computation

Article

Greedy Texts Similarity Mapping

Aliya Jangabylova [1,*], Alexander Krassovitskiy [1,*], Rustam Mussabayev [1,*] and Irina Ualiyeva [2,*]

[1] Institute of Information and Computational Technologies, Pushkin Str., 125, Almaty 050010, Kazakhstan
[2] Faculty of Information Technology, Al-Farabi Kazakh National University, 71 Al-Farabi Ave., Almaty 050040, Kazakhstan
* Correspondence: ajangabylova@gmail.com (A.J.); akrassovitskiy@gmail.com or a.krassovitskiy@ipic.kz (A.K.); rmusab@gmail.com or rustam@iict.kz (R.M.); i.ualiyeva@gmail.com or ualiyeva.irina@kaznu.kz (I.U.)

Abstract: The documents similarity metric is a substantial tool applied in areas such as determining topic in relation to documents, plagiarism detection, or problems necessary to capture the semantic, syntactic, or structural similarity of texts. Evaluated results of the similarity measure depend on the types of word represented and the problem statement and can be time-consuming. In this paper, we present a problem-independent algorithm of the similarity metric *greedy texts similarity mapping* (GTSM), which is computationally efficient to be applied for large datasets with any preferred word vectorization models. GTSM maps words in two texts based on a decision rule that evaluates word similarity and their importance to the texts. We compare it with the well-known word mover's distance (WMD) algorithm in the k-nearest neighbors text classification problem and find that it leads to similar or better results. In the correlation evaluation task of similarity measures with human-judged scores, we demonstrate its higher correlation scores in comparison with WMD and sentence mover's similarity (SMS) and show that GTSM is a decent alternative for both word-level and sentence-level tasks.

Keywords: text similarity; word mover distance; k-nearest neighbors; word embedding; text classification

check for updates

Citation: Jangabylova, A.; Krassovitskiy, A.; Mussabayev, R.; Ualiyeva, I. Greedy Texts Similarity Mapping. *Computation* **2022**, *10*, 200. https://doi.org/10.3390/computation10110200

Academic Editor: Xinwei Cao

Received: 6 October 2022
Accepted: 4 November 2022
Published: 8 November 2022

1. Introduction

Similarity is the primary element that shows the relation between two objects, numerical or categorical, and gives an interpreted result for human perception and machine processing. The concept of similarity measure is widely used in many scientific and interdisciplinary fields such as decision making, cognitive science, natural language processing (NLP), recommender systems, and many other areas [1–3]. In NLP, the similarity metric is used to capture the semantic relation of words in a given context that shows to what extent words express the same meaning.

Semantic similarity measures the strength of interactions between words and captures any direct or indirect semantic relations, Harispe et al. [4]. Textual semantic similarity is utilized to identify common characteristics between cases, such as word–word, word–document, document–document, or query–document. Semantic similarity is essentially helpful for topic modeling analysis and semantic information retrieval systems where it identifies the most optimal match of two objects. Generally speaking, different approaches to semantic measurements should be used, depending on the type of data, task formulation, and algorithm structure in which it is applied [5,6]. Given the words that can be represented in vector forms, one can use a distance measure to decide if their numerical representations are similar based on the decision rule: "the words are semantically similar if the distance between them is less than some threshold".

The success of any similarity measure metrics relies on the vector representation of words. Zhang et al. [7] show that bag of words (BOW) counts the frequency of a word appearing in a sentence or document without considering the order, syntactic or semantic of words, where the dimension of a vector space is the number of all unique words in the corpus. Term frequency-inverse document frequency (TF-IDF) works in the same manner, but instead of giving more importance to common words, it reduces its weight as the word appears more often in other documents, as stated in Ramos et al. [8]. Two different word2vec methods proposed in Mikolov et al. [9], continuous bag-of-words and skip-gram model, obtain weights of shallow neural network based on proximity probability in its surrounding context to use it as vectors for each word with the dimension of a predefined number of neurons. To bypass the issue of uninterpreted word vectors, Pennington et al. [10] introduced global vectors for word representation (GloVe) which is based on statistical calculations of corpus' word occurrences and generates numerical representation of words, regardless of their location in the sentence or possibility of homonym pair. In 2019, Devlin et al. [11] introduced a bidirectional encoder representations from transformer (BERT) which is a context-dependent embedding where the same word *apple* will have different vectors in *juicy apple* and *apple juice*. These vectors are coordinates of words in vector space, and by calculating the distance between them, the similarity of words can be found.

