Semantic Similarities in Natural Language Requirements

Henning Femmer(✉), Axel Müller, and Sebastian Eder

Qualicen GmbH, Lichtenbergstr. 8, 85748 Garching, Germany
{henning.femmer,axel.muller,sebastian.eder}@qualicen.de

Abstract. Semantic similarity information supports requirements tracing and helps to reveal important requirements quality defects such as redundancies and inconsistencies.

Previous work has applied semantic similarity algorithms to requirements, however, we do not know enough about the performance of machine learning and deep learning models in that context.

Therefore, in this work we create the largest dataset for analyzing the similarity of requirements so far through the use of Amazon Mechanical Turk, a crowd-sourcing marketplace for micro-tasks. Based on this dataset, we investigate and compare different types of algorithms for estimating semantic similarities of requirements, covering both relatively simple bag-of-words and machine learning models.

In our experiments, a model which relies on averaging trained word and character embeddings as well as an approach based on character sequence occurrences and overlaps achieve the best performances on our requirements dataset.

Keywords: Requirements engineering · Similarity detection · Machine learning

1 Introduction

Since a requirements specification defines the outcome of a particular product development process, it is necessary that the contained requirements fulfill important *quality factors* [1,2]. This is important because requirements are worthless if they are, for instance, not understandable or the defined set of requirements is not complete. In that case, the developers or manufacturers could misunderstand the desired characteristics and thus create a product that diverges from the expected result. Therefore, the quality of the requirements specification needs to be assured which is typically accomplished by inspecting and validating the created requirements with respect to different quality characteristics. Accordingly, requirements can have different *defects* if they do not satisfy these characteristics [3,4].

Several of these characteristics are related to the semantic similarity of requirements. For example, semantic similarity information would help to identify redundant requirements which impair the maintainability of a requirements

D. Winkler et al. (Eds.): SWQD 2020, LNBIP 371, pp. 87–105, 2020.
https://doi.org/10.1007/978-3-030-35510-4_6

document since changes would have to be carried out for all duplicate items. Furthermore, automatic similarity estimations could help revealing inconsistencies within the specification by helping to track similar requirements which might turn out to be contradictory regarding particular details. Besides that, the process of requirements tracing could be supported by automatically suggesting links between similar project artifacts like requirements, test cases or designs. By generating these traces in a faster and easier way due to automatic similarity analyses, requirements engineers would be supported in understanding the relationships between different artifacts and can thus better detect duplicates, inconsistencies or missing items. When extending this idea to specifications of different projects, the information about the similarity of their artifacts may reveal reusable components of prior projects thus helping to reduce project effort.

Therefore, having information about the semantic similarities of requirements could help to support requirements engineers or analysts during the requirements review [5].

As we discuss in Sect. 2, previous works looking into this topic focused mostly on information retrieval approaches. However, modern advances in machine learning, e.g. Alpha Go [6] give us a glimpse of the potential of machine learning. Each year, the most promising approaches for similarity detection are discussed in the SemEval community. In this work, we want to test their knowledge in the domain of requirements engineering.

1.1 Contribution of This Work

This paper provides a novel analysis of the performance of a variety of similarity detection algorithms, including both baseline information retrieval algorithms, but also machine learning based approaches, on a large dataset of 1000 pairs of natural language requirements.

1.2 Structure of This Work

In Sect. 2, we describe related work with respect to both requirements-independent semantic similarity algorithms as well as already applied similarity approaches within the domain of requirements engineering. In Sect. 3, we explain the required background knowledge to understand the content of this paper in particular regarding our applied semantic similarity algorithms. The design of our experiment which is intended to evaluate the different algorithms on a requirements dataset is portrayed in Sect. 4. We present and analyze the results of this experiment in Sect. 5. Based on these results, we conclude and come up with several interpretations which are discussed in Sect. 6. Finally, in Sect. 7, we summarize the content and gathered insights of this work.

2 Related Work

Our related work can be divided into semantic similarity approaches using general text data and approaches only focusing on requirements data.

2.1 Semantic Similarity Estimation of General Texts

The approach and study of our paper draws on the Natural Language Processing task of Semantic Textual Similarity introduced by Agirre et al. For this task, algorithms try to estimate the grade of semantic similarity between given sentence pairs [7]. However, models that have been proposed for this task have not been investigated in the context of requirements engineering yet.

Several introduced semantic similarity models use machine learning techniques with manually designed and engineered features. These features often rely on string-based lexical information such as word and character overlaps, on knowledge-based semantic word relations based on lexical-semantic resources like WordNet, on corpus-based vector space models like Latent Semantic Analysis, or on syntactic similarities and dependencies [7,8].

Other researchers have proposed artificial intelligence models that are capable of capturing semantic differences of sentences based on word order or sentence structures [9]. Such algorithms can, for example, use sentence vectors provided by models such as Nie and Bansal's sentence encoder [10], employ interaction modules for computing word and phrase relationships of sentences like in Parikh et al.'s model [11] or apply combinations of such components such as the neural network model proposed by He and Lin [12].

