Interactive optimization of embedding-based text similarity calculations

Daniel Witschard¹, Ilir Jusufi¹, Rafael M Martins¹, Kostiantyn Kucher¹,² and Andreas Kerren¹,²

Abstract
Comparing text documents is an essential task for a variety of applications within diverse research fields, and several different methods have been developed for this. However, calculating text similarity is an ambiguous and context-dependent task, so many open challenges still exist. In this paper, we present a novel method for text similarity calculations based on the combination of embedding technology and ensemble methods. By using several embeddings, instead of only one, we show that it is possible to achieve higher quality, which in turn is a key factor for developing high-performing applications for text similarity exploitation. We also provide a prototype visual analytics tool which helps the analyst to find optimal performing ensembles and gain insights to the inner workings of the similarity calculations. Furthermore, we discuss the generalizability of our key ideas to fields beyond the scope of text analysis.

Keywords
Text embedding, ensemble methods, text similarity, similarity calculations, visual analytics

Introduction
Comparing text documents for similarity is an important task since the similarity distribution within a corpus may give important insights to it, which could in turn potentially be used for a variety of applications, such as recommender systems, topic detection, plagiarism control, analysis of authorship of historical documents, semi-supervised machine learning, and others. However, comparing natural language text is an inherently complex and ambiguous task and many challenges therefore exist, mainly because similarity is an elusive concept that very much lies in the eye of the beholder. Several methods have been developed to tackle these challenges, including string-, corpus-, and knowledge-based approaches,¹ ³ for instance, and among these distributional word/text embedding technologies are especially interesting since they can handle semantic similarity and also be computationally efficient (if a pre-trained model is used). In this paper, we focus on the problem of finding similar articles within a large set of scientific publications. And to augment the quality of the similarity calculations, we propose a new method for combining several different embeddings. As we will show, this method could in turn be used to enhance and/or extend applications within fields such as Bibliometrics, Scientometrics, or Science Mapping, among others.

Embeddings are (often low-dimensional) numeric vector representations of complex and/or unstructured data, created in order to be suitable for computational analysis tasks such as clustering, classification, and similarity calculations.⁴ The main goal of embedding algorithms is usually to produce embeddings where items that are similar in the original data set (according
to some domain-specific aspect) are embedded into vectors that lie close to each other in the embedding space, with regard to some chosen distance metric. This makes embeddings highly suitable as input for computational analysis tasks such as clustering, classification, and similarity calculations since it is more straightforward to calculate a distance measure (such as Euclidean or cosine distance) with numeric vectors than it is with other types of complex and/or unstructured data. How well the embedding captures the underlying targeted similarity is crucial since poor quality embeddings will elevate the risk for poor quality results when used in any further calculations. Therefore, the search for new and better ways to embed different types of data has attracted a lot of interest in recent years. For some data types, such as graphs/networks and words/text, there exist several different algorithms, each with its specific characteristics and trade-offs. As a consequence, choosing the best embedding technology for a given application is an important and often non-trivial task. A straightforward way to handle this type of choice would be to: (1) choose a quality metric of importance for the current application, (2) evaluate all algorithms on this metric on a representative data set, and then (3) choose the one with the highest score. This intuitively appealing strategy provides a deterministic way for an optimal single-component choice and is in line with the existing works. However, one might also consider an alternative approach inspired by the question: “Would it be possible to combine several different embedding types as a means to achieve higher quality?”, and this is the path that we will explore in this paper. Our starting point for this new approach is the observation that ensemble methods (i.e., different strategies for combining the results from several classification algorithms) are a well-established and powerful means for augmenting the quality of the results for supervised classification problems. Hence, if a similar methodology could be applied to embedding-based similarity calculations, this approach could hold the potential to achieve state-of-the-art results and outperform any of the single embeddings taken on its own.

Since both embedding calculations and ensemble methods typically involve intricate computational models (i.e., neural networks or computational optimizations), our problem domain is clearly related to the general challenges of interpretability, explainability, and trustworthiness in machine learning (ML) and artificial intelligence (AI). One strategy proposed for these challenges is to make use of perceptual and cognitive abilities of human analysts, allowing them to construct and interact with ML models through the means of interactive visual analytic (VA) solutions. In particular, several VA approaches focusing on exploration or comparison of individual embedding algorithms have been discussed in the literature, including the works by Smilkov et al., Park et al., or Ji et al., for instance. Inspired by these ideas and also by the general ideas of VA we have attempted to bring the human into the process by developing a prototype VA tool which provides a visual representation of the optimization process and also displays specific directive guidance. The use of the tool gives direct insights to the inner workings of the similarity calculations, and hence it supports the construction of a mental model of this complex process. Our hope is therefore that our work will also contribute to the field of human-centered AI in the sense that it opens up the “black box” of ML a little bit and thus enables better understanding and higher trust.

The main contributions of our work are:

1. a novel way to apply ensemble methods to embeddings which can be used as a means to improve the quality of embedding-based similarity calculations; and
2. a VA tool, called EEVO (see Figure 1), that guides the search for better hyperparameter settings and allows the analyst to build a mental model of the inner workings of the ensemble calculations.

Furthermore, the generalizability of our method is promising since the general ideas could be used for any embeddable data type, as we will discuss in the final section of the paper.

The rest of this manuscript is organized as follows. In Section Related Work, we discuss the relevant work on both the underlying computational analyses as well as existing VA approaches relevant to our problem. We describe the general methodology and specific tasks supported by our proposed approach in Section General Methodology and describe the data set and the computational steps in Section Data Set and Process Steps. In Section Visualization, we present our proposed visualization approach, and the results of our user study are presented in Section Evaluation. Finally, in Section Discussion and Conclusions, we present the outcomes and limitations of this work as well as the opportunities for future work.

**Related work**

In this section, we present previous research that is related to our work.

**Word and text embedding**

In general, word embeddings are distributed representations obtained from unsupervised training of a deep
learning model on some large corpus of natural language text. By using a large amount of training data to predict words given a specific context (or vice versa), the model will be able to learn semantic similarities of word pairs, for example, *good* is similar to *super*, and *bad* is similar to *awful*. The algorithm then projects such similar word pairs to embedding vectors that lie close to each other in the embedding space. This ability to group together words with similar meaning can then be exploited for advanced tasks such as, for instance, text alignment. Arguably, the single most influential word embedding technology is Word2Vec, which was introduced in 2013, and the current state-of-the-art is the BERT model. There are different approaches on how to use word embeddings to obtain embeddings for sentences or paragraph-sized text, starting from the intuitive (but limited) approach to take the average of the embeddings of each word in the text. However, sophisticated approaches are needed in order to exploit the syntactical structure of sentences, which is crucial to do since the same set of words may be arranged to form sentences with very different meanings and the same word may have different meaning depending on the context. More specifically, this means that many words will need to have several different embeddings so that a context-dependent choice can be made for each situation. To achieve this, the
use of deep learning models is a popular choice, and approaches have, for example, been developed for recursive neural networks, convolutional neural networks, and recurrent neural networks. Arguably, the current state-of-the-art technology for text embedding is the Universal Sentence Encoder (USE), but the previously mentioned BERT algorithm also works for text of sentence length. Both of these can be downloaded as generically trained models which, if needed, can be further fine-tuned on domain-specific corpora. The models can be used as “black boxes” in the sense that they can be fed text to return an embedding vector that tries to capture the semantic meaning of the whole sentence or paragraph.

**Ensemble methods**

Ensemble methods are a well-studied and successful field of classification optimization. The main goal is to find a combination (called an ensemble) of several classifiers that provides better results than any of the individual classifiers taken on its own. The bagging approach involves classifiers of the same type of algorithm that are trained in parallel, and the final combined result is obtained by applying a deterministic algorithm (e.g. average or majority vote) to the set of individual predictions. In contrast, the boosting approach involves classifiers of the same type of algorithm that are trained in sequence, where each misclassification is given a higher weight of importance for the next training round. Hence, classifiers added late to the ensemble will have put more focus on correctly classifying items that were misclassified by early added classifiers. In this way, the total ensemble will be actively steered toward having the potential for correctly classifying a major part of all the items. The final combined result is obtained by taking a linear combination (inversely weighted by the error of each classifier) of the individual results. Another alternative approach is stacking: classifiers of different types of algorithms are trained in parallel, and the final combined result is obtained by applying a deterministic algorithm to the set of individual predictions or by using this set to train a meta-model for making the final decision.