Kusner et al. [12] proposed a new metric, denoted as word mover distance (WMD), which measures the distance between texts in a semantic way based on the word2vec model. The intuition behind the WMD method is that it minimizes the cumulative traveling distance between sets of words in two documents, D and T.

However, WMD relies on the minimization of the total traveling distance between all document's words, including low relevancy ones. It implies limitations of WMD to certain embedding types and has low relevancy for many NLP tasks. In [13], the author stresses the need to incorporate word importance regarding their syntactic level of connectivities into WMD.

In our work, we develop such a text similarity metric, denoted as greedy texts similarity mapping (GTSM), that combines the semantics of word embeddings with word weights and assembles by meta-heuristics into a reasonably practical estimator. Hence, the estimator can be used with preferable vectorization models, word weights, and distance metrics. We construct a heuristic that is simple enough, has a small number of parameters, and can still handle the semantic features of words.

The benchmarked algorithm which will be used to compare the performance of GTSM is WMD, sentence mover similarity (SMS), which is identical to WMD with averaged sentence embeddings introduced by Clark et al. [14] and the cosine similarity metric. The choice of WMD method is justified by its low retrieval error rate as stated by Kusner et al. [12] and the error rate estimation by k-NN document classification task that outperformed other similarity measurements, such as BOW, TF-IDF, latent semantic indexing, latent Dirichlet allocation, and others. The full list and their descriptions can be found in Kusner et al. [12].

2. Related Work

In 2015, Kusner et al. [12] published a paper on WMD that inflamed the NLP society with its impressive results on similarity metrics. The main engine on which WMD is constructed is earth mover distance (EMD), also known as Wasserstein distance, that minimizes the total work needed to change one probability distribution into the form of another. EMD was mainly applied in cases such as content-based image retrieval, Rubner et al. [15], histogram comparisons, Ling and Okada [16], phishing web pages, Fu et al. [17], and recently in vector-based text similarity, Kusner et al. [12].

Kusner et al. [12] assign each word i in document D the weight $D_i = count(i)/|D|$, where $|D|$ is the word count of document D.

The semantic similarity for the WMD metric between a pair of words can be captured through Euclidean distance denoted as a "travel cost" between word i and word j:

$$c(i,j) = ||\mathbf{x}_i - \mathbf{x}_j|| \tag{1}$$

where $||\cdot||$ is L_2 norm in Euclidean space.

The transportation problem below states that from each word in document D it finds the closest word in document T and aggregates to a total distance that D needs to travel to exactly match T.

$$\min_{\mathbf{T}_{ij}\geq 0} \sum_{i=1}^{n}\sum_{j=1}^{n}\mathbf{T}_{ij}\, c(i,j) \tag{2}$$

$$\sum_{j=1}^{n}\mathbf{T}_{ij} = D_i \tag{3}$$

$$\sum_{i=1}^{n}\mathbf{T}_{ij} = T_j \tag{4}$$

where $\mathbf{T}_{ij} \geq 0$ denotes what 'amount' of word i in the document D transfers to word j in the document T, and n is the size of vocabulary. Equations (3) and (4) ensure that the amounts transformed from word i to words in T is D_i and that the total amount of words i in D transformed into j is T_j.

Although the results of WMD are very promising, this method is computationally inefficient with average complexity time $O(p^3 \log p)$, where p is the number of unique words in documents. As Kusner et al. [12] states, it is "prohibitive" to use this method for high-dimensional datasets. As an alternative, to speed up the process, they also proposed the word centroid method or relaxing boundaries WMD with the tradeoff in the error rate reduction.

The sentence mover's similarity (SMS) introduced by Clark et al. [14] circumvented complexity issues by averaging word embeddings so that the similarity evaluation performs only on sentence-levels which significantly saves computational time. That is, *sentence to documents* is equivalent to *word to sentences*. The weight of each sentence embedding is estimated by the number of words in the document $D_i = |i|/|D|$, where in this case i is the sentence in document D, and $|i|$ is the word count in the corresponding sentence. As was suggested in Kilickaya et al. [18], we transform WMD to word mover similarity (WMS) as $\text{WMS}(D, D') = \exp(\text{-WMD}(D, D'))$.

