Similarity Measures and Requirements for Recommending User Stories in Large Enterprise Development Processes

Matthias Jurisch, Stephan Böhm, Maria Lusky

Faculty of Design – Computer Science – Media
RheinMain University of Applied Sciences
Wiesbaden, Germany
Email: {stephan.boehm, matthias.jurisch, maria.lusky}@hs-rm.de

Katharina Kahlcke

User Experience Consulting
DB Systel GmbH
Frankfurt, Germany
Email: katharina.kahlcke@deutschebahn.com

***Abstract*—In mobile application development projects, large enterprises have to face special challenges. To meet these challenges and to meet today's high expectations on user centered design, inter-project knowledge transfer of software artifacts becomes an important aspect for large software development companies. For supporting this kind of knowledge transfer, we propose two approaches based on textual similarity of user stories for a recommendation system: the first approach uses standard information retrieval techniques whereas the second approach uses a more recent approach from language modeling, namely word embeddings. We also present a three-step evaluation of these approaches, comprising of a data analysis, a survey and a user study. The results tend to support the information retrieval approach and not only show that user story similarity rated by users and rated by such an algorithm is connected, but also demonstrate a strong relation between user story similarity and their usefulness for inter-project knowledge transfer. Besides, our evaluation shows that using word embeddings showed worse results than the established information retrieval approach in the domain of large enterprise application development.**

***Keywords*–Mobile Enterprise Applications; User Stories; Recommendation Systems; User Centered Design.**

I. Introduction

In recent years, the user centered design approach has become an integral part of software development, and also for mobile application (app) development. Often, at the beginning of an agile development process, requirements for an app are analyzed and are written down in the form of user stories. They are short requirement descriptions from the user's point of view. Based on these user stories, during the further development process other software artifacts, such as documentation, screen designs, or source code are created to support the development process. Especially in large enterprises, the reuse of these software artifacts can save time and resources, since large software development companies are facing several challenges: There are multiple development projects at the same time, resulting in a large number of software artifacts. Due to a lack of time, these artifacts are often not properly documented in order to support a reuse of these materials and if there is a documentation, the form and content are not standardized. Furthermore, team members often do not know if there is a project with similar requirements and which coworker they can contact about a reuse of software artifacts. In general, large software development companies deal with lack of transparency in development projects, contact persons, and software artifacts.

Saving time and resources through reuse is especially desirable for organizations in the Mobile Enterprise Application (MEA) market: due to digitalization trends, these enterprises are developing many apps for various customers at the same time. Nevertheless, quick time to market is important because of a rapidly changing mobile ecosystem. Additionally, enterprises can only access a limited number of specialists that should focus on demanding tasks and work on difficult problems that have not been solved in the company already. Given this background, in this paper we propose an approach that supports the reuse of software artifacts in mobile app development projects based on textual similarity of user stories. This paper is an extension of [1]. In a previous paper on this problem, we showed that similar user stories can be identified via classical methods of information retrieval [2]. In the following evaluation, we investigate how well these methods work in a real world dataset and compare it to a mor recent approach from the language modeling area, namely word embeddings. We also evaluate how useful our approach is for employees of a large software development company. Therefore, Section II provides an overview on related work. Section III introduces our approach. An evaluation is described in Section IV and its results are presented in Section V. These results are discussed afterwards in Section VI. Section VII concludes this paper and gives an outlook on further research.

II. Foundations and Related Work

Supporting reuse in the context of software development can be facilitated in many ways. One of these ways is best practice sharing, where good solutions for common problems are exchanged within a community of mobile app developers and designers. However, this requires a lot of work and time to find common problems and respective solutions. Especially in the context of MEA development, time is an important factor. Therefore, supporting this process with automated approaches seems to be promising. An automated approach in this area is the use of recommendation systems [4]. Recommendation systems recommend items to users based on item similarities or preferences of similar users. This idea can be applied to software engineering, where a system can recommend software engineering artifacts to developers [5]. The goal of these recommendation systems is mainly to support the software development process, especially focused on the implementation phase and fixing bugs.

An important field in this area is issue triage. This field deals with supporting the management of bug reporting systems. This includes both recommending specific developers for a given bug report, as well as detecting duplicate bugs. Usually, standard information retrieval techniques or shallow machine learning approaches are used [6]: An approach by Runeson et al. [7] is based on information retrieval techniques and tries to detect duplicates. Other approaches build on including other information besides text, e.g., execution information [8]. In [9], a framework for building recommendation systems for issue triage is presented. This is done by linking both developers and bug reports to software components. Besides using classical information retrieval methods, more recent approaches use deep learning-techniques like word embeddings [10]. While this area also includes computing similarities between textual descriptions in the area of software development, there are some important differences: (1) bug reports are often written from a very developer-centric perspective. (2) They usually contain a lot of technical information like log output. (3) The main goal of issue triage is not to support reuse, but to support bug management tasks.

Another approach to improve the knowledge management in software development projects is to document and store project information in an accessible way, e.g., in architectural knowledge management tools [11]. These approaches have also been applied in industrial case studies and were deemed fit for usage in an industry context: [12] evaluated a semantic architectural knowledge management tool that is based on existing data on software design patterns and their application in software projects. However, if this kind of data is not already available the overhead for documenting usage of design patterns can be too high for an application in practice. This is especially an issue for the fast-paced market of mobile enterprise applications.

In the last decade, user stories as a user-centric representation of requirements were introduced [13]. A typical user story is at most two sentences long and consists of a reference to a certain type of user, a description of an activity this user wants to do with the software and a reason why this will help the user. As an attachment to the user story, acceptance criteria add more detailed information to the user story. Only few approaches exist to support software reuse in the context of user stories: [14] proposes a recommendation system based on user stories and evaluates this system on a project history. However, it is not clear how helpful these recommendations would be when actually working on a new project. In our previous work [2], we evaluated how well information-retrieval-based approaches can distinguish between two types of user stories and which aspects of the user story are important to it.