**Ensemble methods for embeddings.** The existing work within this field is mainly focused on: (1) the task of combining word embeddings obtained from different models and/or different text corpora and (2) on enhancing existing word embeddings with specific domain knowledge. The aim of these methods is to leverage the information from all input sources and produce a new set of word embedding vectors with the potential to outperform the original set(s).

**Visual analytic methods**

Previous information visualization (InfoVis) and VA contributions have provided guidelines for designing, implementing, and evaluating interactive solutions that allow users to gain and externalize knowledge about data and complex computational analyses. Such approaches often rely on individual or multiple interactive views designed to facilitate certain user tasks, including comparison, provenance, and guidance, among others. The existing work in InfoVis and VA covers multiple techniques supporting various data types, including texts and graphs/networks, and various applications, including the analyses of social media and scientific publications.

One core idea of VA is the involvement of human analysts in complex computational analyses via interactive user interfaces. The necessity for such human-in-the-loop approaches was recognized decades ago, for instance, by the operations research community, and applied for the tasks associated with combinatorial complexity, human-guided search, and multiple criteria decision making; the survey by Meignan et al. covers this field of interactive optimization methods in operations research. Some of the relevant contributions for this problem originated from the VA community, including the visual optimization techniques for RFID benchmarking by Wu et al., visual multiobjective optimization approaches by Berger et al., and hybrid visual steering technique for simulation ensembles by Matkovic et al. Further review of visual analytic methods for interactive optimization is provided in the recent work by Hakanen et al. Our work shares the idea of involving the human analyst in the interactive search for optimized configurations via VA. But in contrast to the approaches discussed above, our work has a focus on different models and tasks, discussed next.

Recently, the attention of the VA research community has been drawn to the various problems in ML. More specifically, VA approaches have been applied for the results of embedding calculations and also for the purposes of understanding such embeddings better. For instance, Embedding Projector applies dimensionality reduction (DR) methods to display a projection plot for embedding vectors while allowing the users to search and inspect the underlying data items in the original space. Several other existing approaches directly relate to the tasks of text and document visualization as well as visual text analytics, the research subfields concerned with representation of textual data and results of text mining.
approaches, as discussed by Kucher and Kerren\textsuperscript{60} and Liu et al.\textsuperscript{62} The more traditional text visualization approaches mentioned in the respective surveys have been joined during the past years by the approaches that either use word or text embedding features in order to arrive at a suitable visual mapping, or treat the embeddings themselves as the study objects. For example, cite2vec by Berger et al.\textsuperscript{77} focuses on the particular task of interactive citation-driven document collection exploration that is based on joint word-document embeddings. ConceptVector by Park et al.\textsuperscript{25} allows the users to construct lexicon-based concepts for text analysis purposes, which involves interaction with the output of one of the supported word embedding algorithms. Another relevant application for document collection analysis is discussed by Ji et al.,\textsuperscript{26} who make use of a paragraph embedding approach in their VA system. Word Embedding Visual Explorer by Liu et al.\textsuperscript{78} focuses on the investigation of semantic relationships in word embeddings; this approach is supplemented by a case study with a comparison of embeddings produced by two algorithms, Word2Vec\textsuperscript{40} and GloVe.\textsuperscript{79} Liu et al.\textsuperscript{80} discuss Latent Space Cartography, a more general approach for interactive analysis and interpretation of latent spaces and distributed representations, which includes the task of comparing latent space variants (i.e. embeddings), among others. embComp by Heimerl et al.\textsuperscript{81} allows the user to explore word similarity between two different corpora, or for the same corpus embedded by two different methods, by analysis of nearest neighbors within the two embedding spaces. Finally, Parallel Embeddings by Arendt et al.\textsuperscript{82} support exploration and comparison of clusters and cohorts of embedded data over time. While the contributions discussed above provide an important foundation for visual analysis of embeddings, we focus on investigation and comparison of not only \textit{individual} embeddings for the given data sets, but rather \textit{ensembles} of multiple embedding types used for joint decision making. Here, we should acknowledge the existing works discussing VA support for ensemble learning, including EnsembleMatrix by Talbot et al.,\textsuperscript{83} the workflow discussed by Schneider et al.\textsuperscript{84} StackGenVis by Chatzimparmpas et al.,\textsuperscript{85} and ExMatrix by Neto and Paulovich,\textsuperscript{86} for instance. However, these approaches address construction of ML model ensembles for tasks such as classification, while the focus of our proposed approach is on investigating ensembles of \textit{embeddings} for similarity calculations.

\textbf{Text similarity calculations}

While the problem of calculating text similarity can be approached at different scopes and from different perspectives,\textsuperscript{3} there are two major subgroups of methods for calculating the similarity between two text documents that are of interest for our work: word-based and embedding-based. The main advantage of the word-based group is that it is conceptually simple and easy to implement, while the main disadvantage is that pre-processing of the text is usually needed and that semantic similarity is not supported. The main advantage of the embedding-based group is that it can handle semantic similarity and exploit syntactical structure, while the main disadvantage is that they are complex to implement and require substantial training. (The disadvantage of training can however be alleviated by using pre-trained models.) One of the earliest word-based methods is the Jaccard index/similarity\textsuperscript{87} which is calculated by dividing the number of unique common words by the total number of unique words. A more sophisticated, and very popular word based method, is the TF-IDF-method\textsuperscript{88} in which a vector representation of the text is created with a dimension for each unique word in the corpus and with the values calculated as the fraction \textit{[the number of occurrences of the word in the document]/[the number of occurrences of the word in the corpus]}. The similarity score can then be calculated by using the document vectors, for instance by calculating the cosine similarity value.

As already mentioned in this section, most embedding based text similarity methods make use of some form of deep neural networks to compute a vector representation of the text, and a similarity metrics of choice can then be calculated by using the embedding vectors.

\textbf{General methodology}

The main goal of our method is to combine several embeddings (of the same underlying data) with the aim to achieve higher quality in the similarity calculations than what would have been possible if using only a single embedding. As previously stated, we intend to use ensemble methods for this, but since ensemble methods are intended for item classification problems (and not for similarity calculations) we first need to adapt the main ideas to fit our purposes. We first make this adaptation in a general scope (i.e. not only targeting text embeddings), and we then end this section by relating this back to our specific problem of text similarity.

