The Ineffectiveness of Domain-Specific Word Embedding Models for GUI Test Reuse

Farideh Khalili
Sharif University of Technology
Tehran, Iran
khallili@ce.sharif.edu

Ali Mohebbi
USI Università della Svizzera italiana
Lugano, Switzerland
ali.mohebbi@usi.ch

Valerio Terragni
University of Auckland
Auckland, New Zealand
v.terragni@auckland.ac.nz

Mauro Pezzè
USI Università della Svizzera italiana
Lugano, Switzerland
Schaffhausen Institute of Technology
Schaffhausen, Switzerland
mauro.pezze@usi.ch

Leonardo Mariani
University of Milano - Bicocca
Milan, Italy
leonardo.mariani@unimib.it

Abbas Heydarnoori
Sharif University of Technology
Tehran, Iran
heydarnoori@sharif.edu

ABSTRACT

Reusing test cases across similar applications can significantly reduce testing effort. Some recent test reuse approaches successfully exploit word embedding models to semantically match GUI events across Android apps. It is a common understanding that word embedding models trained on domain-specific corpora perform better on specialized tasks. Our recent study confirms this understanding in the context of Android test reuse. It shows that word embedding models trained with a corpus of the English descriptions of apps in the Google Play Store lead to a better semantic matching of Android GUI events. Motivated by this result, we hypothesize that we can further increase the effectiveness of semantic matching by partitioning the corpus of app descriptions into domain-specific corpora. Our experiments do not confirm our hypothesis. This paper sheds light on this unexpected negative result that contradicts the common understanding.

CCS CONCEPTS

• **Software and its engineering → Software testing and debugging**; • **Human-centered computing → Mobile phones**; • **Computing methodologies → Natural language processing**.

KEYWORDS

GUI test reuse, mobile testing, Android, word embedding, NLP

ACM Reference Format:
Farideh Khalili, Ali Mohebbi, Valerio Terragni, Mauro Pezzè, Leonardo Mariani, and Abbas Heydarnoori. 2022. The Ineffectiveness of Domain-Specific Word Embedding Models for GUI Test Reuse . In *30th International Conference on Program Comprehension (ICPC '22), May 16–17, 2022, Virtual Event, USA.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3524610.3527873

1 INTRODUCTION

Reusing GUI test cases across similar Android applications is a recent and promising research direction [2, 13, 20, 32]. Test reuse approaches automatically migrate GUI tests from a source to a target app that shares similar functionalities, by combining *semantic matching of GUI events* with *test generation*. Semantic matching identifies similar events across the source and target apps by applying word embedding techniques [22] to the textual information of the GUI widgets associated with the events. Test generation exploits the similarities identified with semantic matching to migrate GUI tests from the source to the target app.

At ISSTA 2021 we presented the first study on the semantic matching of GUI events for GUI test reuse techniques [19]. The study identifies four main components of semantic matching, and comparatively evaluates the impact of different choices for each component on the effectiveness of semantic matching. Our study discloses some useful findings that both help engineers identify better matching algorithms and offer important insights on GUI test reuse for Android applications [19].

One of such insights is that training word embedding models with corpora of documents specific to the mobile app domain generally leads to better results. In particular, we built GOOGLE-PLAY, a new corpus that consists of the English descriptions of 900,805 Android apps in the Google Play Store. The results that we present in our ISSTA 2021 paper show that the semantic matching configurations that use GOOGLE-PLAY to train word embedding models outperform semantic matching configurations that use general corpora. This result is in line with the common understanding in the NLP community that word embedding techniques trained on domain specific corpora perform better on related specialized tasks [12]. Indeed, a word can have different meanings depending on the context of usage (polysemy). Models trained on GOOGLE-PLAY should reflect the same word usage that mobile apps commonly adopt.

Motivated by this result, we investigated how to partition the GOOGLE-PLAY corpus into finer-grained corpora, focusing on different application domains. Indeed, mobile applications refer to many unrelated domains that use the same words differently. For example, applications of categories "Fitness & Health" and "Food & Drink" use the word "bar" differently. Our hypothesis is that we can further improve the semantic matching of Android test reuse by exploiting word embedding models trained on specialized corpora that contain only semantically related app descriptions. Test reuse approaches will automatically select the word embedding model that corresponds to the most semantically related partition based on the Google Play descriptions of the source app.