The established method for text representation in information retrieval is the vector space model which dates back to the 1960s: each document is represented by a vector, where each vector component represents how often a term occurs in the document. This is often accompanied by weighting the terms given their prevalence in the overall corpus. The similarity between documents is computed by the cosine of the angles of their vector representation. However, this approach has a significant drawback: the semantic of terms can not be taken into account: terms like ”pretty” and ”lovely” are treated as completely unrelated terms, the same way as terms like ”driving” and ”universe”. To also represent the meaning of words in search corpora, latent semantic indexing was introduced (LSI) [15]. LSI is based on a singular value decomposition of the term by document matrix, which is the matrix built from all document vectors. While LSI can deal with the synonymy problem in some cases, it still conceptualizes language at the abstraction level of documents and can not determine meaning on a lower level – terms used in similar documents will be regarded as semantically similar by LSI, regardless of their immediate context.

To overecome this issue, more recent approaches use word embeddings, where an embedding should represent a term's context on the level of sentences. One of the most popular embedding approaches, Word2Vec [3], is able to represent semantic information based on an unsupervised learning procedure. This learning procedure is based on two models, Skip-

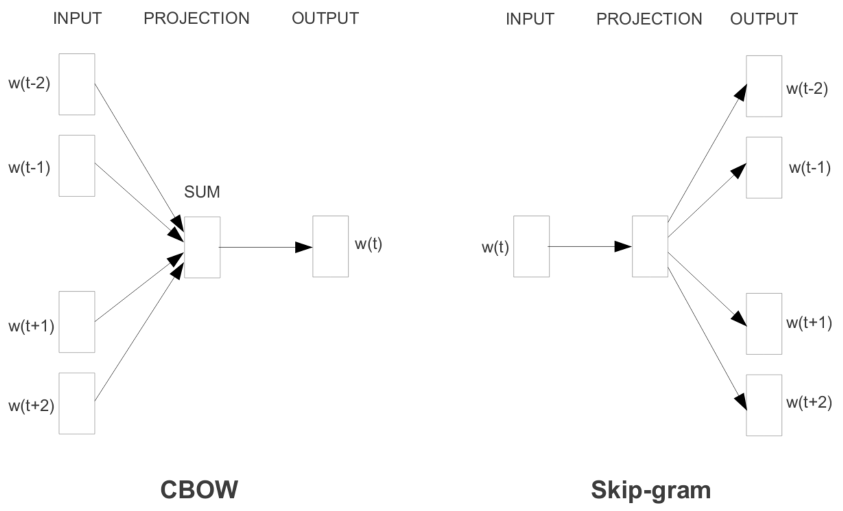

Figure 1. Continuous Bag of Words (CBOW) and Skip-Gram model [3]

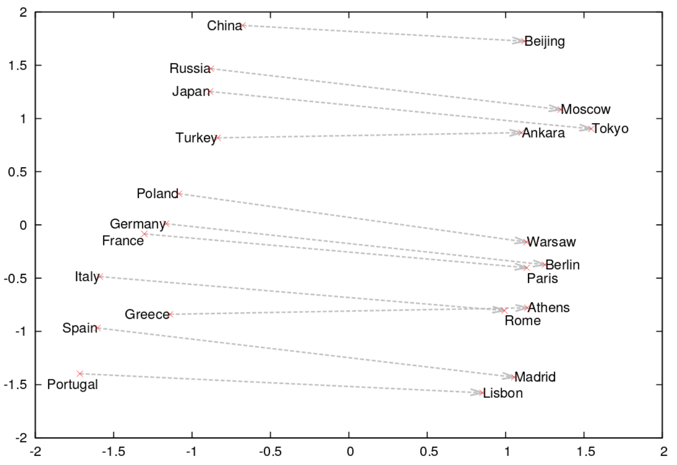

Figure 2. Principal component analysis of word embeddings for countries and capitals [16]

Gram and Continuous Bag of Words (CBOW). Both models are shallow neural network architectures depicted in Figure 1. Given a context of a few words (words $w(t-2) - w(t+2)$) CBOW predicts the word in the center ($w(t)$) using a weighted sum and a hidden embedding layer. E.g., given ”The dog ran very fast” as context, CBOW would try to predict the term ”ran”. The Skip-Gram model is built on the reverse task: given a centroid word (”ran” in the previous example), this model tries to predict the context. As a byproduct of learning these models, entity-specific weight vectors are produced, which capture some semantic relations. An example for these semantic relations is shown in Figure 2. This figure shows selected vectors for countries and capital cities. The semantic relation ”is capital of” can be observed through a shift operation in the vector space. This property holds for other relations, too. E.g., the vector from ”king” to ”queen” is very similar to the vector from ”man” to ”woman”. To our knowledge, text similarity based on word embeddings has not been used for recommending similar user stories.

In this paper, we expand our previous work to an evaluation of how useful recommendations on a real-world software engineering dataset are and what information needs to be contained in these recommendations to make them actually helpful. We also evaluate, how established methods for information retrieval compare to modern approaches like word embeddings in relation to these aspects. This issue has not been addressed by the approaches we mentioned in this section and is required to make recommendations in the context of user stories usable in practice. The only way to evaluate the usefullness of recommendations is to conduct a survey with practitioners from the industry.

III. Recommendation Approaches

Our general approach to supporting reuse is to use similarity measures for documents to recommend textually similar user stories to the story a participant in an app development-project is currently working on. As similarity measures we use established techniques from information retrieval as well as a newer methods from the area of language modeling, namely word embeddings. The information that is attached to the recommended user stories (e.g., screen designs, textual documents or source code) can then be used to support current efforts. In this way, team members could reuse results from different projects without previously knowing about these projects.