2.2 Semantic Similarity Estimation for Requirements

Several semantic similarity approaches have specifically been proposed for the domain of requirements engineering and often utilize lexical similarity measures.

Mihany et al. introduced a system for identifying reusable projects and components by the similarity of their requirements which was calculated based on word overlaps [13,14].

Natt och Dag et al. compared different lexical similarity measures for identifying equivalent requirements [5]. They further refined these approaches in order to map customer wishes to product requirements which relate to the same functional requirements. For that, they constructed and compared sentence vectors based on word occurrences and frequency weights [15,16].

Hayes et al. compared several similarity methods for the requirements tracing process in order to automatically identify potential links between similar artifacts. They experimented with term frequencies and weights (TF-IDF), Latent Semantic Indexing (LSI), incorporating thesaurus information as well as relevance feedback analysis. Thereby, artifacts were represented by word occurrence vectors [17]. Eder et al. also applied LSI for automatic requirements tracing intending to automate the determination of LSI configurations [18].

Mezghani et al. proposed a k-means clustering algorithm for detecting redundancies and inconsistencies in requirements. They applied their algorithm on combinations of given requirements and their extracted business terms using the Euclidean distance as a similarity metric [19].

Juergens et al. investigated clone detection for requirements specifications. Their approach tried to identify duplicates by analyzing suffix trees which were constructed based on the word sequences of requirements [20].

Falessi et al. experimented with different NLP techniques regarding the identification of equivalent requirements. Their applied approaches comprised combinations of algebraic models, term extraction techniques, weighting schemes and similarity metrics. Falessi et al. reported a bag-of-words approach as the best single NLP technique, however, they pointed out that a combination of different NLP techniques outperformed all available individual approaches [21].

2.3 Research Gap

Researchers on semantic textual similarity tasks have proposed different state-of-the-art machine learning approaches that have shown to outperform simpler information retrieval methods on general text data. Nevertheless, it has never been investigated whether such approaches can also yield superior performances when applied to requirements data.

3 Background

For this work, two concepts are relevant. First, we need to define similarity. Second, we need to define the algorithms that we want to apply.

3.1 Semantic Similarity

For the definition of semantic similarity within this work, we utilize an ordinal similarity scale with six different values. This scale has been introduced by Agirre et al. for the SemEval research workshops on semantic textual similarity and have been successfully applied in this linguistic community since 7 years (details e.g. in [22]).

The applied semantic similarity scale is shown in Table 1. As can be seen, the different levels reach from total dissimilarity in meaning to complete meaning equivalence. The intermediate similarity grades represent various degrees of partial similarity and meaning overlap [7], for example, considering the topics and details of given texts.

Table 1. Ordinal semantic similarity scale

Score	Explanation
0	The two sentences are completely dissimilar
1	The two sentences are not equivalent, but are on the same topic
2	The two sentences are not equivalent, but share some details
3	The two sentences are roughly equivalent, but some important information differs/is missing
4	The two sentences are mostly equivalent, but some unimportant details differ
5	The two sentences are completely equivalent, as they mean the same thing

3.2 Applied Algorithms

In this work, we compare the algorithms listed in Table 2. The selection is based on the most common and successful algorithms from the SemEval community [7], since these are obviously the most promising approaches. We cannot explain all used algorithms in detail. For a deeper introduction into this, please refer to the respective original works. The selected algorithms listed in Table 2 vary between *baselines*, *pre-trained*, *self-trained*, and *non-trained* approaches:

Baseline approaches are very simplistic approaches, e.g. counting tokens, that help to reflect on the complexity of the problem and the actual advantage of more sophisticated and complex approaches. **Pre-Trained** approaches came with already trained machine learning models provided by the original authors. **Self-Trained** approaches are machine learning algorithms that we trained ourselves using data that has been published for the SemEval workshops. **Non-Trained** approaches do not require training for applications.

4 Study Design

In this chapter, we describe the structure and setup of our study which we use to compare the performances of different semantic similarity algorithms on requirements data. The description and design of our study correspond to the experiment process as introduced by Wohlin et al. [32].

According to the *Goal Question Metric* approach of Basili et al. [33], we first define the goal of our study as well as related research questions that we will investigate and answer based on the obtained results measured by appropriate metrics. Afterwards, we describe the context and setup of our study including selected subjects, objects and instruments.

4.1 Goal Definition

To understand the overall setting and intention of our experiment, we first define the goal of this study:

Table 2. The different algorithms used in this study grouped into *baseline approaches*, as well as *pre-trained*, *self-trained*, and *non-trained* approaches, each in alphabetical order.