We investigated the validity of our hypothesis by experimenting with various configurations and algorithms of state-of-the-art topic modeling approaches. Our results are negative. *They indicate that specialized word embedding models do not improve the effectiveness of semantic matching*. In this paper, we present our methodology for creating specialized word embedding models and discuss the negative results as well as our insights that might shed light on this unexpected outcome. We released a replication package to support future work at https://doi.org/10.5281/zenodo.6092139

2 PARTITIONING OF APP DESCRIPTIONS INTO DOMAIN-SPECIFIC CORPORA

We partitioned the app descriptions of the GOOGLE-PLAY corpus [19] into semantically coherent clusters by means of *topic modeling* [3, 8, 30], commonly used to classify apps into meaningful categories [1, 28, 33, 34]. A topic model is a statistical model for discovering the abstract "topics" that occur in a collection of documents. In the GOOGLE-PLAY corpus, a document is the English description of an app in the Google Play Store. There are three key design choices to customize a topic modeling approach to a specific problem: (i) the topic modeling algorithm, (ii) the target number of topics, and (iii) pre-processing of the corpus. We investigated different combinations of these design choices to select the best approach.

Topic Modeling Algorithms We experimented with three of the most commonly used topic modeling algorithms:

Latent semantic analysis (LSA) [8] is a mathematical method, based on a distributional hypothesis that takes into account how frequently words appear in a document and in the whole corpus.

Latent Dirichlet Allocation (LDA) [3] is a probabilistic method that assumes that the distributions of both topics in a document and words in topics are Dirichlet distributions.

Hierarchical Dirichlet process (HDP) [30] is an extension of LDA. HDP uses statistical inference to learn the number of topics based on the corpus.

Target Number of Topics LSA and LDA take the number of clusters as an input. We experimented with a number of topics that ranges from 2 to 102.

Pre-processing We performed the canonical pre-processing steps on the GOOGLE-PLAY corpus, that we used in our ISSTA 2021 study. We converted all letters to lower case, removed punctuations, removed non-alphabetic letters and English stop words and performed lemmatization [18].

We also considered additional pre-processing steps that are often crucial to obtain a meaningful topic modeling [18]: Vocabulary and document pruning. Vocabulary pruning removes words that have either a very low or a very high frequency in the corpus since such words produce noise and result in low-quality topic models. Document pruning removes documents that are either too short or too long, which might lead to a low-quality topic model [9]. Short documents might not contain enough information to characterize a topic [14], whereas long documents usually cover multiple topics.

To identify the strategy that works best for the corpus at hand, we experimented with different pre-processing strategies for vocabulary and document pruning. Indeed, the best strategy can only be determined empirically, because the effectiveness of such strategies depends on the intrinsic characteristics of the corpus [7, 17]. We experimented with the following popular strategies:

Vocabulary pruning strategies:

S1 It defines lower and upper bounds based on the word frequency. It prunes words that either occur in more than X_{up}% documents or in less than X_{low}% documents (default values $X_{up} = 15\%$ and $X_{low} = 0.5\%$ [9, 16]).

S2 It defines lower and upper bounds by assuming a Gaussian distribution of words frequency. It prunes words with a frequency that belongs to the tails of the Gaussian plot: it prunes the first and last X% of the distribution (default value $X = 5\%$).

Document pruning strategies:

S3 It prunes documents that have more than X_{up} and less than X_{low} words (default values $X_{up} = 1000$ and $X_{low} = 50$ [9]).

S4 It sorts the documents based on their size and prunes the top and bottom X% (default value $X = 5\%$).

S5 It defines lower and upper bounds by assuming that documents size has a Gaussian distribution. Similar to vocabulary pruning, it prunes the first and last X% of the distribution (default value $X = 5\%$).

