrea Di Sorbo, Sebastiano Panichella, Carol V Alexandru, Corrado A Visaggio, and Gerardo Canfora. Surf: summarizer of user reviews feedback. In *2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)*, pages 55–58. IEEE, 2017.

[17] Kawin Ethayarajh. Unsupervised random walk sentence embeddings: A strong but simple baseline. *ACL 2018*, page 91, 2018.

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

[19] Christopher Fox. A stop list for general text. In *Acm sigir forum*, volume 24, pages 19–21. ACM New York, NY, USA, 1989.

[20] Cuiyun Gao, Jichuan Zeng, Michael R Lyu, and Irwin King. Online app review analysis for identifying emerging issues. In *Proceedings of the 40th International Conference on Software Engineering*, pages 48–58, 2018.

[21] Cuiyun Gao, Jichuan Zeng, Zhiyuan Wen, David Lo, Xin Xia, Irwin King, and Michael R Lyu. Emerging app issue identification via online joint sentiment-topic tracing. *arXiv preprint arXiv:2008.09976*, 2020.

[22] Emitza Guzman, Muhammad El-Haliby, and Bernd Bruegge. Ensemble methods for app review classification: An approach for software evolution (n). In *2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 771–776. IEEE, 2015.

[23] Emitza Guzman, Mohamed Ibrahim, and Martin Glinz. A little bird told me: Mining tweets for requirements and software evolution. In *2017 IEEE 25th International Requirements Engineering Conference (RE)*, pages 11–20. IEEE, 2017.

[24] Emitza Guzman and Walid Maalej. How do users like this feature? a fine grained sentiment analysis of app reviews. In *2014 IEEE 22nd international requirements engineering conference (RE)*, pages 153–162. IEEE, 2014.

[25] Felix Hill, Roi Reichart, and Anna Korhonen. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, 41(4):665–695, 2015.

[26] Claudia Iacob and Rachel Harrison. Retrieving and analyzing mobile apps feature requests from online reviews. In *2013 10th working conference on mining software repositories (MSR)*, pages 41–44. IEEE, 2013.

[27] Karen Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 1972.

[28] Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hérve Jégou, and Tomas Mikolov. Fasttext. zip: Compressing text classification models. *arXiv preprint arXiv:1612.03651*, 2016.

[29] Alexandros Komninos and Suresh Manandhar. Dependency based embeddings for sentence classification tasks. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*, pages 1490–1500, 2016.

[30] Omer Levy and Yoav Goldberg. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 302–308, 2014.

[31] Tong Li, Fan Zhang, and Dan Wang. Automatic user preferences elicitation: A data-driven approach. In *International Working Conference on Requirements Engineering: Foundation for Software Quality*, pages 324–331. Springer, 2018.

[32] Yuliang Li, Jinfeng Li, Yoshihiko Suhara, AnHai Doan, and Wang-Chiew Tan. Deep entity matching with pre-trained language models. *arXiv preprint arXiv:2004.00584*, 2020.

[33] Sachiko Lim, Aron Henriksson, and Jelena Zdravkovic. Data-driven requirements elicitation: A systematic literature review. *SN Computer Science*, 2(1):1–35, 2021.

[34] Dayi Lin, Cor-Paul Bezemer, Ying Zou, and Ahmed E Hassan. An empirical study of game reviews on the steam platform. *Empirical Software Engineering*, 24(1):170–207, 2019.

[35] Jinfeng Lin, Yalin Liu, Qingkai Zeng, Meng Jiang, and Jane Cleland-Huang. Traceability transformed: Generating moreaccurate links with pre-trained bert models. *arXiv preprint arXiv:2102.04411*, 2021.

[36] Walid Maalej, Zijad Kurtanović, Hadeer Nabil, and Christoph Stanik. On the automatic classification of app reviews. *Requirements Engineering*, 21(3):311–331, 2016.

[37] Maleknaz Nayebi, Henry Cho, and Guenther Ruhe. App store mining is not enough for app improvement. *Empirical Software Engineering*, 23(5):2764–2794, 2018.

[38] Emanuel Oehri and Emitza Guzman. Same same but different: Finding similar user feedback across multiple platforms and languages. In *2020 IEEE 28th International Requirements Engineering Conference (RE)*, pages 44–54. IEEE, 2020.

[39] Dennis Pagano and Walid Maalej. User feedback in the appstore: An empirical study. In *2013 21st IEEE international requirements engineering conference (RE)*, pages 125–134. IEEE, 2013.

[40] 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.

[41] Christian S Perone, Roberto Silveira, and Thomas S Paula. Evaluation of sentence embeddings in downstream and linguistic probing tasks. *arXiv preprint arXiv:1806.06259*, 2018.

[42] Jipeng Qiang, Zhenyu Qian, Yun Li, Yunhao Yuan, and Xindong Wu. Short text topic modeling techniques, applications, and performance: a survey. *IEEE Transactions on Knowledge and Data Engineering*, 2020.

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

[44] Simone Scalabrino, Gabriele Bavota, Barbara Russo, Massimiliano Di Penta, and Rocco Oliveto. Listening to the crowd for the release planning of mobile apps. *IEEE Transactions on Software Engineering*, 45(1):68–86, 2017.

[45] Christoph Stanik, Marlo Haering, and Walid Maalej. Classifying multilingual user feedback using traditional machine learning and deep learning. In *2019 IEEE 27th International Requirements Engineering Conference Workshops (REW)*, pages 220–226. IEEE, 2019.

[46] Md Arafat Sultan, Steven Bethard, and Tamara Sumner. Back to basics for monolingual alignment: Exploiting word similarity and contextual evidence. *Transactions of the Association for Computational Linguistics*, 2:219–230, 2014.

[47] James Tizard, Hechen Wang, Lydia Yohannes, and Kelly Blincoe. Can a conversation paint a picture? mining requirements in software forums. In *2019 IEEE 27th International Requirements Engineering Conference (RE)*, pages 17–27. IEEE, 2019.

[48] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.

[49] Lorenzo Villarroel, Gabriele Bavota, Barbara Russo, Rocco Oliveto, and Massimiliano Di Penta. Release planning of mobile apps based on user reviews. In *2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE)*, pages 14–24. IEEE, 2016.

[50] Grant Williams and Anas Mahmoud. Mining twitter feeds for software user requirements. In *2017 IEEE 25th International Requirements Engineering Conference (RE)*, pages 1–10. IEEE, 2017.

[51] Xiaohui Yan, Jiafeng Guo, Yanyan Lan, and Xueqi Cheng. A biterm topic model for short texts. In *Proceedings of the 22nd international conference on World Wide Web*, pages 1445–1456, 2013.

[52] Jianhua Yin and Jianyong Wang. A dirichlet multinomial mixture model-based approach for short text clustering. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 233–242, 2014.