Algorithm	Description
Char ngram BOW	A baseline token-occurrence-based model incorporating both character trigrams and fourgrams as features into the sentence vectors whereby the corresponding vector values represent binary occurrence indicators of these tokens
Word2vec CBOW	As a baseline, Word2vec continuous bag of words (CBOW) is a widely used word embeddings approach
BiLSTM Avg	A pre-trained sentence encoder by Wieting et al. that uses a bi-directional LSTM and concatenates the hidden states of the forward and backward LSTM [23,24]
Charagram	A pre-trained sentence encoder by [25] based on character n-gram embeddings which are added together in order to retrieve sentence vectors
InferSent	A pre-trained sentence encoder model that is a bi-directional LSTM trained on Natural Language Inference data [26]
USE	As a pre-trained sentence encoder, Universal Sentence Encoders (USE) is an approach focussing on task and context generalizability [27]
Word-trigram	This pre-trained sentence encoder combines word and character trigram embeddings by averaging all embeddings for the character sequences and words contained in the given sentence, which outperformed other models on SemEval tasks [23,24]
DecAttn	A self-trained supervised algorithm only based on word and phrase alignments which are used to partition the problem into subtasks [11]
MPCNN	As a self-trained supervised algorithm, Multi-Perspective Convolutional Neural Networks are a CNN specifically tuned for semantic similarities [8]
PWIM and Subword PWIM	As a self-trained supervised algorithm, Pairwise Word Interaction Model (PWIM) is similar to MPCNNs, but directly applies word-interaction computations on the individual word context representations of the given sentences [12]. The Subword PWIM model uses the same functionality but has been adapted to work with character sequence embeddings [28]
Random Forest	Self-trained supervised algorithm that creates multiple trees on specific subsets of the sample data and aggregates the results. We apply the NLP features proposed at SemEval 2017 [29]
SSE	As a self-trained supervised algorithm, Shortcut-stacked Sentence Encoder is an ML approach originally developed for multi-domain natural language inference tasks [10]
Tree LSTM	A self-trained supervised algorithm that processes sentences according to the syntactic sentence structure [9]
Word Aligner	This is a non-trained model that has worked very well on previous similarity tasks outside the RE world [30,31]

Our goal is to analyze *semantic similarity algorithms*
for the purpose of *evaluating and comparing their performances*
with respect to *the accuracy of their predicted semantic similarity scores*
from the point of view of *laymen*
in the context of *natural language requirements pairs with human-annotated semantic similarity labels.*

4.2 Research Questions

In this work, we focus on the following research questions:

– **RQ1:** How do semantic similarity algorithms trained on non-requirements data perform in comparison to algorithms trained on requirements data?
– **RQ2:** Which algorithm performs most accurately for predicting the semantic similarities of natural language requirements?

4.3 Metrics

In order to answer the research questions, we test and analyze the performance of each algorithm based on its semantic similarity prediction accuracy. For this, we apply mean squared error (for a discussion of the adequacy, check [34]) as the performance metric for the algorithms, where n indicates the number of samples and y_i and $\hat{y}_i$ represent the expected and the predicted scores respectively:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

4.4 Experiment Design and Execution

The overall procedure of our experiment is illustrated in Fig. 1 and will be further explained in the following sections.

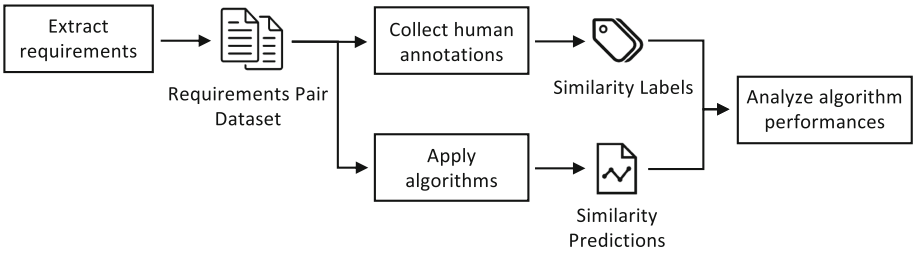

Fig. 1. Overview of the experiment procedure

Requirements Pair Dataset and Human Similarity Annotations. In order to collect human scores, we extract requirements from several requirements specifications and assemble 1000 different requirements pairs. We upload this dataset to the crowdsourcing marketplace *Amazon Mechanical Turk*[1] where human workers, called *turkers*, assign semantic similarity scores to each of our requirements pairs. For each requirements pair, we collect annotations from five different workers and take the median. Annotators are asked to assign a semantic similarity category $S \in \{0, 1, 2, 3, 4, 5\}$ to each requirements pair according to the similarity definition given in Table 1. We argue that this does not require the participants to have any expertise in the domains of linguistics or requirements analysis (c.f. Dagan et al. [35]).

[1] https://www.mturk.com/ (accessed 06 February 2019).

We choose to retrieve annotations this way due to the findings of Agirre et al. in the context of their preparations for the SemEval workshops. They have shown that similarity scores with good rates of agreement among the annotators can be observed for similar semantic similarity annotation tasks (cf. e.g. Sect. 2.1).

Each turker who participates in our task on Amazon Mechanical Turk gets paid $0.04 per annotated requirements pair. The total number of annotations per worker over all of our uploaded requirements pairs is not restricted, however, one particular worker can only annotate each requirements pair once.

Similarity Annotation Retrieval. We executed several trial runs with only five requirements pairs each. This is intended to evaluate and compare the performance of the turkers, which enabled us to evaluate the required qualification for the turkers. When we required masters qualification from our turkers, the turkers provided us annotations on average close to our own similarity estimations for the corresponding requirements pairs.