Automated Identification of the Best Model A topic model computes the *word probability distribution* over topics, which is represented as words sorted in descending order with respect to their contribution to the topic. A good topic model is interpretable, that is, the words with the highest probability in the probability distribution are semantically coherent [5]. We automatically evaluated the topic coherence, by computing the topic *coherence value* (cv) metric [25], which uses co-occurrence of words to quantify the semantic coherence of topics [23]. We relied on *cv* as it is recognized to be the best quantitative metric that captures the coherence of topic models [25]. In fact, *cv* outperforms existing metrics with respect to the correlation to human judgments [25]. This metric captures the coherence of a model as a value between 0 and 1, representing highly coherent models with high values.

We experiment on a random sample of 50,000 documents in GOOGLE-PLAY (~5.5%), to be more efficient when identifying the best configuration for topic modeling [16]. To reduce the number of configurations to evaluate, we incrementally considered the number of topics with step 10 from 2 to 102, for each combination of pre-processing strategy and algorithm (LDA and LSA). After identifying the best range for each configuration, we tried all numbers close to that range with a radius of 10 numbers, to see if we can find a number of topics that leads to a better result (higher cv). For each pruning strategy, we explored different parameters values in addition to default ones and selected the value that yields the better performance.

The configuration with the best performance among the ones that we explored ($cv = 0.64$) uses the LDA topic modeling algorithm with a target of 27 topics, and applies strategies S1 for vocabulary pruning (using the default values), followed by S5 for document pruning (using the tuned value of $X = 15\%$).

The results of these experiments gave us three important insights: (i) The order in which the vocabulary and document pruning are performed affects the results, and we generally get better results if we apply vocabulary pruning first; (ii) The LDA algorithm performs better than HDP and LSA; and (iii) There are some domain-specific common words (such as Application and Google) that appear as the most contributing words for some of the topics of the best models. Such common words reduces the quality of the models.

Best Model Validation We recomputed the coherence values of the three models that achieved the highest values on the random sample by referring to the whole GOOGLE-PLAY corpus. We confirmed that the selected topic model configuration achieves the best performance indeed. We observed an even higher coherence value $cv = 0.71$ ($cv = 0.64$ on the 5.5% random sample of GOOGLE-PLAY). We enriched the vocabulary pruning by adding a manually created list of domain-specific common words based on the topics in the sampled dataset, to prune some domain-specific common words (insight (iii)). We build a new topic model with the same configuration, and obtained a model of improved quality, $cv = 0.73$.

We confirmed the high quality of the selected model by manually inspecting the obtained model according to a standard protocol used in previous work [6, 10, 29, 31]. We checked the following conditions: (i) The 15 most probable words in the word probability distribution of each topic are semantically coherent [6, 29]; (ii) The most probable words in the word probability distribution do not contain common words [10, 31]. We observed that there are no more than three common words in the top 15 contributors.

3 EXPERIMENTS

The best topic model yields 27 clusters of the GOOGLE-PLAY apps. In the rest of the paper, we use **TOPICS** to refer to such a partition of GOOGLE-PLAY. The size of each cluster varies from 8,859 to 41,936 documents. We trained 27 domain-specific word embedding models, one for each cluster. We used our ISSTA 2021 framework to both evaluate the word embedding models in the context of semantic matching for Android test reuse and compare them with the ones obtained by GOOGLE-PLAY and other baseline corpora. We considered four word embedding techniques: (i) WORD2VEC [21], (ii) GLOVE [24], (iii) Word Mover's distance (WM) [11], (iv) FAST [4].

We considered the following four baseline corpora ordered from the most general to the most domain-specific:

BLOGS: a general domain dataset composed of 681,288 posts from 19,320 bloggers [27].

MANUALS: the user manuals of 500 Android apps [2].

GOOGLE-PLAY: the complete corpora proposed in our ISSTA 2021 paper [19].

CATEGORIES: a partition of GOOGLE-PLAY according to the categories of the Google Play Store.

Our ISSTA 2021 study considered the first three corpora. We added the CATEGORIES corpus to our experiments as it represents a baseline for a more specialized domain-specific corpus.