Although similarity calculations using embeddings are not equivalent to item classification problems (since the former is a way to score a relation, and the latter is a way to classify an item), there are some resemblances that we will exploit to make our adaptation. First, we note that similarity calculations over a
set of embeddings typically assign similarity scores to all item pairs in the set, for instance by calculating the cosine similarity value for each pair. Second, we observe that a common way for applications to exploit the similarity scores is to introduce the concept of similarity score threshold(s) that divides the set of all item pairs into (at least) two subsets depending on how similar/dissimilar they are. We may therefore regard a similarity calculation with a single similarity score threshold as a classification of a pair of items into one of the two classes, similar and dissimilar. Loosely speaking, we may view this as creating a new set (where the items to classify are all possible item pairs from the original set) and then performing binary classification on these new items. Based on this reasoning, we therefore conclude that: (1) if it is possible to obtain different embeddings for the same underlying data item (e.g. by using different algorithms or by using the same algorithm with different hyperparameter settings), then (2) it should be possible to combine these different embeddings by ensemble methods to yield a combined result for the similarity calculations. In other words, our proposed methodology is to use several different embeddings to calculate several similarity scores (for a given item pair) and then combine these scores to obtain a final classification. For instance, a straightforward way to combine the results of several embeddings would be to use the concept of bagging (outlined in Section Word and Text Embedding) and to apply a majority voting scheme. This combined result would then, hopefully, have the potential to outperform similarity calculations using any of the contributing embeddings by themselves (see Figure 2 for a generic example). Furthermore, we note that the choice of a specific combiner function and the score threshold values act as hyperparameter settings for the ensemble calculations. Hence, trying to achieve the highest possible quality would be equivalent to searching for the best-performing hyperparameter settings for the ensemble calculations. Finally, we need an equivalent to the training step, during which the ensemble performance is evaluated against sets of already correctly labeled (i.e. correctly classified) training and verification data. The goal for this step is both to obtain a high performance score on the training data and, equally important, to obtain a good generalizability to previously unseen data. As an equivalent to the labeled training set, we introduce the concept of a “ground truth” (GT) set which we use for the performance evaluation (see Section Data Set and Process Steps). It contains a limited number of pairs which have been verified by the analyst to be similar. Hence, finding the best possible hyperparameter settings with regards to the GT set is, in essence, equivalent to the training process. Furthermore, giving the analyst the possibility to provide examples of “true similarity” within the current corpus is a key factor for giving the human more control and achieving “real world” quality. That is, we balance the computational-centered approach of “These items received a high similarity score and therefore we classify them as similar” with the more human-centered and real-world-quality-centered approach of “These items are similar and therefore they should receive a high similarity score”.

We end this section by relating the generic adaptation, described above, back to our specific problem of finding similar articles in a large set of scientific publications (see Figure 3). Keeping in mind that our main goal is to explore the benefits of using several embeddings (instead of only a single one) our process can now be outlined as follows: (1) embed the text from each article in several different ways, (2) create a GT set of article pairs which have been verified as being similar/dissimilar, and (3) find a way to determine which embedding combination, which score thresholds, and which combiner function to use to achieve the highest possible quality for the similarity calculations. (The specific steps of the process will be specified in more detail in the next section.) When this process is finished, we have “trained” an optimal ensemble configuration which will hopefully generalize well to the whole data set, or even to an extended data set of the same type of documents. Furthermore, we expect this ensemble to also be suitable for finding similar article pairs that do not belong to the GT set (which is the main goal since those of the GT set are
already known). In other words, similarity is defined by the article pairs in our GT set, and the process seeks to build a classifier that generalizes this knowledge in the best possible way. This ensemble configuration could then in turn be used for classification within a larger application that seeks to exploit text similarity (for instance, a recommender system for scientists that would be able to recommend similar articles from a chosen publication of interest). This final step is however beyond the scope of this paper since our focus lies specifically on augmenting the quality of the similarity calculations.

Data set and process steps

In this section, we outline the step-by-step process that we apply to our data. This process is in turn partly supported by the EEVO tool, which will be presented in detail in the next section. We employ the IEEE VIS data set89 which contains information of articles published at the IEEE VIS conferences. The choice of the data set is motivated by its quality, visibility in our research community, and our own familiarity with its topic. From this set we have extracted roughly 3000 articles published during the period 1990–2018. We use the text from the article abstracts since: (1) it captures the essence of the article content and (2) similarity/dissimilarity between abstracts is expected to generalize well to similarity/dissimilarity between the corresponding publications.

Step 1: Embed the data

We embed the text of each abstract in five different ways by using USE48 on different parts of the text, as described below. Hence, the variation between the embeddings is due to which part of the underlying text that has been fed into the embedding algorithm. The assumption is that the more semantically similar a pair of abstracts are to each other, the higher the similarity score that is yielded by their corresponding embedding vectors. When developing the tool we also tried embedding the text with the BERT model, but our tests showed that ensembles of USE embeddings only were performing better than mixed USE/BERT ensembles or pure BERT ensembles. However, for clarity, it is important to point out the methodology allows for mixing embeddings from any combination of algorithms for paragraph-sized text embedding (cf. Figure 3). Furthermore, we also want to point out that the focus of our work is on ensembles of embeddings rather than new or improved embedding algorithms themselves; and thus we have not performed any corpus-specific retraining or fine-tuning for any of our used models. Nevertheless, as can be seen in our provided use case, these generically trained models are still able to find pairs with high semantic similarity for our corpus, which is a promising sign for the generalizability of our method. For corpora where a generic model is not performing well enough, the option of domain-specific fine-tuning could therefore be used to further augment the quality.

Type 1 – Embed the first 400 characters of the text and thus capture similar beginnings, but ignore everything else.

Type 2 – Embed the last 400 characters of the text and thus capture similar endings, but ignore everything else.

Type 3 – Concatenation of Type 1 and Type 2. Capture abstracts with similar beginnings and similar endings.

Type 4 – Embed the full text and thus capture overall similarity, but with the risk of being “diluted” in the sense that it becomes more and more challenging to capture “a single meaning” as the text grows longer.
**Type 5** – Embed keyword sentences extracted from the text and thus capture overall similarity, but with the risk of the keywords not being representative.

The rationale for this partitioning of the text is that there is often an implicit structure regarding what is written in the beginning and what is written at the end of an article abstract, and therefore this structure could (at least in theory) be exploited in our calculations. Furthermore, the limit of 400 characters (which is a somewhat arbitrary choice) has been set in relation to the average length of the abstracts, which is just below 1000 characters (or roughly 150 words in about 8–10 sentences). Setting it as a fixed limit instead of a relative limit in percentages alleviates the problem of ambiguity if a much shorter text is being compared to a much longer one.

**Step 2: Calculate the pairwise similarity scores**

We calculate the cosine similarity score (which lies in the interval \([-1, 1]\) and yields higher values the more similar the items are) for all article pairs and for all embedding types. Thus, for each article pair we obtain five different similarity scores (one for each embedding type).

**Step 3: Create a GT set**

To be able to assess and compare the quality of the similarity calculations we need to have some a priori knowledge of article pairs that are similar/dissimilar. We therefore create a small GT set consisting of 58 sampled pairs that are verified to be similar and 58 pairs that are verified to be dissimilar. (Coding was carried out by two authors of this article and they agreed on the labeling of all pairs. Sampling was made by selecting pairs with top/bottom similarity scores from the single-embedding calculations in Step 2 above.) The rationale for using such a small GT set (i.e. 116 sampled pairs out of a total of almost 5 million possible pairs) is that we want to specifically demonstrate that our method can be used for real-world scenarios with very little a priori knowledge. However, the larger the GT set, the more accurate the quality can be calculated and assessed.

**Step 4: Calculate single-embedding ensemble performance**

In this step, the single-embedding ensemble performances are evaluated to obtain a benchmark for the coming multiple-embedding ensemble calculations. In order to make performance ranking of the ensembles possible, we first need to choose a quality metric that is adapted to our specific problem and makes good use of our very limited GT set. Since we are searching for similarity, and we have reason to assume that it is sparse within the set of possible pairs (i.e. we expect most of the article pairs to be dissimilar), it is a reasonable strategy to favor ensembles that (loosely speaking) “Correctly classifies as many as possible of the similar pairs of the GT set, but still keeps the total set of pairs classified as similar small.” The main reason for keeping this set small is the overall sparsity of similarity which implies that, all other things equal, a larger set has a higher risk of having a higher fraction of false positives than a smaller set. Since this, more formally, translates to having a balance between recall and precision we have opted to use the \(F_1\) score as our quality evaluation metric (calculated as per below). One main reason for this is the highly unbalanced nature of the classes (with most pairs being dissimilar and only few being similar) which, for instance, rules out using accuracy as performance metric. To summarize: we are anticipating that high quality performance with regards to the GT set will generalize to the whole set, and that the best performing ensemble will therefore find also other similar pairs that were not included in the GT set (since the ones in the GT set are of course already known). However, we need to keep in mind that we are using a limited GT set and therefore will only be able to calculate an approximation of the true \(F_1\) score. This means that we should not pay too much attention to the specific value of the calculated score (but rather use it for relative ranking) and that there is no strict guarantee that a higher scoring ensemble is actually performing better than a lower scoring one. Nevertheless, we are expecting that an ensemble with substantially higher performance score is indeed performing better than an ensemble with substantially lower score. To alleviate this problem, our suggested solution is to give the analyst a possibility to inspect the yields from the ensembles, so that an assessment of the actual similarity of the underlying texts can be made (see Figure 1(c) and Section Visualization).