3. Greedy Texts Similarity Mapping

3.1. Vector Representation of Words

3.1.1. Co-Occurrence

The co-occurrence matrix, also known as the word–context matrix, was presented in Leydesdorff and Vaughan [19]. It preserves the idea that similar words tend to have similar contexts. The co-occurrence matrix of words counts the number of times each word pair appears inside the fixed context window of a particular size and is capable of capturing the semantic relation between words. It might not be the best choice to use a co-occurrence matrix for a very small corpus since it is more accurate if more samples are given. Compression techniques can be applied for extremely large datasets to present every word in a smaller dimension, as shown in Levy et al. [20].

We employ a co-occurrence matrix as a base for the calculation of a pairwise similarity matrix. This distance matrix of size $n \times n$ will be used as a vector representation for each word.

3.1.2. Weights of Words

We assign to every term in a text a unique weight that reflects its importance to the text. There are various ways to calculate a word's contribution level to the text's general meaning. We have considered several ways of calculating weights.

3.1.3. Centroid

Centroid-based word weightings find the coordinate in a vector space such that it is the average of all word embeddings in a document D as was presented in Radev et al. [21]. Leaning on this centroid point C, $||C - v_i||$ is a distance from the vector representation of word i to the centroid; the terms having a larger distance are assigned to a smaller weight value.

$$C = \sum_i \frac{v_i}{|D|}$$
$$W_i = 1 - \frac{||C - v_i||}{\sum_j ||C - v_j||}$$

3.1.4. TF-IDF Weights

It is a simple algorithm yet beneficial at effectively embracing content that is undervalued. TF-IDF is broadly used in search engines since it allows extracting unique keywords among huge datasets of documents. *Term frequency* TF_{id} measures how frequently the term is met in a document. *Inverse document frequency*, IDF, is a measure of how important the term is regarding the whole corpus. The weight of term i in document d is Salton and Buckley [22]:

$$W_{id} = \mathrm{TF}_{id} \cdot \mathrm{IDF}_i \tag{5}$$

where $\mathrm{TF}_{id} = |i|/|d|$, $\mathrm{IDF}_i = \log(N/n_i)$, N is the total number of documents in the corpus, n_i is the number of documents containing the word i, and $|d|$ is the length of document d.

According to Beel et al. [23], "TF-IDF was the most frequently applied weighting scheme". TF-IDF's output can be used for further semantic analysis as was shown in our document similarity metric.

3.1.5. Latent Dirichlet Analysis (LDA)

LDA, presented in Blei et al. [24], is a topic modelling algorithm that generates topics and evaluates the topic distribution for every document in the corpus. It assigns a topic to every word and finds the combination of topics that best describes a certain document.

Let us say given a number of topics such that $\{t_1, \ldots, t_n\}$, we obtain the probability distribution of every document to all topics and distribution of words to topics (obtained by LDA). Given a document d, we choose a topic with a maximum probability such that $t^* = \mathrm{argmax}_t\, \mathbf{P}(t|d)$, and to obtain a weight for the word w in document d, we select the value that belongs to the topic t^* in the word to topics probability distribution.

$$W_{wd} = \mathbf{P}(w|t^*) \tag{6}$$

3.2. Algorithm

The basic idea of GTSM is to estimate the mapping of words in two documents and normalize it by filtering out essential word relations and taking into account corresponding word importance. It consists of the two following parts: truncated mapping and normalization part. The truncated mapping (nominator) estimates all possible relations between word pairs and is truncated by introduced rigging parameters. For the normalization part (denominator), we develop a general form of the "ideal" relation between any two texts/documents. Hence, if the calculated truncated mapping is substantially more significant than the ideal mapping, then the obtained similarity score is significant and $>> 1$. Appendix A.4 illustrates the GTSM procedure.