We evaluated the new configurations of the semantic matching by considering the 337 semantic matching queries. A query specifies an event e^s of the source test cases and the events $\langle E^t \rangle$ of the target app. A query q returns the list of target events sorted by their similarity scores computed with respect to the source event. We define E^t as the set of events that are actionable in all the GUI states traversed by the target test t^t . $E^t = \{e^t : \exists S \in \mathbb{S}, e^t \text{is actionable in } S\}$, where $\mathbb{S}$ is the sequence of state transitions obtained by executing t^t. In our study queries are derived from 139 available test migration scenarios (pairs of source and target test cases $\langle t^s, t^t \rangle$) provided by two of the state-of-the-art test migration approaches [2, 13]. The scenarios include a ground-truth annotation that specifies which events in the source test match which events in the target test.

While BLOGS, MANUALS, and GOOGLE-PLAY corpora lead to a single word embedding model (which can be used for any pair of source and target apps), CATEGORIES and TOPICS lead to multiple word embedding models. Given an arbitrary pair of source and target apps, we select the most appropriate model as follows. For CATEGORIES, we simply select the model associated with the category of the source app as specified in the Google Play Store. For TOPICS, we query our topic model from the Google Play description of the source app, to find the cluster semantically closest to the source app. We then retrieve the word embedding model trained on this cluster and use it for semantic matching the pairs of GUI events from the source and the target apps. Notably, since we are investigating test reuse among applications with similar functionalities, we only considered the source app, as we assume that the source and target applications belong to the same cluster.

Since each query has a ground truth target event (e^t_{gt}) as defined in our ISSTA paper, we can define the *rank* of a query as the position of e^t_{gt} in the ordered list returned by the query. Following our ISSTA 2021 experimental setup, we use two metrics to evaluate the results of queries: MRR and TOP1. *Mean Reciprocal Rank (MRR) [15]* is the average of the reciprocal ranks of the 337 queries Q. *TOP1* is the ratio of queries in which the ground truth (e^t_{gt}) is at the first position of the returned list of events. MRR represents how well a model performs on average, while TOP1 specifically rewards the capability of the model to identify the best matching element, regardless of average performance. These two metrics are representative of how semantic matching is used by test reuse techniques.

In this study, we investigate 240 configurations of the four components in our ISSTA 2021 semantic matching framework [19]: Four word embedding models trained on five corpus combined with four instances of event descriptors and three semantic matching algorithms (4×5×4×3= 240).

3.1 Negative Results

We group the 240 configurations of semantic matching according to the corpus of documents they use. TOPICS refer to all the 48 semantic matching configurations that use the 27 clusters of documents obtained with topic modeling to create the word embedding models. The first five rows of Tables 1 report *avg*, *mean*, *median*, and *max* of each group with respect to the metrics MRR and TOP1. The last two columns show the performance of each group by reporting the ranking based on the average MRR and TOP1.

The results show that the configurations that use TOPICS to train word embedding models perform worse than the ones that use the other corpora, for both TOP1 and MRR (rows from 1 to 5).

Table 1: Distribution of MRR and TOP1 values of the configurations of the semantic matching grouped by corpus of documents

	MRR				TOP1				rank of AVG	
corpus of documents	**AVG**	**Min**	**Median**	**Max**	**AVG**	**Min**	**Median**	**Max**	**MRR**	**TOP1**
BLOGS	0.6991	0.6084	0.7039	0.7740	0.5147	0.3857	0.5163	0.6468	4	4
MANUALS	0.6976	0.6052	0.7042	0.7796	0.5043	0.3768	0.5148	0.6439	5	7
GOOGLE-PLAY	0.7101	0.6165	0.7134	0.7958	0.5273	0.3976	0.5222	0.6706	2	2
CATEGORIES	0.6959	0.6015	0.6999	0.7791	0.5147	0.3798	0.5014	0.6587	6*	5***
TOPICS	0.6907	0.5865	0.7034	0.7834	0.5035	0.3620	0.5059	0.6587	9	8
h_categories_edit	0.6943	0.5934	0.6991	0.7791	0.5116	0.3768	0.4955	0.6587	7	6*
h_googleplay_edit	0.7088	0.6150	0.7133	0.7958	0.5247	0.3946	0.5207	0.6706	3***	3***
h_topics_edit	0.6908	0.5851	0.7017	0.7834	0.5031	0.3620	0.4940	0.6587	8	9
comb_topics_google-play	0.7393	0.6411	0.7438	0.8135	0.5610	0.4273	0.5608	0.6944	1***	1***