For calculating the \(F_1\)-score, we only need the similar pairs of the GT set. (The dissimilar pairs will be used for visualization purposes, see Section Visualization.)

\[
\text{precision} = \frac{\text{sim}_{GT}}{\text{sim}_{TOT}} \\
\text{recall} = \frac{\text{sim}_{GT}}{\text{size}_{GT}} \\
F_1 = \text{harmonic.mean}(\text{precision, recall})
\]

Here \(\text{sim}_{GT}\) denotes the number of items from GT set classified as similar, \(\text{sim}_{TOT}\) denotes the total number of items classified as similar, and \(\text{size}_{GT}\) denotes the
total number of items in the GT similar set. For the single-embedding ensembles, we use the following incremental algorithm to establish the maximal performance:

1. Set the starting similarity score threshold for the embedding to 1 and the granularity of the steps to 0.01.
2. Pairs with scores above or equal to the current threshold are classed as similar, and the others are classed as dissimilar. Check the current classification result against the GT set and calculate the current $F_1$-score.
3. Decrement the similarity score threshold by 0.01 and return to the previous step.
4. When finished (i.e. arrived at threshold $-1$) the maximal performance of the current embedding is the maximum of all the yielded $F_1$-scores.
5. **When max performance has been calculated for all embedding types:** the maximal $F_1$-score of all embedding types is chosen as our benchmark.

This benchmark represents the best possible result that we can achieve with a single-embedding strategy (for our chosen metric) and by calculating it we answer the question "If we could use only one of the embedding types for the task of classifying articles pair on similarity, which one should we choose?" Thus, if we can find a multiple-embedding ensemble that performs better than this, we would have sustained the claim that ensemble methods can work also within the field of embedding-based similarity calculations. As a final note, we would like to point out the fact that the process remains essentially the same even if another quality metric than the $F_1$ score is chosen.

**Step 5: Search for optimal ensemble configuration**

This process step is the main focus of the EEVO tool, and it will be covered in greater detail in the next section. The aim of this step is to determine: (1) which embeddings to combine, (2) which score thresholds to set, and (3) which combiner function to use to achieve the highest possible quality for the similarity calculations.

**Visualization**

In this section, we give an overview of our proposed interactive visual analysis tool, called EEVO, by showing how it can be used for our data set. EEVO is implemented as a web-based tool using D3 and makes use of embeddings computed via the Google Colab platform. As mentioned before, the rationale for providing a visualization for the optimization process is that similarity is an elusive concept which very much lies in the eye of the beholder and is hard to capture by purely computational methods. Furthermore, by visualizing the process, the analyst can get important insights to the inner workings of the similarity calculations and therefore in turn tune the process better to meet the current needs. A generic example of this is shown in Figure 4.

![Figure 4. Using a visual analytic approach, the analyst can combine multiple embeddings computed for the given data and investigate the performance of the ensembles with regards to the specified metrics and the voting scheme. This leads to better-performing embedding ensembles, better hyperparameter settings, and improved knowledge of – and trust for – embedding calculations and ensembles of embeddings. Dashed and dotted edges represent indirect interactions.](image-url)
Since we are loading five different embedding types into the application (see Section Data Set and Process Steps), this means that we will have a total of \(2^5 - 1 = 31\) possible ensemble combinations to evaluate (if counting single embeddings as ensembles as well). The performance of all these are continuously evaluated by the tool so the user does not need to make an active choice on which ensembles to track. Furthermore, EEVO allows for three different combiner functions in form of voting schemes, as specified below. The mental model of this is that: (1) each embedding type first provides a “vote” on whether a specific pair is similar or not and (2) the votes are then combined according to the voting scheme to arrive at a final, unified classification of similar/dissimilar.

Single – A pair is classified as similar if at least one of the embeddings in the ensemble has classified it as similar.

Majority – A pair is classified as similar if more than half of the embeddings in the ensemble have classified it as similar.

Unanimous – A pair is classified as similar if all embeddings have classified it as similar.

As can be seen in Figure 1, the visualization interface of EEVO consists of three views: (A) the Embedding View, (B) the Ensemble Performance View, and (C) the Similarity Assessment View (displayed on demand by clicking a statistics cell in the highscore table). In the Embedding View, the embedding score thresholds can be set, and the corresponding classification statistics can be assessed directly under the sliders. In the Ensemble Performance View, the voting scheme can be selected, and the ensemble performance is displayed on a scatterplot (which can be filtered using the “Filter on score” slider) and in a high-score table. The ensembles are represented by circular multi-colored glyphs in the scatterplot and by multi-colored rectangles in the high-score table. The color encoding (i.e. categorical attributes) corresponds to the colors of the participating embeddings. For instance, an ensemble that consists of the blue, red, and purple embeddings will have these three colors on its glyph and on its high-score rectangle. The glyphs are plotted with regards to the ensembles current precision (y-axis) and recall (x-axis). This means that the higher the \(F_1\)-score, the more to the upper right corner the ensemble glyph will be positioned. To facilitate the comparison of ensembles, the scatterplot is equipped with reclining green bands that indicate the distribution of the \(F_1\)-scores (i.e. any two points on the same border have the same \(F_1\)-score). Furthermore, the currently best performing ensembles will be found, in ascending order, in the highscore table. In the Similarity Assessment View, details of the text pairs classified as similar can be assessed and compared. To facilitate the analysis, common words are highlighted with yellow spans, common authors are highlighted with green spans, and the enclosing rectangles of text pairs that belong to the GT set are color-coded with light green background color.

When EEVO is loaded, all similarity score thresholds are put just above the score of the highest scoring pair for each embedding, and therefore all pairs start out by being classified as dissimilar (see Figure 5). When updating the hyperparameter settings (i.e. the similarity score thresholds or the type of voting scheme), the performance scores of all possible 31 ensembles are continuously updated, so that the analyst does not need to make an active selection on which ensembles to track. The design of EEVO is intended to facilitate the construction of a mental model of the transformation of single-embedding classifications (on the left-hand side) to the resulting ensemble classifications (on the right-hand side). An example of this can be seen in Figure 1, where the number of pairs classified as similar is much higher for some of the individual embeddings (see the numbers below the sliders) than for the combined ensemble results (see the numbers in the high-score table). This is due to the fact that we are using voting scheme “Majority” as combiner function, and it has a filtering effect. Using a different voting scheme would, of course, have yielded a different result.

**Use case**

Assess the different embedding types. Before making any adjustments to the hyperparameter settings, the analyst focuses on the “Embedding Score Distribution” column of the visualization (see Figure 5). Visualizing
the pure score distributions (see Figure 6, left) does not allow for any deeper insights, but by keeping track of the scores of the pairs of the GT set, EEVO is able to provide more interesting details (see Figure 6, right). The analyst notes that there is a general, and encouraging, tendency for all the embeddings to assign higher scores to the similar pairs than to the dissimilar. He/she also observes the differences in how well the embeddings succeed in separating the two sets and the variation in distribution shapes (both regarding the upwards-oriented total distributions and the downwards-oriented GT distributions) and therefore anticipates that this could be exploited by an ensemble combination.

Assess single-embedding performance. By adjusting the similarity score threshold on one slider at a time (leaving the others at their initial values), the analyst can assess the maximum performance score for each embedding one at a time. By doing so (and observing the scatterplot and the high-score table), it is straightforward to verify that the highest scoring single-embedding ensemble is First and last 400 with an $F_1$-score of 0.466 (see Figure 7). As can be seen in the statistics cell of the high-score table, setting the optimal threshold score for this embedding type (which happens to be 0.67) results in to a total of 58 pairs being classified as similar and 27 of those come from the GT set. This is the best similarity classification result that can be achieved (for our chosen quality metric) when using only one single embedding, and the analyst now sets out to find an ensemble configuration which can perform better.