Algorithm Application and Performance Comparison. Afterwards, we apply a variety of different algorithms on this dataset in order to compare their performances.

Balancing. We collect annotations in batches of 100 sentence pairs each, which allows us to control for the balance of similarity scores by appropriately choosing the requirements pairs for our subsequent batches. This means that we check the distribution of similarity scores after every completed batch. Based on that, we create the next batch with more pairs of the less frequent categories and less pairs of the more frequent categories according to our own similarity judgements for the corresponding requirements. However, because our own assessment of these pairs may diverge from the final annotations of the turkers, we cannot completely influence and control the final balance of semantic similarity scores.

Randomization. Before we upload our dataset to Amazon Mechanical Turk, we shuffle the requirements pairs in each batch so that requirements taken from the same document are less likely to be clustered together.

4.5 Study Subjects

Due to Amazon Mechanical Turk, our subjects are primarily laymen. In trial runs, we have retrieved the best results regarding the agreement among annotators when requiring a so-called Masters qualification. Consequently, we take this as a prerequisite for our tasks.

4.6 Study Objects

Our objects are requirements that we extract from 14 different requirements specifications available on the Natural Language Requirements Dataset [36]. These include both real-world industrial requirements specifications and specifications from university projects. We select the software requirements specifications based on our impression of how suitable their requirements would be for getting annotated by laymen. Accordingly, the requirements to be incorporated in our dataset must be understandable without a background briefing. However, we incorporate both requirements that are easy to understand as well as requirements that are more complicated based on their sentence structure and content. Table 3 shows all of the requirements documents that we use for collecting requirements for our dataset.

Table 3. Sources of the requirements in our evaluation dataset

Document	Domain	Number of req.
Pontis	Highway bridge information management system	274
E-store	Online store for consumer electronics	112
Sprat	Goals and scenario management tool	98
NASA	Spacecraft software	86
TCS	Aircraft control software	75
Nenios	Child care management software	71
agentMom	Multi agent communication systems	59
Philips	Messenger software application	42
Mahjong	Web software system for Chinese board game	37
Digital home	Home management system	35
Puget sound	Courseware system	32
Blit	Laboratory information system	29
Colorcast	Web application for paint selection	26
Video search	Video search software	24

As described before, we balance the number of future semantic similarity scores to the extent possible while building the requirements pair dataset based on our own similarity estimations. However, this is difficult because we cannot predict the scores that will be obtained from the annotators. Hence, our evaluation dataset turned out to have a higher number of requirements pairs annotated with the similarity categories 1 or 2, whereas especially the number of requirements pairs annotated with category 4 is small compared to this. The histogram of received semantic similarity categories is illustrated in Fig. 2.

5 Results

In this section, we report the performances of our applied algorithms on our created requirements pair dataset based on the defined metrics and appropriate visualizations.

5.1 Presentation of Results

In the following, we present the performance accuracies of the algorithms introduced in the Sect. 3 when applied to our assembled requirements pair dataset.

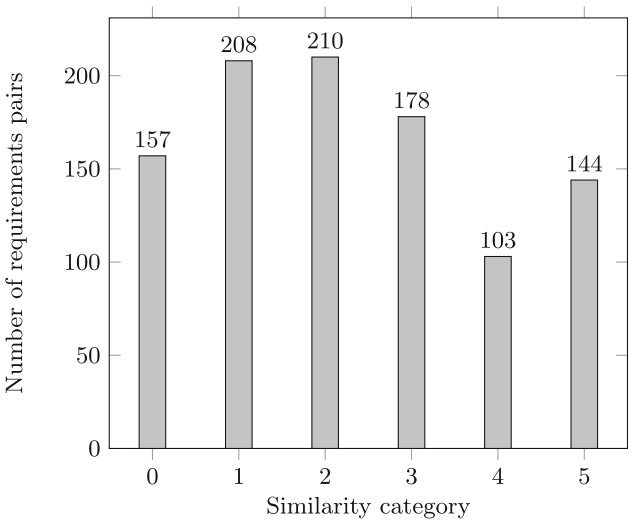

Fig. 2. Distribution of requirements pairs over similarity categories

Performance Accuracy. As described in Sect. 4.3, we use the mean squared error (MSE) as the performance metric to measure the accuracy of the predictions of our applied algorithms based on the collected similarity labels for our requirements evaluation dataset.

Overall and Weighted Mean Squared Error Metrics. Because of the unbalanced distribution of requirements pairs over the six similarity categories, only regarding the overall mean squared error would give a distorted result since there would be a bias towards the more frequent categories. Thus, we calculate the weighted mean squared error by summing up the individual mean squared errors MSE_c for each semantic similarity category c of requirements pairs which have been labeled with this category and dividing this result by 6 according to the number of similarity categories:

$$\text{weighted MSE} = \frac{1}{6} * \sum_{c=0}^{5} \text{MSE}_c$$

We include the overall mean squared error as well as the weighted mean squared error for each algorithm in Table 4 denoted as MSE and MSE_w.