LEGEND:: The table reports the paired t-test [26] p-value computed for TOPICS and each of the other configurations
* == p-values < 0.05 ** == p-values < 0.01 *** == p-values < 0.001

3.2 Out-of-Vocabulary Issue

A possible explanation of the negative result could be the Out-of-Vocabulary issue (OOV) [4], which occurs when querying a word embedding model with words that are not in the corpus. Indeed, the partition of GOOGLE-PLAY leads to small clusters of documents, each of which might contain only a subset of all the unique words in GOOGLE-PLAY. If the OOV issue occurs while semantic matching two GUI events, such events will not match.

To investigate if the negative result is due to the OOV issue, we produced a hierarchy of models that avoids the OOV issue by design. If a query involves a word that does not belong to the current word embedding model, we propagate the query in the model hierarchy. If all models return OOV, we use edit-distance based similarity [19], which is not based on word embedding. We considered the following hierarchy: topics, category, googleplay, and edit distance. In total, we created three hierarchies of models. We refer to a hierarchical model by the first and last levels of the hierarchy. For example, *h_topics_edit* means that we first query word embedding models trained on TOPICS. If a query manifests an OOV issue, we propagate the query up to the model trained on the whole corpus GOOGLE-PLAY. If the query still manifests an OOV issue, we compute the similarity score with edit-distance.

The three bottom but one rows of Table 1 show the results. The TOP1 and MRR values of the hierarchical models with TOPICS as the first level are not significantly better than the ones of TOPICS only. This indicates that the OOV issue is not responsible for the poor results. Interestingly, *h_googleplay_edit* and *h_categories_edit* perform better than TOPICS. This suggests that the OOV issue may sometimes be beneficial by avoiding spurious matching of events.

3.3 Complementary Study

None of the 240 configurations of the semantic matching achieve a perfect semantic matching for all queries (the values of Columns "Max" in Table 1 are always < 1.0). Thus, it is important to understand if the configurations that use TOPICS and GOOGLE-PLAY perform poorly for different queries, that is, to see to what extent the word embedding models trained with TOPICS and GOOGLE-PLAY are complementary. In other words, although the configurations with TOPICS perform worse than those with GOOGLE-PLAY, it might be that TOPICS configurations achieve good results on queries for which GOOGLE-PLAY configurations perform poorly, and vice versa.

We studied the complementarity of TOPICS and GOOGLE-PLAY by creating an artificial semantic matching configuration: *comb_topics _google-play*. This configuration considers the best ranking of the ground truth e^t_{gt} of either TOPICS and GOOGLE-PLAY for each query. Note that such a configuration cannot be constructed in a real scenario because the ground truth e^t_{gt} would be unknown.

The results shown in the bottom row of Table 1 suggest that there exists a moderate level of complementarity between TOPICS and GOOGLE-PLAY. The average MRR and TOP1 of *comb_topics_google-play* are higher than the ones of GOOGLE-PLAY and TOPICS with statistical significance. The average MRR of *comb_topics_google-play* is 0.0292 and 0.0486 higher than the average MRR of GOOGLE-PLAY and TOPICS, respectively. The average TOP1 of *comb_topics_google-play* is 0.0337 and 0.0575 higher than the average TOP1 of GOOGLE-PLAY and TOPICS, respectively.

4 CONCLUSIONS AND FUTURE WORK

In this paper, we extended on our previous study [19] to investigate whether highly-specialized domain-specific corpora improve the effectiveness of the semantic matching of GUI events. We partitioned the GOOGLE-PLAY corpus into 27 semantically coherent clusters with topic modeling, trained the word embedding models with each cluster, and evaluated the effectiveness of the models for the semantic matching of GUI events. Unexpectedly, the results are negative.