A. *Information Retrieval Methods*

To find textually similar user stories based on established information retrieval methods, a search based on the well-known information retrieval approach Term Frequency-Inverse Document Frequency (TF-IDF) and stop word removal is used. Stop words are words that occur frequently in texts so that they do not contain useful information. Examples for stop words are ”I”, ”the”, ”a”, etc. These words are removed from the user stories before processing the user stories with TF-IDF, which represents texts as follows: Each document d (i.e., a user story) is represented by a vector $\boldsymbol{W}_d$, which contains an entry for each term used in the dataset. Each vector component $\boldsymbol{W}_{d,t}$ represents the importance of a term t for the document d. This representation is computed by the frequency of a given term in the document $tf_{d,t}$ multiplied by the inverse document frequency $\log \frac{N}{df_t}$, where N is the number of all documents and df_t is the number of occurrences for a given term in all documents. This yields the following formula for a document's vector representation:

$$\boldsymbol{W}_{d,t} = tf_{d,t} * \log \frac{N}{df_t}$$

To compute the similarity between documents, the cosine of the angle of two vectors is used. The naive approach for similarities would be to compute the euclidian distance between vectors, however, this would favour documents with similar lengths.

Cosine similarities are then used to order texts regarding their relative similarities. Thus, we do not use similarity scores as an absolute value, but only to distinguish between more and less similar documents. To find similar user stories to one given user story, the similarity is computed according to the described procedure. User stories are then ordered by their similarity and the user story with the highest score is considered the most similar.

B. *Word embeddings*

To find similar user stories based on word embeddings, first, a word embedding is needed. Word embeddings are usually trained on very large corpusses such as the Google News dataset, which contains around six Billion tokens [3]. Even in large companies, creating a text corpus of that size to learn embeddings is not realistic. Because this issue is not unique to the domain of large companies, pretrained sets of general purpose word embeddings are publically available on the web [17].

However, these word embeddings need to be transformed into a document embedding. This is usually done by averaging the word embeddings for a document:

$$\boldsymbol{e}_d = \frac{1}{|d|} \sum_{t \in d} \boldsymbol{e}_w$$

where $\boldsymbol{e}_d$ and $\boldsymbol{e}_w$ are embeddings of word w and document d and $|d|$ is the length of d. To compute the similarity between documents, we use the same distance measure as for comparing TF-IDF documents, namely the cosine distance. As with TF-IDF, we can then order documents by their similarity and thus find the most similar user stories to each story.

For implementing this process, we use spaCy [18], a freely available language processing toolkit. The embeddings we use are pre-trained on two corpuses: the TIGER and WikiNER corpuses. The TIGER corpus [19] contains text from newspaper articles from the "Frankfurter Rundschau", a german newspaper. It consist of 50000 sentences containing 900000 tokens. WikiNER [20] is build upon Wikipedia articles from several languages, the german part used in the spacy model consists of 19.8 Million Sentences containing 208 Million Tokens.

IV. Evaluation

The aim of our evaluation is to assess the recommendation approaches described in Section III regarding their suitability in a real-world scenario. To do so, it is relevant to deepen the understanding of the reuse process in order to get a complete picture on the potential usefulness of a recommendation system in large enterprises. Therefore, our evaluation answers the following research questions:

1) Which kind of knowledge transfer is already being practiced?
2) Can an automated recommendation system be useful for supporting knowledge transfer?
3) Is there a relation between user story similarity and their usefulness?
4) How do information retrieval approaches compare to more recent language modeling approaches in this environment?

A. Methodology

To answer these research questions, we conducted an evaluation comprising three steps: In the first step, we analyzed a dataset of user stories from a large German software development company. In the second step, we distributed a questionnaire covering questions about practices in inter-project knowledge transfer in general. In the third step, we invited participants to single sessions where they were asked to solve tasks focusing on user stories in inter-project knowledge transfer.

The number of participants in this study was limited to a small number due to the testing requirements: all participants had to come from the same company with a specific expertise on the implementation of the user stories and as references for the similarity comparison. Thus, the results are far from representative and cannot be considered as an empirical validation. However, the results of this prestudy can five some critical and usable expert feedback on the potential usefulness and applicability of our approach.

B. Dataset

To evaluate the usefulness of recommendations based on user stories in the area of Mobile Enterprise Application Development, we used a dataset of real-world user stories out of nine app development projects provided by an industry partner. The dataset contains 591 user stories, of which 355 are long enough to contain meaningful information. User stories were considered long enough when they were at least 80 characters long, which is roughly two times the length of only the formal aspects of a user story description. This boundary was set by investigating example user stories. A histogram of story length (in characters) is shown in Figure 3. From the distribution of story length and the standard deviation, we can already conclude that the dataset is very heterogeneous, as could be expected in a real-world dataset. The data is not only heterogeneous regarding the textual length, but also regarding their specifity and their degree of abstraction. For example, some user stories describe fixing typos in data protection regulation informations, while others describe a high level view of a location-based service.