Assess ensemble performance. The analyst now focuses on trying to find an ensemble combination that performs better than the benchmark score found in the task above. As mentioned before, this is in essence a search for optimal hyperparameter settings within the parameter space of the possible voting schemes and the possible similarity score threshold values. To facilitate the search, the tool continuously calculates and displays directive visual guidance, which conveys information on what would be the consequences of decrementing or incrementing each score threshold by 0.01. Aggregated guidance information is displayed as “information scent” above each slider (see Figure 8), while the ensemble-specific guidance is displayed in the high-score table (see Figure 9). The aim of the design is that the analyst would combine the guidance with logical reasoning to augment the chances of finding optimal performing ensembles. Furthermore, some key aspects need to be considered:

Inconsistent guidance: A move that is beneficial for one ensemble may not be so for another, so the sum of all guidance may very well appear inconsistent.

Many vs. few: When in conflict, always choosing the move that is beneficial for the highest number of ensembles may not be the best strategy, especially when close to an optimum. Typically, the final moves will only be beneficial to the very top scoring ensemble(s).

Preserving potential: Existing guidance can be viewed as “potential for improvement” and is calculated under the premises that all other sliders are kept fixed. Therefore, if there is guidance on several sliders, moving one of them might very well “destroy” the potential on the others since the conditions now have changed. Thus, when given a scenario with potential on several sliders, it is prudent to proceed in smaller steps on alternating sliders and with readiness to backtrack. Otherwise, there is a risk of “over-shooting” a branching point where new (and possibly important) guidance would have been discovered.

To start the search, the analyst must first consider which initial settings to use on the sliders and also what voting scheme to use. (Here, two different starting points may result in different guidance, which in turn may lead to different local optima.) For both, the main options are: (i) a random setting or (ii) an educated guess. The analyst chooses to start by setting the slider positions to the values that are optimal for

Figure 6. From the pure score distributions (left), we can only observe a slight difference in the shape and the location of the curves, but no detailed conclusions can be drawn. However, adding the downwards score distribution chart for the GT set (right; green and red for the similar and dissimilar pairs, respectively) clearly reveals that the different embedding types distribute the scores differently and that they are not equally successful in separating the two subsets.
the corresponding single embedding. He/she then tries the different voting schemes and observes that using voting scheme “Single” leads to a top score higher than the benchmark for two ensembles (see Figure 10). Encouraged by this, the analyst now applies logical reasoning and follows the main direction of the guidance and is hereby able to achieve a score of 0.553 (which is an improvement of 19%). The analyst can therefore conclude that using an ensemble of embeddings is a better choice than using only a single embedding.

However, as previously stated, since EEVO makes use of a sampled GT set, the analyst cannot fully rely on the $F_1$ scores (especially if the difference is small), since they are only approximative. Therefore, to differentiate between the top scorers the analyst also needs to assess the yielded classifications by using the Similarity Assessment View (see Figures 1 and 11) to determine which ensemble is truly the best. Furthermore, the achieved score is not the highest possible one, so the analyst must now continue the search by trying different starting points and different voting schemes. By using the tool in this way, we have been able to find a configuration with an ensemble consisting of all five embedding types that achieves a 25% improvement of the $F_1$ score (classifying a total of 51 pairs as similar with 32 of those from the GT set), although this might be a local optimum as well.

We end this section by exploring what the described quality gain corresponds to in more concrete terms and what this could imply for a real-world scenario. Recalling that the best performing single embedding gave a classification of 58 similar pairs (with 27 from the GT set) and that the best ensemble configuration gave a classification of 51 similar pairs (with 32 from the GT set), we see that the numbers are in favor of the ensemble strategy, since it provides 5 more correct
classifications from the GT set as well as an “GT precision” of 32/51 = 0.63 as compared to 27/58 = 0.47. Furthermore, for the classifications that do not belong to the GT set, there are several examples of pairs that show high text similarity and which were not discovered by the best single-embedding strategy (see Figure 11). So, this clearly shows that the ensemble configuration has valuable generalization beyond the “training set/GT set. All in all, we therefore argue that the ensemble strategy is the better choice for any application that seeks to exploit text similarity and where the quality of the classification of similar/dissimilar is an important factor. Taking the field of Science Mapping as an example, and a scenario of combining/ augmenting citation analysis with information on article text similarity, we see that our method would allow for a more correct analysis of the documents, which in turn could lead to richer and more correct insights regarding the underlying set of publications.

**Figure 10.** When using the voting scheme “single,” the analyst observes that setting the sliders to the thresholds optimal for each single embedding yields a score that is higher than the benchmark (i.e. the score of the red single-embedding baseline) for two of the ensembles. Following the guidance leads even to higher scores.

**Figure 11.** A truncated example from the Similarity Assessment View showing 3 text pairs (out of the total of 51) which were classified as similar by the best performing ensemble for our use case. These pairs do not belong to the GT set, and they were not classified as similar by the best performing single embedding. Hence, if they can be verified to be similar they provide clear examples of the benefits of using the ensemble strategy. By close reading of the first pair, the analyst can note that both articles treat the problem of interpreting and understanding the results from clustering algorithms. Although the focus of the article by Cao et al. seems to lie on treemap visualizations while the article by Sacha et al. focuses on self-organizing maps, they still both suggest interactive visualizations for the same type of problem, so there is high similarity of the treated topic. For the second pair, both articles treat the problem of feature tracking in large sets of time-varying data. The article by Sauer et al. introduces a new method for handling object tracking over jumps in time, and the article by Dutta and Shen proposes a distribution-based algorithm that targets the objects motion and its similarity to the target feature. So, also for this text pair there is an obvious level of similarity of the treated topic. For the third and final pair, there is also a high level of topic similarity since both the article by Fujishiro et al. and the article by Santos et al. propose solutions to the problem of helping non-expert users to construct and design effective data visualizations.
Evaluation

Evaluation is an important step for determining if a new interactive visualization approach is successful or not with regard to certain criteria, for instance, usability. We have opted to perform an initial user study which focused on two specific questions: (1) how well our proposed tool supports the user in finding high-performing ensembles and (2) if the design of the tool is straightforward enough to allow even users without expert knowledge of embeddings and ensembles to succeed with the search. The study had a total of 6 participants from the field of computer science with the following profiles:

- Participant 1 – Graduate at Master’s level, general knowledge of ML and visualization.
- Participant 2 – Senior lecturer, expert knowledge of ML and intermediate knowledge of visualization.
- Participant 3 – Master’s student, general knowledge of ML and visualization.
- Participant 4 – Post doc, general knowledge of ML and intermediate knowledge of visualization.
- Participant 5 – PhD student, expert knowledge of ML and intermediate knowledge of visualization.
- Participant 6 – Post doc, intermediate knowledge of ML and general knowledge of visualization.

All sessions were individual with a maximal duration of 1 h. Each participant was given an introduction to the EEVO tool and then spent approximately 20–30 min on the task of trying to find an ensemble which could outperform the best single embedding as discussed in Section Use Case. During the sessions the participants were observed by the test leader, and they were encouraged to verbalize their thoughts and questions out loud, as this enables for a more nuanced final analysis of the evaluation results. Apart from answering any questions from the participants, the test leader remained passive throughout the session and did not provide any active help in solving the task. At the end of the sessions, the participants were asked to give their overall impression of the tool and to fill out an ICE-T evaluation form. (This heuristic evaluation approach focuses on self-reported estimates of visualization value aspects such as its ability to decrease the time necessary for answering questions about the data, to facilitate discovery of insights, etc.)