The smaller the mean squared error, the less do the algorithms' predictions diverge from the assigned similarity labels treated as the ground truth. Accordingly, smaller MSEs indicate better algorithm performances.

Performance Results. In Table 4 we list the mean squared error values for every algorithm as described before. We denote the adjusted algorithm settings (where we chose hyper-parameters for the algorithms) by adding *(a)* to the corresponding model names.

5.2 Answers to Research Questions

In this section, we answer our research questions which have been introduced in the previous chapter on the basis of the experiment results presented above.

We will describe the differences between the algorithms relative to each other. Therefore, when we say that the MSE of an algorithm is relatively small or relatively big this refers to the comparison of its performance to the other algorithms.

RQ1: How Do Semantic Similarity Algorithms Trained on Non-requirements Data Perform in Comparison to Algorithms Trained on Requirements Data? For answering this research question, we only consider our self-trained algorithms and the pre-trained sentence encoders because the monolingual word aligner and the baseline algorithms do not need to be trained.

From analyzing the mean squared error results of these algorithms, it can be inferred that most of the evaluated models perform rather well in the context of this work because the majority of the weighted mean squared error values is below or around 1.5. This means that on average the predictions of the corresponding algorithms do not diverge much more than one similarity category from the expected similarity score as interpreted by humans.

In particular, the pre-trained Word-trigram sentence encoder model with both its original and adjusted application settings shows a very good performance on our requirements data, achieving a weighted mean squared error of 0.94 with its adjusted settings and of 0.96 with the standard model. Behind these two model versions, the BiLSTM Avg model ranks third among all trained algorithms and completes the set of the best three algorithms within this experiment, producing prediction results with a weighted MSE of 0.98.

As can be seen in Table 4, for almost all of the trained algorithms we were able to come up with adjusted training or application settings so that better performances on the evaluation dataset could be achieved. For example, especially for the Subword PWIM and the standard PWIM models, the performance differences between the unadjusted and the adjusted settings are very large.

Table 4. Table shows performances on our requirements evaluation dataset ordered by weighted mean squared error. Models that have been self-trained or applied with adjusted settings are marked with (a).

Algorithm	MSE	MSE_w	Type
Word-trigram (a)	0.96	0.94	Pre-Trained
Char ngram BOW	0.94	0.94	Baseline
Word-trigram	0.95	0.96	Pre-Trained
BiLSTM Avg	0.98	0.98	Pre-Trained
Charagram (a)	0.97	1.06	Pre-Trained
Charagram	0.95	1.08	Pre-Trained
Word Aligner (a)	0.99	1.10	Non-Trained
InferSent (a)	1.10	1.14	Pre-Trained
Word Aligner	1.03	1.15	Non-Trained
Subword PWIM (a)	1.20	1.15	Self-Trained
PWIM (a)	1.20	1.16	Self-Trained
Random Forest (a)	1.29	1.21	Self-Trained
Word2vec CBOW	1.25	1.28	Baseline
MPCNN (a)	1.33	1.29	Self-Trained
USE (a)	1.28	1.30	Pre-Trained
USE	1.29	1.31	Pre-Trained
Random Forest	1.38	1.31	Self-Trained
SSE (a)	1.49	1.36	Self-Trained
Tree LSTM (a)	1.44	1.45	Self-Trained
PWIM	1.64	1.56	Self-Trained
Subword PWIM	1.78	1.63	Self-Trained
MPCNN	1.75	1.63	Self-Trained
SSE	1.98	1.85	Self-Trained
Tree LSTM	2.17	2.03	Self-Trained
DecAttn (a)	2.59	2.50	Self-Trained
InferSent	2.77	2.55	Pre-Trained
DecAttn	3.08	2.90	Self-Trained

RQ2: Which Algorithm Performs Most Accurately for Predicting the Semantic Similarities of Natural Language Requirements? The best-performing algorithms in our experiment were the adjusted Word-trigram model and the Char ngram BOW baseline which both achieved the smallest weighted MSE of 0.94.

Furthermore, the third best algorithm, the BiLSTM Avg model, completes the set of algorithms which reached a weighted MSEs below 1.0. This approach provides slightly better results for the lower similarity categories 0, 1 and 2 but performs less accurately for the other, higher similarity categories.

Apart from this, when further analyzing Table 4, it can be seen that there are more algorithms whose performances do not greatly differ from the best results described above.

All in all, despite the minor differences to other model performances, our experiment results have shown that the Word-trigram and the Char ngram BOW models perform most accurately within the scope of this experiment.

6 Interpretation

In this section, we interpret the results and observations described before for each individual research question. Furthermore, we discuss the threats to validity which apply to our conducted experiment.

6.1 RQ1: How Do Semantic Similarity Algorithms Trained on Non-requirements Data Perform in Comparison to Algorithms Trained on Requirements Data?

Our experiment reveals that distinct trained semantic similarity algorithms achieve very different performances. Within this section, we identify and discuss various findings regarding the characteristics of these algorithms and their influence on the performance results.