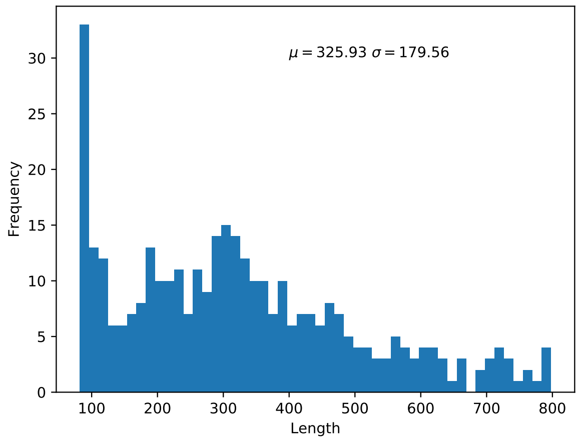

Figure 3. Distribution of User Story Length

For each user story in the dataset, we computed the top five most similar user stories according to TF-IDF, with cosine similarity both for stories from different projects as well as stories from the same project. Stories from the same project shold overall be more similar than stories from different projects, since user stories in one project can deal with overlapping topics. E.g., there can be one story describing a search function, one describing how the search can be accessed and another story describing how search results should be sorted and displayed. A histogram of cosine similarity values between all user stories is shown in Figure 4. Stories in the same project are given higher similarity values than stories from different projects, which indicates that it is possible to differentiate between projects using cosine similarities of TF-IDF vectors.

We followed the same procedure for vectors generated by the word embeddings procedure. Figure 5 shows a histogram of cosine similarities between stories in the same project as well as in different projects. When comparing Figure 4 and Figure 5, one can observe that while for TF-IDF, the average is close to zero, for embedding-based methods it is close to 1. This result can be expected from the nature of the vector spaces: Whereas a TF-IDF vector space has many dimensions and the vectors are relatively sparse, a word embedding space contains only a few hundred dimensions with dense vectors. Another notable difference is that embedding-based methods show a smaller overall variance. In general, the distribu-

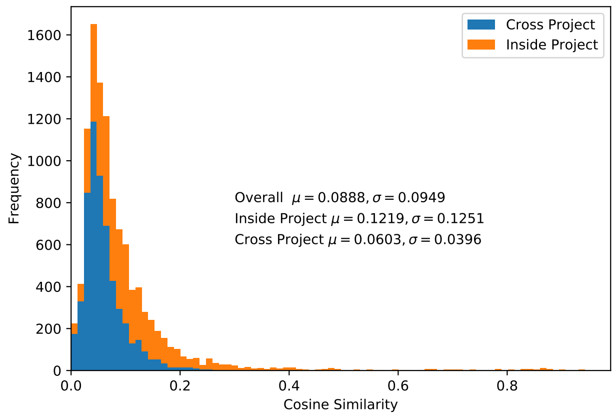

Figure 4. Distribution of TF-IDF User Story Similarities

tion of cross- and inside-project similarity values are more similar for embedding-based methods. This implies that the TF-IDF-based approach might be better suited to distinguish between projects. However, there is a difference between the distributions for cross- and inside project similarities for the embedding approach and the usefulness needs to be further evaluated.

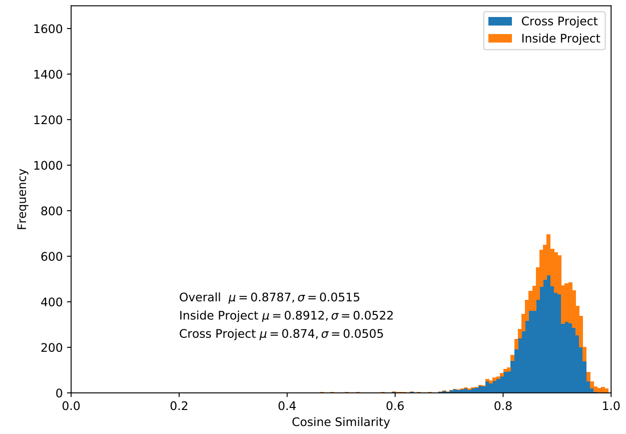

Figure 5. Distribution of Embedding-Based User Story Similarities

To test the effect of combined similarity measures, we repeated these experiments by simply adding the two similarity measures. Results of this approach are shown in Figure 6. Note that this combination of similarity measures has a domain of [0,2]. However, just adding the two similarity measures leads to more influence in the combined similarity measure for the similarity with a higher standard deviation and mean. Hence, with this unscaled version of the combined similarity measure, the similarity is influenced mainly by the embedding-based similarity.

To overcome this issue, we constructed a new similarity measure that scales both similarities to the same mean of 0 and to the same standard deviation. In this way, both similarity

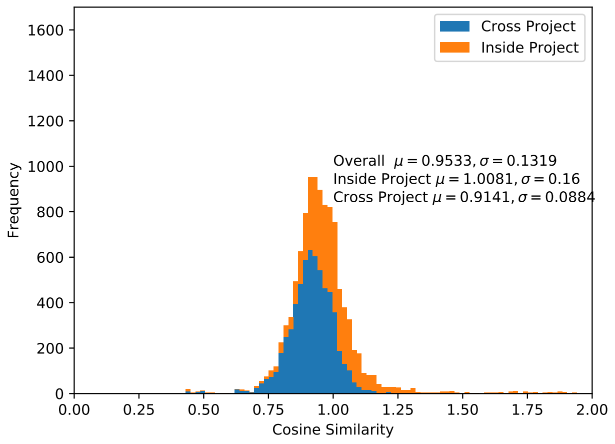

Figure 6. Distribution of Combined Similarities, Unscaled

measures have an equal influence on the value of the combined similarity measure. The similarity distribution of this combined method is shown in Figure 7. This combined similarity measure is the variant with the least difference between means of Cross- and Inside-Project similarities. Hence, the combined method is the weakest method for distinguishing between projects. The highes differences can be found between TF-IDF scores.