All of the participants were able to find an ensemble which outperformed the best single embedding. Surprisingly enough, some were even able to find the settings for the best ensemble that we have been able to locate so far during our own tries. With regard to our focus questions, the results hence suggest that our tool can be used for its intended task, and also that it can be used without prior expert knowledge within the fields of embeddings and ensembles. Furthermore, large parts of the oral feedback given at the end of the sessions was consistent and could be condensed to the following:

- The tool was perceived as being user-friendly and having an appropriate design for the intended use.
- The continuous guidance provided good support for solving the task and the chosen visual metaphors were straightforward and easy to interpret.
- For better analysis of situations when the guidance is ambiguous or non-existent, an extended “guidance horizon” (beyond the current limit of one step left/right) would be beneficial.

Other important common comments were: (1) that the tool provided insights to the inner workings of the similarity calculations, (2) that the task could maybe be solved by direct optimization instead, and (3) that similarity is a “fuzzy” concept which adds a level of subjectivity to the calculations (as compared to normal supervised classification). For aggregating the results from the ICE-T questionnaire responses, we performed a numerical translation of the answer options to a scale from 1 to 7, with higher scores indicating better results. Figure 12 provides an overview of the scores, indicating that a majority of the participants have graded EEVO at the higher end of the scale. On the whole, our general assessment of the study setting and the obtained feedback is that the consistent and positive results provide support for the claims that the methodology is working and that our proposed application can be used for the intended task.

Discussion and conclusions

In this paper, we have presented a new method for augmenting the quality of text similarity calculations by using multiple embeddings and ensemble methods. Furthermore, since we believe that the elusive concept of similarity is ill-suited for a purely computational approach, we have also implemented a prototype VA tool, called EEVO, designed for human-in-the-loop visual optimization of the process of finding the best performing ensembles. By applying our methodology to a large set of scientific documents, we have outlined a step-by-step process for how high quality similarity classifications can be obtained in practice for real-world scenarios. We also discussed how this could be used for applications that seek to exploit text similarity. Moreover, we have outlined how the visual optimization process can give the analyst important insights to the inner workings of the similarity calculations and
increase the understanding of complex aspects of machine learning, which in turn will hopefully lead to even better settings and a higher trust for AI applications. In more concrete terms, when facing the challenge of analyzing text similarity over a large set of documents, the method of first providing a few initial samples of “what similarity looks like” within this specific corpus, followed by an interactive training/optimization session using these samples (and with full possibility of verifying the quality of the classifications) provides an intelligible framework which puts the analyst in control of the process. As previously stated, the balance between a computational-centered approach and a more human-centered enables the analyst to acquire vital insights to the strengths and limitations of the constructed classifier and how to best use it within larger applications.

Furthermore, we note that the main ideas of our proposed method, as well as the main architecture of EEVO, generalizes to any type of similarity-preserving embedding technology since no assumption is made on the type of the underlying data. This is because the interface of the method is based on numerical vectors and cosine similarity calculations only, regardless of how the vectors were produced. Therefore, the potential use of our findings goes beyond the scope of our presented use case of textual data, and we believe that our discussion puts forward strong arguments for the potential gains of using our methodology, and that it is suitable for real-world scenarios.

Regarding our EEVO tool, it is important to note that the current implementation does not scale exceptionally well when loading many different embedding types. This is mainly due to the fact that it is very expensive to calculate the guidance, and this is also the main reason for why the guidance has been limited to only one step. With an optimized implementation it would be possible (and very valuable) to extend the guidance to also look “further ahead,” as well as calculating the results of combinations of moves. Similar to Parallel Embeddings, such results could even be compared over time. We also see the need for adding better functionality for comparing the yields and the strategy against the embedding strategy to see which one that would yield the most correct network reconstruction. Once again the embedding strategy proved to be the better choice even though this is a fundamentally different task on a fundamentally different data type. We intend to explore this further as part of our future work.

On the other hand, we are aware of the fact that our method and results do not fall under the category of exact optimization and thus offer no strict guarantee that the results will be worth the effort for all situations. We also need to acknowledge the fact that showing that an ensemble strategy is a better choice for our data, given the embedding types that we present, does not exclude the possibility of the existence of another embedding type (or even some other approach, not involving embeddings at all) that would have been even a better choice. Another relevant limitation of our approach is the fact that text embedding technologies currently only exist for a limited set of languages, and that extrapolation of our results to non-English or multilingual scenarios therefore must be made with caution. Nevertheless, we believe that our discussion puts forward strong arguments for the potential gains of using our methodology, and that it is suitable for real-world scenarios.

Figure 12. The ICE-T scores with the participants sorted on average score. The respective ICE-T categories focus on the ability of the visualization approach to discover insights, decrease time for task solving, convey essence of the data, and generate confidence about the data. Green indicates good results, as opposed to red, and white indicates the neutral value, same as the color map used in the original ICE-T study.
quality of two different ensembles, so that a well founded decision on which one is the best performing can be made – for instance, this could include a compact overview of the compared text pairs, building upon the existing work on comparison \cite{15,19} and text alignment \cite{102} topics in visualization. In summary, we believe that the most promising areas for further development are: (1) the ability to handle more embeddings, (2) the ability to provide more sophisticated guidance, and (3) the ability to compare the results of two different ensembles. Such improvements are a part of our future work. Furthermore, there is a scaling issue also for the scatterplot since it will be cluttered if the number of ensembles is too high, and it is for this reason that we have included the “Filter on score” slider so that the user can select to hide low performing ensembles. Other possible solutions to this problem would be to implement a user selection for the maximal number of ensembles to display, or replacing individual visual items in dense areas with aggregated/cluster representations.

As a final remark, we would once again like to emphasize that the main focus of our proposed method and tool is augmenting the quality of the text similarity calculations since this, as shown in the survey by Wang and Dong,\(^3\) is a generic task with many important applications within several different fields. It is therefore natural that our work may be perceived as not covering a full use case from an analyst perspective. The reason for this is that our contribution is a methodology that is intended to be part of a larger application within such fields as Bibliometrics, Scientometrics, and Science Mapping among others, and as discussed above, our approach could also make use of embedding vectors not necessarily based on text data, for instance, node embeddings for a citation network, or multivariate embeddings. This is also something that we intend to explore as part of our future work.

**Funding**

This work was partially supported through the ELLIIT environment for strategic research in Sweden.

**ORCID iDs**

Daniel Witschard \(\text{https://orcid.org/0000-0001-6150-0787}\)
Andreas Kerren \(\text{https://orcid.org/0000-0002-0519-2537}\)

**References**

1. Islam A and Inkpen D. Semantic text similarity using corpus-based word similarity and string similarity. *ACM Trans Knowl Discov Data* 2008; 2(2): 1–25.
2. Agirre E, Cer D, Diab M, et al. SemEval-2012 Task 6: a pilot on semantic textual similarity. In: *SEM 2012: 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)*, 7–8 June 2012, pp.385–393. ACM.
3. Wang J and Dong Y. Measurement of text similarity: a survey. *Information 2020*; 11(9): 421.
4. Bengio Y, Courville A and Vincent P. Representation learning: a review and new perspectives. *IEEE Trans Pattern Anal Mach Intell* 2013; 35(8): 1798–1828.
5. Khan A, Shah Q, Uddin MI, et al. Sentence embedding based semantic clustering approach for discussion thread summarization. *Complexity* 2020; 2020: 4750871.
6. Kim J, Yoon J, Park E, et al. Patent document clustering with deep embeddings. *Scientometrics* 2020; 123(2): 563–577.
7. Gutiérrez-Gómez L and Delvenne JC. Unsupervised network embeddings with node identity awareness. *Appl Netw Sci* 2019; 4(1): 82.
8. Goyal P and Ferrara E. Graph embedding techniques, applications, and performance: a survey. *Knowl Based Syst* 2018; 151: 78–94.
9. Zhu D, Cui P, Zhang Z, et al. High-order proximity preserved embedding for dynamic networks. *IEEE Trans Knowl Data Eng* 2019; 31(5): 833–852.
10. Almeida F and Xexéo G. Word embeddings: a survey. *CoRR* 2019, abs/1901.09069. DOI: 10.48550/arXiv.1901.09069.
11. Boggust A, Carter B and Satyanarayan A. Embedding comparator: visualizing differences in global structure and local neighborhoods via small multiples. *CoRR* 2019, abs/1912.04853. DOI: 10.1145/3490099.3511122.
12. Toshevska M, Stojanovska F and Kaladjieski J. Comparative analysis of word embeddings for capturing word similarities. In: *Proceedings of the 6th international conference on natural language processing, NATP’20*, April 2020, pp.9–24. AIRCC Publishing Corporation. DOI:10.5121/csit.2020.100402.
13. Kamiński B, Pralat P and Théberge F. An unsupervised framework for comparing graph embeddings. *J Complex Netw* 2021; 8. DOI: 10.1093/comnet/cnz043.
14. Opitz D and Maclin R. Popular ensemble methods: an empirical study. *J Artif Intell Res* 1999; 11: 169–198.
15. Dong X, Yu Z, Cao W, et al. A survey on ensemble learning. *Front Comput Sci* 2020; 14(2): 241–258.
16. Amershi S, Cakmak M, Knox WB, et al. Power to the people: the role of humans in interactive machine learning. *AI Magazine* 2014; 35(4): 105–120.
17. Adadi A and Berrada M. Peeking inside the black-box: a survey on explainable artificial intelligence (XAI). *IEEE Access* 2018; 6: 52138–52160.
18. Gilpin LH, Bau D, Yuan BZ, et al. Explaining explanations: an overview of interpretability of machine learning. In: *Proceedings of the IEEE international conference on data science and advanced analytics*, 1–3 October 2018, pp.80–89. New York, NY: IEEE. DOI: 10.1109/DSSA.2018.00018.
19. Guidotti R, Monreale A, Ruggieri S, et al. A survey of methods for explaining black box models. *ACM Comput Surv* 2019; 51: 1–42.