While our results are negative, this study gives important insights. Since GOOGLE-PLAY outperforms general domain corpora [19], and GOOGLE-PLAY is domain-specific, we speculate that there exists an ideal level of specialization that lay between GOOGLE-PLAY and TOPICS. We also learned that creating highly-specialized word embeddings could still be useful in the context of test reuse. Indeed, we observed some complementarity between word embedding models trained with TOPICS and GOOGLE-PLAY. This could be because even highly specialized corpora benefit in parts from the information from other corpora. Understanding how to exploit such complementarity is important future work. In the future, we also plan to use the metrics MRR and TOP1 rather than semantic-matching agnostic metrics like *cv*, to select a proper clustering of GOOGLE-PLAY.

REFERENCES

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

[2] Farnaz Behrang and Alessandro Orso. 2019. Test migration between mobile apps with similar functionality. In *Proceedings of the International Conference on Automated Software Engineering (ASE'19)*. IEEE Computer Society, 54–65.

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

[4] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching Word Vectors with Subword Information. *arXiv* (2016).

[5] Jonathan Chang, Sean Gerrish, Chong Wang, Jordan Boyd-Graber, and David Blei. 2009. Reading tea leaves: How humans interpret topic models. *Advances in neural information processing systems* 22 (2009).

[6] Yong Chen, Hui Zhang, Rui Liu, Zhiwen Ye, and Jianying Lin. 2019. Experimental explorations on short text topic mining between LDA and NMF based Schemes. *Knowledge-Based Systems* 163 (2019), 1–13.

[7] Matthew J Denny and Arthur Spirling. 2018. Text preprocessing for unsupervised learning: Why it matters, when it misleads, and what to do about it. *Political Analysis* 26, 2 (2018), 168–189.

[8] Susan T Dumais, George W Furnas, Thomas K Landauer, Scott Deerwester, and Richard Harshman. 1988. Using latent semantic analysis to improve access to textual information. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. 281–285.

[9] George Forman. 2004. A pitfall and solution in multi-class feature selection for text classification. In *Proceedings of the twenty-first international conference on Machine learning*. 38.

[10] Yuening Hu, Jordan Boyd-Graber, Brianna Satinoff, and Alison Smith. 2014. Interactive topic modeling. *Machine learning* 95, 3 (2014), 423–469.

[11] Matt J. Kusner, Yu Sun, Nicholas I. Kolkin, and Kilian Q. Weinberger. 2015. From Word Embeddings to Document Distances. In *Proceedings of the International Conference on International Conference on Machine Learning (ICML '15)*. 957–966.

[12] Hongmin Li, Xukun Li, Doina Caragea, and Cornelia Caragea. 2018. Comparison of word embeddings and sentence encodings as generalized representations for crisis tweet classification tasks. *Proceedings of ISCRAM Asia Pacific* (2018).

[13] Jun-Wei Lin, Reyhaneh Jabbarvand, and Sam Malek. 2019. Test Transfer Across Mobile Apps Through Semantic Mapping. In *Proceedings of the International Conference on Automated Software Engineering (ASE'34)*. IEEE Computer Society, 42–53.

[14] Chi-Yu Liu, Zheng Liu, Tao Li, and Bin Xia. 2018. Topic Modeling for Noisy Short Texts with Multiple Relations.. In *SEKE*. 610–609.

[15] Tie-Yan Liu. 2009. Learning to Rank for Information Retrieval. *Found. Trends Inf. Retr.* 3, 3 (2009), 225–331.

[16] Daniel Maier, Andreas Niekler, Gregor Wiedemann, and Daniela Stoltenberg. 2020. How document sampling and vocabulary pruning affect the results of topic models. *Computational Communication Research* 2, 2 (2020), 139–152.

[17] Daniel Maier, Annie Waldherr, Peter Miltner, Gregor Wiedemann, Andreas Niekler, Alexa Keinert, Barbara Pfetsch, Gerhard Heyer, Ueli Reber, Thomas Häussler, et al. 2018. Applying LDA topic modeling in communication research: Toward a valid and reliable methodology. *Communication Methods and Measures* 12, 2-3 (2018), 93–118.