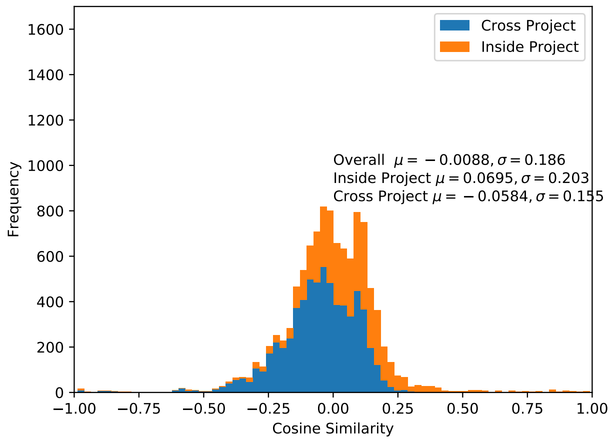

Figure 7. Distribution of Combined Similarities, Scaled

C. *Survey*

To get an overall overview of the requirements in user story-centered reuse, we designed a questionnaire that comprised ten questions about inter-project knowledge transfer. The questionnaire was online for 17 days and was distributed among the employees of a large German software development company. It was also used for recruitment of participants for the user study. First, the participants had to specify their field of activity. We then explained to them our approach for inter-project knowledge transfer that underlay the questions, which is the re-use of software artifacts and knowledge, such as user stories, screen designs, documentation or source code, during development projects of mobile applications.

We asked the participants, how useful such a knowledge transfer is and in which way and how regularly it is already being practiced in their department. Further, they had to name obstacles that occur with inter-project knowledge transfer. Then, we asked them to rate the usefulness of particular software artifacts in this context and they had to assess the viability of such a knowledge transfer in their department. They were asked to rate the usefulness of a software that would support knowledge transfer. Lastly, the participants were asked to rate the importance of additional information to specific software artifacts on a five-point scale. Importance may differ from usefulness in certain situations, since it is used to prioritize between different potentially useful artifacts.

D. *User Study*

The user study was carried out in one-on-one sessions with employees of a large German software development company. Each session lasted 20 to 30 minutes. We selected three user stories for which related user stories were known to be in the dataset. We computed the most similar user stories with both similarity approaches for each of these *reference* user stories with varying levels of similarity: one user story that the algorithm listed most similar, one that it listed as medium similar, and one that it listed as less similar, which lead us to three user story groups, one for each reference story.

Based on the reference user stories, the participants were asked to solve three tasks. First, they had to rank the user stories obtained by the algorithm regarding their similarity to the reference user story from the most similar to the least similar one. Then, they should rate the usefulness of each of the similar user stories. To determine the usefulness of the user stories, participants were told to estimate how much artifacts (e.g., source code, design documents or documentation) produced during an implementation of a ranked user story could contribute to the implementation of the corresponding reference user story. Note that this is not included in similarity aspects: user stories can cover a roughly similar topic, however, different levels of abstraction, different user types or platforms or technical aspects could make it impossible to actually reuse the results of the implementation of a user story in a different context. Such user stories would be considered similar by users, but finding these stories would not actually support reuse of artifacts related to one user story to another. Concluding the session, they were asked to name additional information that should be provided by the recommendation system in order to support the implementation of the reference user story.

V. RESULTS

The questionnaire was answered by nine employees of a large German Software development company. While nine participants are not enough to allow a detailed statistical analysis, this number is in general considered enough for usability testing [21]. Eight participants specified their field of expertise as conception, one as implementation.

All of the participants rated the knowledge transfer described by us (that is, the re-use of software artifacts and knowledge, such as user stories, screen designs, documentation or source code, during development projects of mobile applications), using a five-point scale from 1 – not useful at all to 5 – very useful, as very useful or rather useful (median=5; maximum=5; minimum=4). The currently used types of knowledge transfer selected from a list of pre-made options are shown in Figure 8. All of the participants stated that they already practiced this kind of knowledge transfer, seven via one-on-one conversations or group conversations, four via e-mail, chats or by using knowledge bases, and three via phone calls. On average, each person practices three methods of knowledge transfer. Only one stated to practice it on a regular basis, and eight practice it as needed. Obstacles for knowledge transfer selected by participants from a list of possible obstacles are shown in Figure 9. The most often named obstacle was missing information about a contact person (eight), followed by missing documentation of content and materials and lack of time (six respondents each). The participants described further obstacles as being unaware of the existence of reusable materials, as well as not knowing where to look for information regarding reusable artifacts.

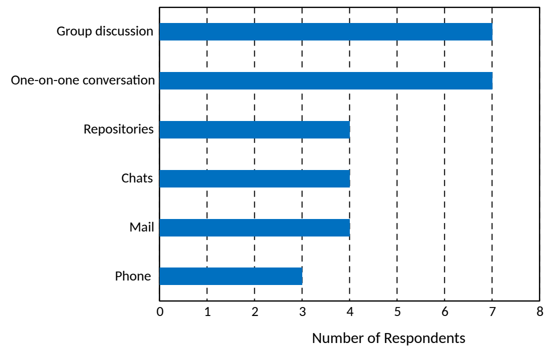

Figure 8. Currently Used Types of Knowledge Transfer

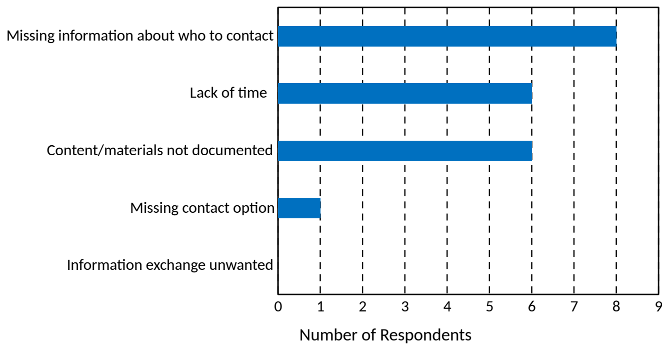

Figure 9. Existing Barriers for Knowledge Transfer

User ratings for usefulness of artifacts for Knowledge transfer on a five-point scale are shown in Figure 10. Screen designs were rated as most useful (median=5, maximum=5, minimum=3), followed by documentation of the software architecture (median=4, maximum=5, minimum=3). Ratings for potential usefulness, implementability and importance are shown in Figure 11. Regarding the viability of such a knowledge transfer in their department and in relation to specific software artifacts, the highest implementability was considered for screen designs, followed by documentation of the software