20. Endert A, Ribarsky W, Turkay C, et al. The state of the art in integrating machine learning into visual analytics. *Comput Graph Forum* 2017; 36(8): 458–486.

21. Sacha D, Sedlmair M, Zhang L, et al. What you see is what you can change: human-centered machine learning by interactive visualization. *Neurocomputing* 2017; 268: 164–175.

22. Sacha D, Kraus M, Keim DA, et al. VIS4ML: an ontology for visual analytics assisted machine learning. *IEEE Trans Vis Comput Graph* 2019; 25(1): 385–395.

23. Chatzimparmpas A, Martins RM, Jusufi I, et al. The taxonomy of interactive optimization methods in operations research. *ACM Trans Interact Intell Syst* 2015; 5(3): 1–43.

24. Smilkov D, Thorat N, Nicholson C, et al. Embedding Projector: Interactive visualization and interpretation of embeddings. In: *Proceedings of the NIPS 2016 workshop on interpretable machine learning for complex systems*. DOI: 10.48550/arXiv.1611.05469.

25. Park D, Kim S, Lee J, et al. ConceptVector: Text visual analytics via interactive lexicon building using word embedding. *IEEE Trans Vis Comput Graph* 2018; 24(1): 361–370.

26. Ji X, Shen HW, Ritter A, et al. Visual exploration of neural document embedding in information retrieval: semantics and feature selection. *IEEE Trans Vis Comput Graph* 2019; 25(6): 2181–2192.

27. Keim DA, Kohlhammer J, Ellis G, et al. (eds) *Mastering the Information Age: Solving pProblems with Visual Analytics*. Goslar: Eurographics Association, 2010.

28. Kerren A and Schreiber F. Toward the role of interaction in visual analytics. In: *Proceedings of the winter simulation conference, WSC ’12*, 9–12 December 2012. New York, NY: IEEE. DOI: 10.1109/WSC.2012.6465208.

29. Meignan D, Knust S, Frayret JM, et al. A review and taxonomy of interactive optimization methods in operations research. *ACM Trans Intell Syst 2015; 5(3): 1–43.

30. Hakanen J, Miettinen K and Matković K. Task-based visual analytics for interactive multiobjective optimization. *J Oper Res Soc* 2021; 72: 2073–2090.

31. Ceneda D, Andrienko N, Andrienko G, et al. Guide me in analysis: A framework for guidance designers. *Comput Graph Forum* 2020; 39(6): 269–288.

32. Sacha D, Stoffel A, Stoffel F, et al. Knowledge generation model for visual analytics. *IEEE Trans Vis Comput Graph* 2014; 20(12): 1604–1613.

33. Andrienko N, Lammarsch T, Andrienko G, et al. Viewing visual analytics as model building. *Comput Graph Forum* 2018; 37(6): 275–299.

34. Aggarwal CC. *Machine Learning for Text*. New York, NY: Springer, 2018.

35. Bengio Y, Ducharme R, Vincent P, et al. A neural probabilistic language model. *Journal of Machine Learning Research* 2003; 3: 1137–1155.

36. Collobert R, Weston J, Bottou L, et al. Natural language processing (almost) from scratch. *Journal of Machine Learning Research* 2011; 12: 2493–2537.

37. Turian J, Ratinov L and Bengio Y. Word representations: A simple and general method for semi-supervised learning. In: *Proceedings of the 48th annual meeting of the association for computational linguistics, ACL’10*, 2010, pp.384–394. ACL.

38. Meinecke C, Wrisley D and Janicke S. Automated alignment of medieval text versions based on word embeddings. In: *Proceedings of the Leipzig symposium on visualization in applications, LEVI’19*, 2019. DOI:10.31219/osf.io/ta9h3y.

39. Meinecke C, Wrisley D and Janicke S. Explaining semi-supervised text alignment through visualization. *IEEE Trans Vis Comput Graph* 2021. DOI: 10.1109/TVCG .2021.3105899.

40. Mikolov T, Sutskever I, Chen K, et al. Distributed representations of words and phrases and their compositionality. In: *Proceedings of the 26th international conference on neural information processing systems—Volume 2, NIPS’13*, 2013, pp.3111–3119. Red Hook, NY: Curran Associates Inc.

41. Devlin J, Chang MW, Lee K, et al. BERT: Pre-training of deep bidirectional transformers for language understanding. In: *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, Volume 1 (long and short papers), NAACL ’19*, 2019, pp.4171–4186. ACL. DOI: 10.18653/v1/n19-1423.

42. Le Q and Mikolov T. Distributed representations of sentences and documents. In: *Proceedings of the 31st international conference on machine learning, ICML’14*, 2014, pp.1188–1196. PMLR.

43. Mitchell J and Lapata M. Composition in distributional models of semantics. *Cogn Sci* 2010; 34(8): 1388–1429.

44. Sevastjanova R, Kalouli AL, Beck C, et al. Explaining contextualization in language models using visual analytics. In: *Proceedings of the 59th annual meeting of the association for computational linguistics and the 11th international joint conference on natural language processing (volume 1: long papers), ACL-IJCNLP’21*, 2021, pp.464–476. ACL. DOI: 10.18653/v1/2021.acl-long.39.

45. Socher R, Perelygin A, Wu J, et al. Recursive deep models for semantic compositionality over a sentiment treebank. In: *Proceedings of the 2013 conference on empirical methods in natural language processing, EMNLP’13*, 2013, pp.1631–1642. ACL.

46. Kalchbrenner N, Grefenstette E and Blunsom P. A convolutional neural network for modelling sentences. In: *Proceedings of the 52nd annual meeting of the association submitted for double-blind review (revision 2) for computational linguistics (volume 1: long papers), ACL’14*, 2014, pp.655–665. ACL. DOI: 10.3115/v1/P14-1062.

47. Kiros R, Zhu Y, Salakhutdinov R, et al. Skip-thought vectors. In: *Proceedings of the 28th international conference on neural information processing systems—Volume 2, NIPS’15*, December 2015, pp.3294–3302. Cambridge, MA: MIT Press.
50. Breiman L. Arcing classifiers (with discussion and a rejoinder by the author). *Ann Stat* 1998; 26: 801–849.