3.2.1. Ideal Relations

First, we demonstrate how to calculate the normalization factor of the similarity score. Let us say we are given two documents D and T where D = $\{mother, lasagna\}$ and T = $\{dinner, quantum, cheese\}$. The pairwise similarities of words in D to words in T are known. From every word in D, we choose the maximum similarity with the word in T and vice versa, as shown in Equations (7) and (8) and Figure 1:

$$m_{D_i} = \underset{w_{T_j}}{\arg\max}\, S(w_{D_i}, w_{T_j}) \text{ for all } i = 1, \ldots, |D| \tag{7}$$

$$m_{T_j} = \underset{w_{D_i}}{\arg\max}\, S(w_{D_i}, w_{T_j}) \text{ for all } j = 1, \ldots, |T| \tag{8}$$

where m_D is a distribution of maximum similarity values for words in D, and m_T is a distribution for words in T.

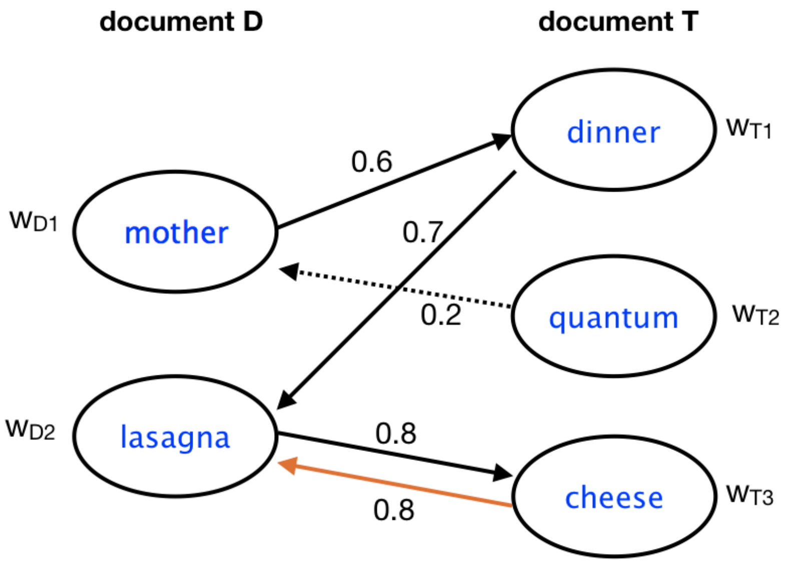

Figure 1. A graph with the significant relations between documents D and T. For a detailed explanation refer to Appendix A.4.

We obtained distributions of maximum similarity value for each word in two documents, $m_D = (0.6, 0.8)$ and $m_T = (0.7, 0.2, 0.8)$, keeping the records of indices for what words they joined $id_D = [1, 3]$ and $id_T = [2, 1, 2]$. We assume that these distributions are normal and want to find the words whose similarity values are out of the bounds of one sigma $\mu - \sigma$, so that we eliminate 'outliers' (see Equation (9)). Attention is paid to the lower bounds, the values that are less than mean, since a higher similarity means a stronger relation.

$$\begin{aligned} \text{decision}_{D_i} &= m_{D_i} - \text{mean}(m_D) + \text{std}(m_D) \\ \text{decision}_{T_j} &= m_{T_j} - \text{mean}(m_T) + \text{std}(m_T) \end{aligned} \tag{9}$$

where $i = 1, \ldots, |D|$, and $j = 1, \ldots, |T|$

Corresponding values in (9) allow us to decide whether the corresponding words would contribute to the similarity score or not (see Equation (10)). Back to our example, using the above equation we obtain $\text{decision}_D = (0, 0.2)$ and $\text{decision}_T = (0.477, -0.123, 0.577)$. The values in decision_D are within 1σ and, thus, accept all words in D. There is a negative value in decision_T, and that is why we reject the corresponding word *quantum* and conclude that it is not significant enough to contribute to the true similarity value between D and T. Note that as more words are considered, the higher the similarity.

$$g_{D_i} = \begin{cases} 1, & \text{if decision}_{D_i} \geq 0 \\ 0, & \text{otherwise.} \end{cases} \tag{10}$$

3.2.2. Implementation Procedure

Consider a set of documents D and T are from the set. Then, w_{D_i} and w_{T_j} represent the word $i \in D$ and word $j \in T$. The GTSM metric calculates the similarity between documents in the following way:

1. Obtain vector representation of words.
2. The similarity of two words is $S(w_{D_i}, w_{T_j}) = S_{D_i T_j} = 1 - d(w_{D_i}, w_{T_j})$, provided d is normalized. Calculate a matrix of pairwise similarities.
3. Estimate word weights using one of the preferred methods (e.g., TF-IDF, etc.).
4. Calculate a truncated mapping (nominator in Equation (12)).
5. Set up a threshold, ϵ, by looking at which we decide whether a certain pair of words has a significant connection. Set the decision function of whether a pair of words is considered to be significant or not as:

$$f(w_{D_i}, w_{T_j}) = \begin{cases} 1, & \text{if } S_{D_i T_j} \geq \epsilon \\ 0, & \text{otherwise.} \end{cases} \tag{11}$$

6. Evaluate Equation (12), where we conclude whether a pair of words coming from two documents have a significant relation or not decided by $f(w_{D_i}, w_{T_j})$. Then, considering their connection strength and weights, we normalize by "standard" true relation of all possible pairs from two documents which already knows what words contribute to the similarity value.

$$GTSM(D,T) = \frac{\sum_{i=1}^{|D|} \sum_{j=1}^{|T|} f(w_{D_i}, w_{T_j}) k_{D_i} k_{T_j} (P + S_{D_i T_j})}{\sum_{i=1}^{|D|} g_{D_i} k_{D_i} k_{T_i^*} S_{D_i T_i^*} + \sum_{j=1}^{|T|} g_{T_j} k_{T_j} k_{D_j^*} S_{D_j^* T_j}} \tag{12}$$

where k_{D_i} and k_{T_j} are weights of word i in document D and word j in document T, subscripts $_{T_i^*}$ and $_{D_j^*}$ denote indices i in id_D and j in id_T, and P is a regularization parameter.

In addition, note that if significant relations obtained by Equation (10) are dual, such as in Figure 1 where *lasagna* is connected to *cheese* and vice versa, then we ignore one of these connections while calculating the denominator in $GTSM(D,T)$.

Assuming that all significant relations between documents are obtained, the intuition of Step 6 is that the words with larger weights are much stronger representatives of their documents, and those with smaller distances strengthen their relations. On the other hand, if one of the words is essential and the second has a small contribution to its document, then the distance should be larger since their significant alliance is supposed to reflect similar relative importance. So, the multiplication of weights to the similarity value regularizes the alliance likeness. The computation complexity of GTSM is $O(n^2)$, as pairwise word similarities have to be calculated.

4. Evaluation

To examine the performance of GTSM, we set up two different tasks and tested them in the context of (i) k-NN classification problem by comparing the F1 scores of every method and (ii) correlation between metric benchmarks to learn if the GTSM's results are more significant. For both tasks, we used TF-IDF word weight in the GTSM metric. All hyperparameters of GTSM were optimized with the Bayesian optimization Python library provided by Martinez-Cantin [25].

4.1. Word Vectors

A shortage of data is one of the main challenges in NLP. Thus, it is popular and convenient to use embeddings that are already pre-trained on millions or billions of data and allow us to extract useful vector information. However, it underestimates the power of the regular co-occurrence matrix learned on a given dataset. We compared four word representation baselines:

4.1.1. Co-Occurrence

Co-occurrence (cooc) is a statistical matrix that is trained on the entire corpus. First, we calculated the BOW and multiplied the output of its transpose to count the frequency of a word in every document.

4.1.2. Word2vec

Word2vec is a three-layer neural network-based model that, through n-grams of words, predicts every word. We used a model (https://code.google.com/archive/p/word2vec/ accessed on 1 October 2022) trained on Google News dataset with 300-dimensional vectors. Words that are not contained in a pre-trained word2vec model are dropped.

4.1.3. GloVe

Glove builds a co-occurrence matrix by estimating the probability of a word co-occurring with the others. Since it is trained on a gigantic matrix, it factorizes a matrix to a lower dimensional representation. The implementation procedure is the same as for Word2vec with a pre-trained GloVe model (https://nlp.stanford.edu/projects/glove/ accessed on 3 October 2022).

4.1.4. BERT

BERT masks 15% of words in the document, and the Transformer generates a prediction on erased words based on known words. It is a bidirectional language representation, so it keeps the order of words. To extract word embeddings, we sum the vectors from the last four layers, as was suggested in Alammar [26], and for texts/documents that have more than 512 tokens (it is a limit), we keep the first and last 256 tokens as it is assumed that they are most informative.