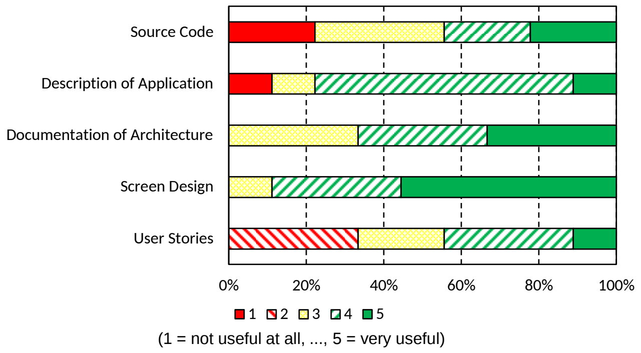

Figure 10. Perceived Usefulness of Artifacts

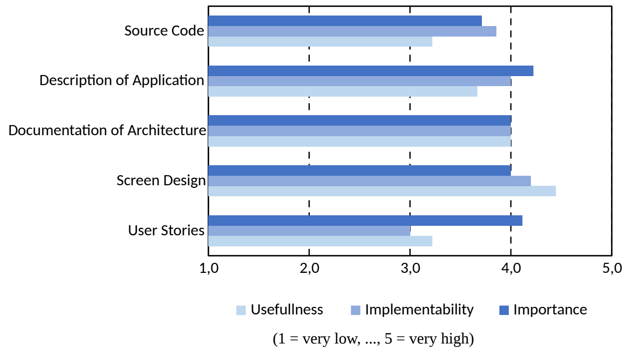

Figure 11. Responses on Potential Recommendation System Artifacts

architecture and use case descriptions. Furthermore, the answers revealed that most knowledge transfer that is already practiced concerns screen designs (done by 4 participants), documentation of the software architecture (2 participants) and user stories (1 participant).

The participants rated a software solution for supporting knowledge transfer on a five-point scale as rather useful (median=4; maximum=5; minimum=1), with 6 participants considering it rather useful or useful. Regarding the importance of additional information, information on use case descriptions were rated as most useful (median=4; maximum=5; minimum=3). In general, any kind of additional information (e.g., source code, screen designs, architecture documentation) was rated as ”rather important” for all kinds of software artifacts.

Of the eight participants of the user study, all were working in conception respectively design. Results of the user study are shown in Figures 12-15. Results of the first task show that the participants ranked the user stories similar to the ranking of the algorithm. Figure 12 shows the ranking of story similarity to the reference story by the TF-IDF algorithm and the mean ranking by the participants. The data shows that user story similarity of the algorithm seems to resemble the perceived user story similarity by humans: The three user stories ranked as most similar by the algorithm also got the highest similarity rankings by the participants. The user stories ranked as second by the algorithm were partially ranked as more and partially as less similar, but in general reflect the algorithmic ranking. For stories that are not obviously the least or most similar,

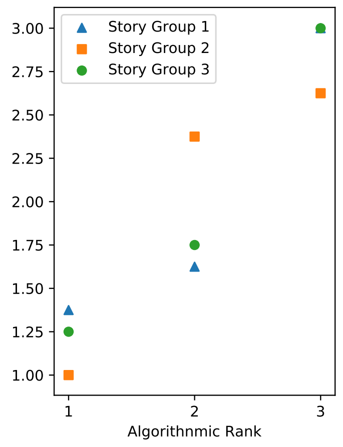

Figure 12. Ranking of user stories by TF-IDF and mean ranking by participants.

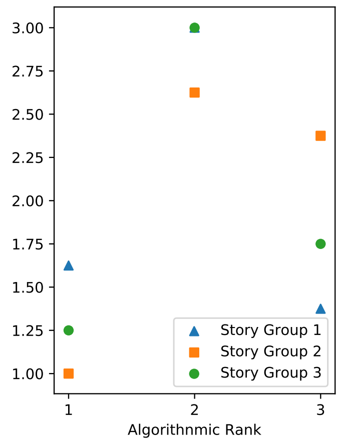

Figure 13. Ranking of user stories by Word-Embedding-Similarity and mean ranking by participants.

this result was to be expected. Accordingly, the three least similar user stories of the participants match those ranked by the algorithm.

The results of the same experiment with the embedding model are shown in Figure 13. While the stories rated as most similar by the algorithm were also perceived as rather similar by the users, some stories that where perceived as least similar by the algorithm where rated as very similar by the algorithm. This is especially the case in story group 1, where the story that was perceived as most similar by the users was ranked as least similar by the algorithm. The stories ranked as second most similar by the algorithm are ranked least similar by the users. Overall, there does not seem to be much agreement between user- and algorithmic ratings when using the embedding-based approach.

We also repeated this experiment with a combination of TF-IDF and Embeddings we already used in Section IV-B.

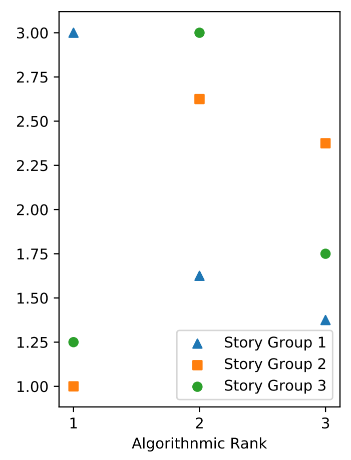

Figure 14. Ranking of user stories by Word-Embedding-Similarity combined with TF-IDF-Similarity and mean ranking by participants.