51. Wolpert DH. Stacked generalization. *Neural Netw* 1992; 5(2): 241–259.

52. Speer R and Chin J. An ensemble method to produce high-quality word embeddings. arXiv:1604.01692 2016.

53. Muromagi A, Sirts K and Laur S. Linear ensembles of word embedding models. In: *Proceedings of the 21st Nordic conference on computational linguistics, NoDaLiDa’17*, pp.96–104. ACM.

54. Xia C, He T, Wan J, et al. Ensemble methods for word embedding model based on judicial text. In: *Web information systems and applications: 16th international conference, WISA 2019*, Qindao, China, 20–22 September 2019, vol. 11817, pp.309–318. Cham: Springer. DOI: 10.1007/ 978-3-030-30952-7_31.

55. Roberts JC. State of the art: Coordinated & multiple views in exploratory visualization. In: *Proceedings of the fifth international conference on coordinated and multiple views in exploratory visualization*, CMV 2007, 2 July 2007, pp.61–71. New York, NY: IEEE. DOI: 10.1109/CMV.2007.20.

56. Heer J and Shneiderman B. Interactive dynamics for visual analysis: A taxonomy of tools that support the fluent and flexible use of visualizations. *Queue* 2012; 10(2): 30–55.

57. Brehmer M and Munzner T. A multi-level typology of abstract visualization tasks. *IEEE Trans Vis Comput Graph* 2013; 19(12): 2376–2385.

58. Gleicher M, Albers D, Walker R, et al. Visual comparison for information visualization. *Inf Vis* 2011; 10(4): 289–309.

59. Ragan ED, Endert A, Sanyal J, et al. Characterizing provenance in visualization and data analysis: an organizational framework of provenance types and purposes. *IEEE Trans Vis Comput Graph* 2016; 22(1): 31–40.

60. Kucher K and Kerren A. Text visualization techniques: Taxonomy, visual survey, and community insights. In: *Proceedings of the IEEE pacific visualization symposium, PacificVis’15*, pp.117–121. New York, NY: IEEE. DOI: 10.1109/PACIFICVIS.2015.7156366.

61. Jänicke S, Franzini G, Cheema MF, et al. On close and distant reading in digital humanities: a survey and future challenges. In: *Proceedings of the EG/GVT/CG conference on visualization—STARS, EuroVis’15*, pp.83–103. The Eurographics Association. DOI:10.2312/eurovisstar.20151113.

62. Liu S, Wang X, Collins C, et al. Bridging text visualization and mining: a task-driven survey. *IEEE Trans Vis Comput Graph* 2019; 25(7): 2482–2504.

63. Kerren A, Purchase H and Ward MO (eds). *Multivariate Network Visualization*. Cham: Springer, 2014.

64. Nobre C, Meyer M, Streit M, et al. The state of the art in visualizing multivariate networks. *Comput Graph Forum* 2019; 38(3): 807–832.

65. Chen S, Lin L and Yuan X. Social media visual analytics. *Comput Graph Forum* 2017; 36(3): 563–587.

66. Federico P, Heimerl F, Koch S, et al. A survey on visual approaches for analyzing scientific literature and patents. *IEEE Trans Vis Comput Graph* 2017; 23(9): 2179–2198.

67. Fisher ML. Interactive optimization. *Ann Oper Res* 1985; 5(3): 539–556.

68. Pirkul H, Gupta R and Rolland E. VisOpt: a visual interactive optimization tool for P-median problems. *Decis Support Syst* 1999; 26(3): 209–223.

69. Klau GW, Lesh N, Marks J, et al. Human-guided search. *J Heuristics* 2010; 16(3): 289–310.

70. Eskelinen P, Miettinen K, Klamroth K, et al. Pareto navigator for interactive nonlinear multiobjective optimization. *Spectrum* 2010; 32(1): 211–227.

71. Wu Y, Chung KK, Qu H, et al. Interactive visual optimization and analysis for RFID benchmarking. *IEEE Trans Vis Comput Graph* 2009; 15(6): 1335–1342.

72. Berger W and Piringer H. Interactive visual analysis of multiobjective optimizations. In: *Proceedings of the IEEE symposium on visual analytics science and technology, VAST’10*, pp. 215–216. New York, NY: IEEE. DOI: 10.1109/VAST.2010.5651694.

73. Berger W, Piringer H, Filzmoser P, et al. Uncertainty-aware exploration of continuous parameter spaces using multivariate prediction. *Comput Graph Forum* 2011; 30(3): 911–920.

74. Matković K, Gračanin D, Splechtna R, et al. Visual analytics for complex engineering systems: Hybrid visual steering of simulation ensembles. *IEEE Trans Vis Comput Graph* 2014; 20(12): 1803–1812.

75. Stolper CD, Perer A and Gotz D. Progressive visual analytics: user-driven visual exploration of in-progress analytics. *IEEE Trans Vis Comput Graph* 2014; 20(12): 1653–1662.

76. Sedlmair M, Heinzl C, Bruckner S, et al. Visual parameter space exploration: A conceptual framework. *IEEE Trans Vis Comput Graph* 2014; 20(12): 2161–2170.

77. Berger M, McDonough K and Seversky LM. Cite2vec: citation-driven document exploration via word embeddings. *IEEE Trans Vis Comput Graph* 2017; 23(1): 691–700.

78. Liu S, Bremer PT, Thigarajan JJ, et al. Visual exploration of semantic relationships in neural word embeddings. *IEEE Trans Vis Comput Graph* 2018; 24(1): 553–562.

79. Pennington J, Socher R and Manning C. GloVe: Global vectors for word representation. In: *Proceedings of the conference on empirical methods in natural language processing, EMNLP’14*, pp.1532–1543. ACM. DOI: 10.3115/v1/D14-1162.

80. Liu Y, Jun E, Li Q, et al. Latent space cartography: visual analysis of vector space embeddings. *Comput Graph Forum* 2019; 38(3): 67–78.

81. Heimerl F, Kralj C, Moller T, et al. embComp: visual interactive comparison of vector embeddings. *IEEE Trans Vis Comput Graph* 2022; 28: 2953–2969.
82. Arendt DL, Nur N, Huang Z, et al. Parallel embeddings: a visualization technique for contrasting learned representations. In: Proceedings of the 25th international conference on intelligent user interfaces, IUI’20, pp.259–274. ACM. DOI:10.1145/3377325.3377514.

83. Talbot J, Lee B, Kapoor A, et al. EnsembleMatrix: interactive visualization to support machine learning with multiple classifiers. In: Proceedings of the SIGCHI conference on human factors in computing systems, CHI’09, pp.1283–1292. ACM. DOI: 10.1145/1518701.1518895.

84. Schneider B, Jackle D, Stoffel F, et al. Integrating data and model space in ensemble learning by visual analytics. IEEE Trans Big Data 2021; 7(3): 483–496.

85. Chatzimparmpas A, Martins RM, Kucher K, et al. StackGenVis: alignment of data, algorithms, and models for stacking ensemble learning using performance metrics. IEEE Trans Vis Comput Graph 2021; 27(2): 1547–1557.

86. Neto MP and Paulovich FV. Explainable matrix - visualization for global and local interpretability of random forest classification ensembles. IEEE Trans Vis Comput Graph 2021; 27(2): 1427–1437.

87. Manning CD and Schutze H. Foundations of Statistical Natural Language Processing. Cambridge, MA: MIT Press, 1999.

88. Salton G and Buckley C. Term-weighting approaches in automatic text retrieval. Inf Process Manag 1988; 24(5): 513–523.

89. Isenberg T, Isenberg P, Chen J, et al. A systematic review on the practice of evaluating visualization. IEEE Trans Vis Comput Graph 2013; 19(12): 2818–2827.

90. Ruder S, Vulic I and Søgaard A. A survey of cross-lingual word embedding models. J Artif Intell Res 2019; 65: 569–631.

91. Gleicher M. Considerations for visualizing comparison. IEEE Trans Vis Comput Graph 2018; 24(1): 413–423.

92. Cao N, Gotz D, Sun J, et al. DICON: Interactive visual analysis of multidimensional clusters. IEEE Trans Vis Comput Graph 2011; 17(12): 2581–2590.