For our self-trained models, we used the same training data, classification layer, loss function, and training objective. However, these models exhibit substantially different performances. Thus, we conclude, that the actual architecture (esp. of neural networks) of the underlying models impacts accuracy.

Algorithms that do not consider the word order can perform equal or better than algorithms sensitive to the word order. This is especially true for the bag of words based algorithms. Thus, we conclude that word order is not important to detect similar requirements, in contrast to other NLP tasks.

We adjusted the parameters of various algorithms to make them perform better on requirements data. This included the pre-processing steps. Thereby we noticed that for the different algorithms, different pre-processing steps have a positive influence on their accuracy. However, the best choices of pre-processing steps which yield the highest performance gains largely differ between models. We assume that this is likely linked to the way of how models process the input texts and how they model input representations so that, for instance, some models prefer to keep stop words and original word forms in order to better understand sentence structures and word relationships.

6.2 RQ2: Which Algorithm Performs Most Accurately for Predicting the Semantic Similarities of Natural Language Requirements?

As already identified in Sect. 5.2, the models with the best overall performance in our experiment are the Word-trigram and Char ngram BOW models. In the following we further discuss these algorithms and their performances.

We believe that the Word-trigram model might perform better than the Char ngram BOW baseline when they are both applied to other datasets by using the same implementation settings like in our experiment. This is because the Word-trigram model has already proven its transferability from its training dataset to another dataset, that is, our requirements dataset whereas the settings of the Char ngram BOW model are completely adjusted and dependent on our evaluation dataset.

For this reason, we consider the Word-trigram model as the best overall model not only because it outperformed all other models applied within our experiment, but also because of its proven transferability. Moreover, we think that its large paraphrase training corpus and the combination of word and character embeddings allow it to capture important and meaningful characteristics of words and sentences that are crucial for the determination of semantic similarity. Because of this embedding information, we assume that this model can better capture important word semantics and semantic relations between words compared to the simpler token occurrence-based approaches which merely rely on lexical token overlaps. This might be even more important for other requirements specifications which may use less consistent terminology.

6.3 Threats to Validity

We discuss the validity threats according to the different issues described by Wohlin [32].

Reliability of Measures. In our experiment, we apply the ordinal similarity scale which is used to collect human interpretations of semantic similarity for given requirements pairs. This measure can be unreliable because humans may interpret semantic similarities differently. However, we collect similarity annotations from five different raters for every requirements pair and take the median in order to retrieve the final similarity label.

The inter-rater agreement according to the Kendall's coefficient of concordance W of 0.607 suggests that there is a correlation between the scores of the different annotators for each requirements pair. Thus, there is a good degree of agreement between the raters regarding the semantic similarities of our requirements pairs which is why we assume that reasonable similarity labels have been obtained.

Finally, we review the obtained dataset. We note that while some scores diverge from how we would have rated the corresponding requirements pairs, the majority of these labels agrees with our own point of view. Consequently, despite the sometimes large divergences between the individual annotations of different raters, we argue that we have retrieved a suitable dataset for the purpose of this study where the potential disagreement between raters is counteracted by taking the median score.

Random Irrelevancies in Experimental Setting. This issue is concerned with possible influences on the result due to external disturbances like noise or interruptions. Since our subjects can log on to Amazon Mechanical Turk and participate in our experiment from any place and at any time, we cannot control their environment and outside influences.

Random Heterogeneity of Subjects. Since all Amazon Mechanical Turk workers who fulfill the defined qualification requirements of our experiment are able to accept and participate in our created task, there might be a certain heterogeneity of subjects.

We tried to mitigate the effects of individual differences by requiring the Master qualification as well as by taking the median from five different annotators for every requirements pair. Furthermore, we conducted several trial runs for obtaining similarity annotations where we investigated and selected the most suitable qualification requirements In these trial runs, the median value of the obtained scores for the best selected qualification requirements seemed to be reasonable and suitable for the tested pairs.

Mono-operation Bias. The mono-operation bias describes the problem of not representing the construct broadly enough, for instance, by only including one subject, variable or object. In our experiment, we only used requirements from the domain of information technology. Hence, results might not generalize.

Mixed Scales. In our experiment, we use an ordinal similarity scale to record the similarity annotations assigned by our subjects so that every annotation corresponds to one of the six similarity categories. Our applied algorithms produce predictions according to a similar idea of similarity, however, their similarity estimations are continuous so that they can lay between categories and thus correspond to an interval scale. Since we calculate the mean squared error based on the ordinal human similarity annotations and the continuous algorithm predictions, we compare values from an ordinal scale to values from an interval scale. This constitutes an error according to measure theory and thus poses a threat to the validity of our results.

Interaction of Selection and Treatment. In our case, we utilized laymen as subjects for our study. However, the study results are intended for evaluating the suitability of the tested semantic similarity algorithms for requirements engineering in industry where requirements analysts and experts are concerned with the topic of semantic similarities of requirements. Due to their background knowledge and experience, requirements experts might interpret semantic similarities of requirements differently than laymen. Thus, this might negatively influence the generalizability of our results to industrial practice.