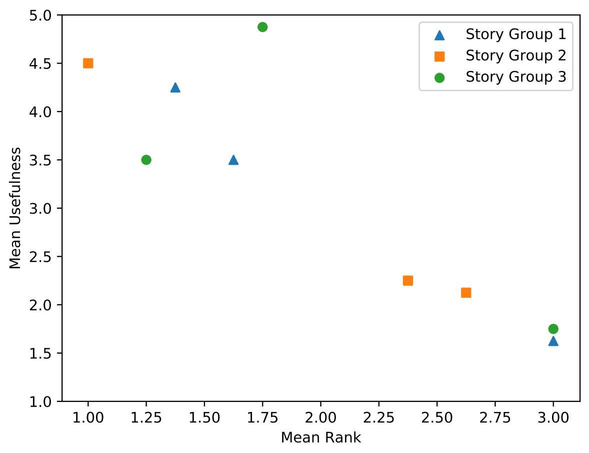

Figure 15. Mean estimated usefulness and mean ranking by participants per user story.

We only used the version with un-weighted addition of the combination, since this lead to slightly better results when distinguishing between projects. From the graph we can see no connection between algorithmic rank and average user scores. In contrast to the similarity measure using only word embedding similarities, where some connection between human and algorithmic rank was visible, no connection can be found.

Further on, for each user story in the three user story groups, we calculated the mean similarity ranking to the reference story given by the participants and the mean usefulness rating, according to the rated usefulness of a computed user story for the implementation of the reference story. As Figure 15 shows, there are two groups of user stories that are delineated from each other. The first group has higher usefulness values (3.50 to 4.88) and higher similarity rankings (1.00 to 1.75), while the second group has lower usefulness values (1.63 to 2.25) and lower similarity rankings (2.38 to 3.00). However, the user story with the highest usefulness (4,88) is ranked with medium similarity (1.75), while the two user stories with the lowest usefulness (1.63 and 1.75) are ranked as least similar.

VI. Discussion

The results of the questionnaire show that, in general, people appreciate knowledge transfer and that our approach for it meets the users' needs. This is also reflected in the fact that seven of our participants already practice this kind of knowledge transfer. Although, direct contact and personal conversations are the preferred ways of doing so. Electronic ways for contacting each other are rarely used. One reason for that might be that electronic methods, such as emails, chats or knowledge bases, do not meet the user needs for knowledge transfer. Nevertheless, for all these methods the user needs to know, which person can be contacted for further information – our approach takes this important feature into account, which was also the most often named obstacle for knowledge transfer. The second obstacle is insufficient documentation – this problem is also taken account of in our approach, since it simplifies documentation by searching existing software artifacts based on similarities. In general, the answers confirm that our approach addresses the right issues and helps to eliminate the obstacles for knowledge transfer that have been named by the participants. The software tool proposed by us was said to be most useful for screen design. This answer is not unusual, since almost all participants of our questionnaire work in design and conception. However, screen design is their main field of activity and thus, we consider it positive that our approach is rated as useful for this field. Further, the viability was rated highest for screen designs. These results give some evidence that our approach can create additional value in one of the most important fields for knowledge transfer. All in all, two thirds of the participants consider our approach useful or very useful.

The results of the user study show that the user story similarity of the TF-IDF algorithm seems to be connected to the perceived human user story similarity. For the embedding-based similarity algorithm this is not the case. The most likely cause of embeddings not working in this case is that our dataset contains a lot of words that are very specific to the domain of mobile application development. Therefore, they do not have a meaningful embedding or no embedding at all, since we used embeddings theat were pretrained on a general purpose dataset. However, these domain-specific words are often the words that carry a lot of meaning, since they express concepts that are highly relevant to a domain.

One mitigation for this issue would be retraining the word embeddings on a set of documents from the domain. However, given the small number of documents that use these words it is unclear if this can work. Even if documents of a relevant size were available, a significant amount of computing power would be required to train these embeddings.

Another mitigation would be a manual clustering of these domain specific important terms to capture their semantic relations. These clusters could then be used to improve search results for TF-IDF-based searches. However, finding these clusters and maintaining them requires a significant amount of manual work.

A combination of TF-IDF and Word-Embedding-based methods showed less connection between perceived similarity and algorithmic similarity. The most likely cause for this is that the disagreement between both similarity measures causes so much noise, so that the overall performance is impacted. This might be addressed by further tuning the wights of a combined approach, but given the poor performance of general-purpose embeddings for this use case, this does not seem like a very promising approach.

Furthermore, user story similarity mostly coincides with their usefulness. The data indicates that a low similarity implies a low usefulness. However, the most similar stories are not always the most useful ones, but to some extent a high similarity seems to be connected to higher usefulness. This confirms our assumption that similarity between user stories and usefulness for the implementation of another user story are not necessarily the same, since usefulness can be influenced by several dimensions of similarity: two user stories can share the same implementation technology, but not the same domain or vice versa. Two user stories can share the same domain and technology but one may use an outdated version of an API or adhere to a human interface guideline that has become obsolete. Hence, it is important to address these factors like this when building a recommendation system in the area of mobile enterprise application development.