[18] Masoud Makrehchi and Mohamed S Kamel. 2017. Extracting domain-specific stopwords for text classifiers. *Intelligent Data Analysis* 21, 1 (2017), 39–62.

[19] Leonardo Mariani, Ali Mohebbi, Mauro Pezzè, and Valerio Terragni. 2021. Semantic Matching of GUI Events for Test Reuse: Are We There Yet?. In *Proceedings of the 30th International Symposium on Software Testing and Analysis (ISSTA 21)*. ACM.

[20] Leonardo Mariani, Mauro Pezzè, Valerio Terragni, and Daniele Zuddas. 2021. An Evolutionary Approach to Adapt Tests Across Mobile Apps. In *International Conference on Automation of Software Test (AST '21)*. 70–79.

[21] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv* (2013).

[22] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and Their Compositionality. In *Proceedings of the International Conference on Neural Information Processing Systems (NIPS '13)*. 3111–3119.

[23] David Newman, Jey Han Lau, Karl Grieser, and Timothy Baldwin. 2010. Automatic evaluation of topic coherence. In *Human language technologies: The 2010 annual conference of the North American chapter of the association for computational linguistics*. 100–108.

[24] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing (EMNLP)*. 1532–1543.

[25] Michael Röder, Andreas Both, and Alexander Hinneburg. 2015. Exploring the space of topic coherence measures. In *Proceedings of the eighth ACM international conference on Web search and data mining*. 399–408.

[26] EB Roessler, RM Pangborn, JL Sidel, and H Stone. 1978. Expanded statistical tables for estimating significance in paired—preference, paired–difference, duo–trio and triangle tests. *Journal of food Science* 43, 3 (1978), 940–943.

[27] Jonathan Schler, Moshe Koppel, Shlomo Argamon, and James W Pennebaker. 2006. Effects of age and gender on blogging.. In *AAAI spring symposium: Computational approaches to analyzing weblogs*, Vol. 6. 199–205.

[28] Didi Surian, Suranga Seneviratne, Aruna Seneviratne, and Sanjay Chawla. 2017. App Miscategorization Detection: A Case Study on Google Play. *IEEE Transactions on Knowledge and Data Engineering* 29, 8 (2017), 1591–1604.

[29] Shaheen Syed and Marco Spruit. 2017. Full-text or abstract? Examining topic coherence scores using latent dirichlet allocation. In *2017 IEEE International conference on data science and advanced analytics (DSAA)*. IEEE, 165–174.

[30] Yee Whye Teh, Michael I Jordan, Matthew J Beal, and David M Blei. 2006. Hierarchical dirichlet processes. *Journal of the american statistical association* 101, 476 (2006), 1566–1581.

[31] Xukun Wang, Matthias Lee, Angie Pinchbeck, and Fatemeh Fard. 2019. Where does LDA sit for GitHub?. In *2019 34th IEEE/ACM International Conference on Automated Software Engineering Workshop (ASEW)*. IEEE, 94–97.

[32] Yixue Zhao, Justin Chen, Adriana Sejfia, Marcelo Schmitt Laser, Jie Zhang, Federica Sarro, Mark Harman, and Nenad Medvidovic. 2020. FrUITeR: a framework for evaluating UI test reuse. In *Proceedings of the Joint Meeting on Foundations of Software Engineering (ESEC/FSE 20)*. 1190–1201.

[33] Hengshu Zhu, Huanhuan Cao, Enhong Chen, Hui Xiong, and Jilei Tian. 2012. Exploiting enriched contextual information for mobile app classification. In *Proceedings of the 21st ACM international conference on Information and knowledge management*. 1617–1621.

[34] Hengshu Zhu, Enhong Chen, Hui Xiong, Huanhuan Cao, and Jilei Tian. 2013. Mobile app classification with enriched contextual information. *IEEE Transactions on mobile computing* 13, 7 (2013), 1550–1563.