Interaction of Setting and Treatment. This threat concerns the risk of using a different experimental setting or relying on non-representative objects during the study compared to what is standard in industrial practice. In our experiment, we used requirements from industrial requirements specifications or specifications from university that are similar to industrial specifications. Most of these specifications have been created for real-life projects. Thus, we believe that they are at least to some extent representative of requirements used in industrial practice.

7 Summary

In this work, we researched and investigated suitable approaches for automatically estimating the semantic similarities of requirements pairs. In order to evaluate and compare these approaches, we designed an experiment in which the algorithms' predictions were measured against human similarity interpretations that were treated as the ground truth. For this purpose, we assembled an evaluation requirements dataset containing 1000 distinct requirements pairs which were extracted from several requirements specifications for industrial and university projects. For this dataset, we obtained similarity labels from human annotators according to an ordinal similarity scale from 0 to 5 using Amazon Mechanical Turk as a crowdsourcing platform.

The requirements pair dataset was used to determine the performances of our selected and applied algorithms by calculating the mean squared error between their predictions and the corresponding human similarity labels. Due to the unbalanced distribution of the requirements pairs in our evaluation dataset over the similarity categories, we calculated a weighted mean squared error which determines and averages individual MSE values for each similarity class. Based on these performance results, we were able to draw different conclusions regarding our research questions which we summarize in the following.

RQ1: How do Semantic Similarity Algorithms Trained on Non-requirements Data Perform in Comparison to Algorithms Trained on Requirements Data? We found that the different algorithms perform very differently on requirements data both regarding their overall performances as well as regarding their performances for individual similarity categories. This indicated that the models have different prediction tendencies regarding the various similarity categories. Furthermore, we suggested that the performances of algorithms which do not capture characteristics about word order and sentence structures do not seem to be negatively influenced because these types of information do not seem to noticeably affect the semantic similarity of requirements pairs.

RQ2: Which Algorithm Performs Most Accurately for Predicting the Semantic Similarities of Natural Language Requirements? In our study, the Word-trigram sentence encoder model developed by Wieting et al. [23] as well as the Char ngram BOW baseline approach achieved the best overall performance accuracy

with a weighted mean squared error of 0.94. The Word-trigram model combines word embeddings with character trigram embeddings and averages these combinations in order to retrieve sentence vector representations whereas the Char ngram BOW method is based on lexical character sequence overlaps. Despite the equal performance results, we believe that the Word-trigram model would provide better performances in practice due to its use of token embeddings which capture individual word and sentence semantics instead of just relying on token occurrences and overlaps.

References

1. Femmer, H., Vogelsang, A.: Requirements quality is quality in use. IEEE Softw. **36**(3), 83–91 (2018)
2. Femmer, H., Fernández, D.M., Wagner, S., Eder, S.: Rapid quality assurance with requirements smells. J. Syst. Softw. **123**, 190–213 (2017)
3. Femmer, H.: Automatic requirements reviews - potentials, limitations and practical tool support. In: Felderer, M., Méndez Fernández, D., Turhan, B., Kalinowski, M., Sarro, F., Winkler, D. (eds.) PROFES 2017. LNCS, vol. 10611, pp. 617–620. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-69926-4_53
4. Wiegers, K.E., Beatty, J.: Software Requirements. Microsoft Press, Redmond (2013)
5. Natt och Dag, J., Regnell, B., Carlshamre, P., Andersson, M., Karlsson, J.: A feasibility study of automated natural language requirements analysis in market-driven development. Requir. Eng. **7**(1), 20–33 (2002)
6. Silver, D., et al.: Mastering the game of go with deep neural networks and tree search. Nature **529**(7587), 484 (2016)
7. Cer, D., Diab, M., Agirre, E., Lopez-Gazpio, I., Specia, L.: SemEval-2017 task 1: semantic textual similarity multilingual and crosslingual focused evaluation. In: Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval 2017), pp. 1–14. Association for Computational Linguistics (2017)
8. He, H., Gimpel, K., Lin, J.: Multi-perspective sentence similarity modeling with convolutional neural networks. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, pp. 1576–1586 (2015)
9. Tai, K.S., Socher, R., Manning, C.D.: Improved semantic representations from tree-structured long short-term memory networks. CoRR abs/1503.00075 (2015)
10. Nie, Y., Bansal, M.: Shortcut-stacked sentence encoders for multi-domain inference. In: Proceedings of the 2nd Workshop on Evaluating Vector Space Representations for NLP, pp. 41–45. Association for Computational Linguistics (2017)
11. Parikh, A., Täckström, O., Das, D., Uszkoreit, J.: A decomposable attention model for natural language inference. In: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pp. 2249–2255. Association for Computational Linguistics (2016)
12. He, H., Lin, J.: Pairwise word interaction modeling with deep neural networks for semantic similarity measurement. In: Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 937–948. Association for Computational Linguistics (2016)