VII. Conclusions and Further Research

In conclusion, the evaluation provided valuable findings, so that our research questions can be answered as follows:

1) Which kind of knowledge transfer is already being practiced?
   Mainly, knowledge transfer takes place in personal conversations between two people and groups. It is not carried out on a regular basis, but as required. Our results suggest that everyone does practice knowledge transfer in one way or the other.
2) Can an automated recommendation system be useful for supporting knowledge transfer?
   Our results show that an automated recommendation system is a useful tool for supporting knowledge transfer, especially for screen designs.
3) Is there a relation between user story similarity and their usefulness?
   The results of our user study indicate that similarity and usefulness are not necessarily the same, but there is a relation between user story similarity and their usefulness. Further, there also is a connection between user story similarity rated by an algorithm on the one hand and humans on the other hand.
4) How do information retrieval approaches compare to more recent language modeling approaches in this environment?
   On our dataset, established information retrieval approaches performed better than an embedding-based language modeling approach. The likely cause of this is the domain-specific vocabulary in this area.

As a next step, another iteration of the evaluation could be made in different companies, in order to receive results that are applicable in several contexts of work. Also, more approaches for computing similar user stories could be evaluated: A comparative study of textual similarity approaches such as word movers distance [22] or taking metadata into account, would provide valuable insights.

Acknowledgment

This work was funded by the German Federal Ministry of Education and Research, grant no. 03FH032PX5; the PROFRAME project at RheinMain University of Applied Sciences. All responsibility for the content of this paper lies with the authors.

References

[1] M. Lusky, M. Jurisch, S. Böhm, and K. Kahlcke, "Evaluating a User Story Based Recommendation System for Supporting Development Processes in Large Enterprises," in CENTRIC 2018, The Eleventh International Conference on Advances in Human-oriented and Personalized Mechanisms, Technologies, and Services, 2018, pp. 14–18.

[2] M. Jurisch, M. Lusky, B. Igler, and S. Böhm, "Evaluating a recommendation system for user stories in mobile enterprise application development," International Journal On Advances in Intelligent Systems, vol. 10, no. 1 and 2, 2017, pp. 40–47.

[3] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," CoRR, vol. abs/1301.3781, 2013. [Online]. Available: http://arxiv.org/abs/1301.3781

[4] M. Robillard, R. Walker, and T. Zimmermann, "Recommendation systems for software engineering," IEEE Software, vol. 27, no. 4, 2010, pp. 80–86.

[5] D. Cubranic, G. C. Murphy, J. Singer, and K. S. Booth, "Hipikat: A project memory for software development," IEEE Trans. Softw. Eng., vol. 31, no. 6, Jun. 2005, pp. 446–465. [Online]. Available: https://doi.org/10.1109/TSE.2005.71

[6] A. Goyal and N. Sardana, "Machine learning or information retrieval techniques for bug triaging: Which is better?" e-Informatica Software Engineering Journal, vol. 11, no. 1, 2017, pp. 117–141.

[7] P. Runeson, M. Alexandersson, and O. Nyholm, "Detection of duplicate defect reports using natural language processing," Proceedings - International Conference on Software Engineering, 2007, pp. 499–508.

[8] X. Wang, L. Zhang, T. Xie, J. Anvik, and J. Sun, "An Approach to Detecting Duplicate Bug Reports using Natural Language and Execution Information," Proceedings of the 30th international conference on Software engineering, 2008, pp. 461–470.

[9] J. Anvik and G. C. Murphy, "Reducing the Effort of Bug Report Triage: Recommenders for Development-Oriented Decisions," ACM Transactions on Software Engineering and Methodology (TOSEM), vol. 20, no. 3, 2011, pp. 10:1–10:35.

[10] S. Mani, A. Sankaran, and R. Aralikatte, "Deeptriage: Exploring the effectiveness of deep learning for bug triaging," arXiv preprint arXiv:1801.01275, 2018.

[11] R. Capilla, A. Jansen, A. Tang, P. Avgeriou, and M. A. Babar, "10 years of software architecture knowledge management: Practice and future," Journal of Systems and Software, vol. 116, 2016, pp. 191–205.

[12] M. Sabou et al., "Exploring enterprise knowledge graphs: A use case in software engineering," in European Semantic Web Conference. Springer, 2018, pp. 560–575.

[13] M. Cohn, User Stories Applied: For Agile Software Development. Redwood City, CA, USA: Addison Wesley Longman Publishing Co., Inc., 2004.

[14] H. Pirzadeh, A. D. S. Oliveira, and S. Shanian, "ReUse : A Recommendation System for Implementing User Stories," in International Conference on Software Engineering Advances, 2016, pp. 149–153.

[15] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, "Indexing by latent semantic analysis," Journal of the American society for information science, vol. 41, no. 6, 1990, pp. 391–407.

[16] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, “Distributed representations of words and phrases and their compositionality,” in Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States., 2013, pp. 3111–3119. [Online]. Available: http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality

[17] T. Mikolov, “Google code: Word2vec,” https://code.google.com/archive/p/word2vec/, [retrieved: February 2019], 2013.

[18] Explosion AI, “spaCy – Industrial-Strength Natural Language Processing,” spacy.io, [retrieved: February 2019], 2019.

[19] S. Brants, S. Dipper, P. Eisenberg, S. Hansen-Schirra, E. König, W. Lezius, C. Rohrer, G. Smith, and H. Uszkoreit, “Tiger: Linguistic interpretation of a german corpus,” Research on language and computation, vol. 2, no. 4, 2004, pp. 597–620.

[20] J. Nothman, N. Ringland, W. Radford, T. Murphy, and J. R. Curran, “Learning multilingual named entity recognition from Wikipedia,” Artificial Intelligence, vol. 194, 2012, pp. 151–175. [Online]. Available: http:/dx.doi.org10.1016/j.artint.2012.03.006

[21] J. Nielsen, “How many test users in a usability study?” https://www.nngroup.com/articles/how-many-test-users/, website. [retrieved: February, 2019], 2012.

[22] M. Kusner, Y. Sun, N. Kolkin, and K. Weinberger, “From word embeddings to document distances,” in International Conference on Machine Learning, 2015, pp. 957–966.