13. Mihany, F.A., Moussa, H., Kamel, A., Ezat, E.: A framework for measuring similarity between requirements documents. In: Proceedings of the 10th International Conference on Informatics and Systems. INFOS 2016, pp. 334–335. ACM, New York (2016)
14. Mihany, F.A., Moussa, H., Kamel, A., Ezzat, E., Ilyas, M.: An automated system for measuring similarity between software requirements. In: Proceedings of the 2nd Africa and Middle East Conference on Software Engineering, AMECSE 2016, pp. 46–51. ACM New York (2016)
15. Natt och Dag, J., Gervasi, V., Brinkkemper, S., Regnell, B.: Speeding up requirements management in a product software company: linking customer wishes to product requirements through linguistic engineering. In: Proceedings of 12th IEEE International Requirements Engineering Conference, September 2004, pp. 283–294 (2004)
16. Natt och Dag, J., Regnell, B., Gervasi, V., Brinkkemper, S.: A linguistic-engineering approach to large-scale requirements management. IEEE Softw. **22**(1), 32–39 (2005)
17. Hayes, J.H., Dekhtyar, A., Sundaram, S.K.: Advancing candidate link generation for requirements tracing: the study of methods. IEEE Trans. Softw. Eng. **32**(1), 4–19 (2006)
18. Eder, S., Femmer, H., Hauptmann, B., Junker, M.: Configuring latent semantic indexing for requirements tracing. In: Proceedings of the Second International Workshop on Requirements Engineering and Testing, RET 2015, pp. 27–33. IEEE Press, Piscataway (2015)
19. Mezghani, M., Kang, J., Sèdes, F.: Industrial requirements classification for redundancy and inconsistency detection in SEMIOS. In: 26th IEEE International Requirements Engineering Conference, RE 2018, Banff, AB, Canada, 20–24 August 2018, pp. 297–303 (2018)
20. Juergens, E., et al.: Can clone detection support quality assessments of requirements specifications? In: Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 2, ICSE 2010, pp. 79–88. ACM, New York (2010)
21. Falessi, D., Cantone, G., Canfora, G.: Empirical principles and an industrial case study in retrieving equivalent requirements via natural language processing techniques. IEEE Trans. Softw. Eng. **39**(1), 18–44 (2013)
22. Agirre, E., Diab, M., Cer, D., Gonzalez-Agirre, A.: SemEval-2012 task 6: a pilot on semantic textual similarity. In: Proceedings of the First Joint Conference on Lexical and Computational Semantics - Volume 1: Proceedings of the Main Conference and the Shared Task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation, SemEval 2012, pp. 385–393. Association for Computational Linguistics, Stroudsburg (2012)
23. Wieting, J., Gimpel, K.: Pushing the limits of paraphrastic sentence embeddings with millions of machine translations. CoRR abs/1711.05732 (2017)
24. Wieting, J., Mallinson, J., Gimpel, K.: Learning paraphrastic sentence embeddings from back-translated bitext. In: Proceedings of Empirical Methods in Natural Language Processing. (2017)
25. Wieting, J., Bansal, M., Gimpel, K., Livescu, K.: Charagram: embedding words and sentences via character n-grams. CoRR abs/1607.02789 (2016)

26. Conneau, A., Kiela, D., Schwenk, H., Barrault, L., Bordes, A.: Supervised learning of universal sentence representations from natural language inference data. In: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, Copenhagen, Denmark, September 2017, pp. 670–680. Association for Computational Linguistics (2017)
27. Cer, D., et al.: Universal sentence encoder. CoRR abs/1803.11175 (2018)
28. Lan, W., Xu, W.: Character-based neural networks for sentence pair modeling. In: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers), pp. 157–163. Association for Computational Linguistics (2018)
29. Al-Natsheh, H.T., Martinet, L., Muhlenbach, F., ZIGHED, D.A.: UdL at SemEval-2017 task 1: semantic textual similarity estimation of English sentence pairs using regression model over pairwise features. In: Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017), Vancouver, Canada, August 2017, pp. 115–119. Association for Computational Linguistics (2017)
30. Brychcín, T., Svoboda, L.: UWB at SemEval-2016 task 1: semantic textual similarity using lexical, syntactic, and semantic information. In: SemEval@NAACL-HLT, pp. 588–594. The Association for Computer Linguistics (2016)
31. Sultan, M.A., Bethard, S., Sumner, T.: Back to basics for monolingual alignment: exploiting word similarity and contextual evidence. Trans. Assoc. Comput. Linguist. **2**, 219–230 (2014)
32. Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslén, A.: Experimentation in Software Engineering. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29044-2
33. Basili, V.R., Caldiera, G.: Rombach, D.H.: The goal question metric approach. In: Encyclopedia of Software Engineering, pp. 528–532 (1994)
34. Shepperd, M., MacDonell, S.: Evaluating prediction systems in software project estimation. Inf. Softw. Technol. **54**(8), 820–827 (2012)
35. Dagan, I., Dolan, B., Magnini, B., Roth, D.: Recognizing textual entailment: rational, evaluation and approaches. J. Nat. Lang. Eng. **4**, I-Xvii (2010)
36. Ferrari, A., Spagnolo, G.O., Gnesi, S.: PURE: a dataset of public requirements documents. In: IEEE 25th International Requirements Engineering Conference (RE), pp. 502–505. IEEE (2017)