In Appendix A.1, you can find the combination of baselines we used to compare against our algorithm.

4.2. Classification Task

4.2.1. Datasets

For the classification task, we evaluated all methods on supervised datasets: BBCSPORT [27] is a sports article from 2004 to 2005 labeled by categories {*athletics, cricket, football, rugby, tennis*}, CLASSIC is a set of extracts from academic papers labeled by names of publishers, AMAZON—a set of labeled reviews that are categorized into five product groups {*arts and crafts, brands, games, hobbies, party supplies*}, and RECIPE is recipe description documents classified by region of origin.

All four datasets were obtained from Kusner et al. [12] with ready preprocessed steps that are described in more detail (https://github.com/mkusner/wmd accessed on 6 October 2022). The data characteristics are shown in Appendix A.2.

4.2.2. Test Results

Document similarity metrics are applicable in various tasks, and our choice falls on *k*-NN to compare with the results presented in Kusner et al. [12]. To assess the effectiveness of GTSM as an evaluation metric, we compared *k*-NN results with the aforementioned word representation vectors and metric baselines.

It is fair to say that GTSM requires some additional tests to fine-tune its parameters. However, we consider it to be an advantage as it allows our model to be more flexible and adapt to certain datasets. Through tests and observations, we share some insights on its effect on metric performance.

4.2.3. Epsilon ϵ Parameter

Figure 2 depicts how the change in the ϵ parameter affects the trends of the F1 score, where all other parameters are fixed. For *word2vec*, if we assume the similarity threshold to be less than 0.5, then it adds a lot of noise with false connections and leads to the deterioration at the final similarity value. The peak for CLASSIC is ϵ from 0.50 to 0.52 where if less or more, then the F1 score decreases, while BBC and AMAZON rapidly increase at $\epsilon > 0.60$. We observe interesting trends when the vectorization method is a *co-occurrence matrix*. There is a steep increase for BBC at $\epsilon = 0.96$ and moderate increase when $\epsilon > 0.80$ for the rest data.

We see that for any dataset, tuning an ϵ parameter plays a key role in the GTSM metric's success, and considering its fast computational performance, it is quite easy to find the threshold that suits some specific dataset best.

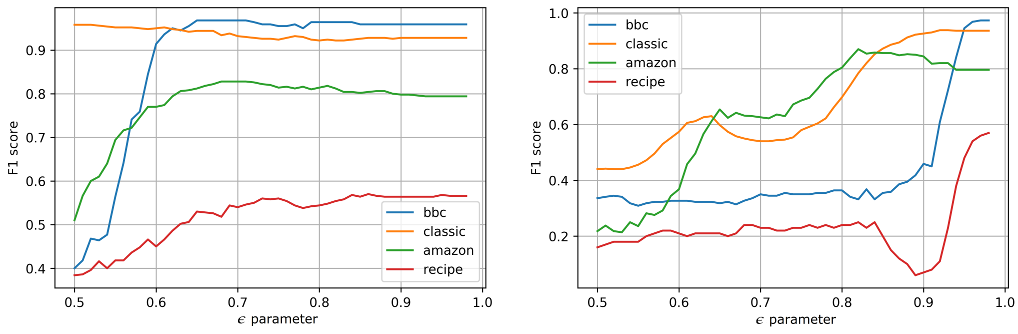

Figure 2. Representation of GTSM method's ϵ parameter behaviour: (**left**) for word2vec; (**right**) for co-ocurrence matrix.

4.2.4. *P* Parameter

Parameter *P* is a free parameter that can increase the similarity measure slightly to some extent. It is important to note that the denominator that describes "ideal" relations for every pair of documents will have some errors since it is just a generalization and does not consider every possible case. Thus, the parameter *P* compensates for this miscarriage. The improvement resulted from sharpening word-to-word similarities and, hence, can be regarded as the normalization of this pairwise metric. Figure 3 depicts how the change in *P* parameter affects the trends of the F1 score.

For *word2vec*, there are peaks at range $[-0.7, -0.5]$ followed by steady lines with some improvements from 1% to 5% in comparison with *P* = 0, whereas for *co-occurrence*-based vectors, the increase in the F1 score is only for AMAZON.

P parameter is helpful when there is a need to obtain as accurate results as possible, but if it is not critically important, then *P* can be set to 0 by default.

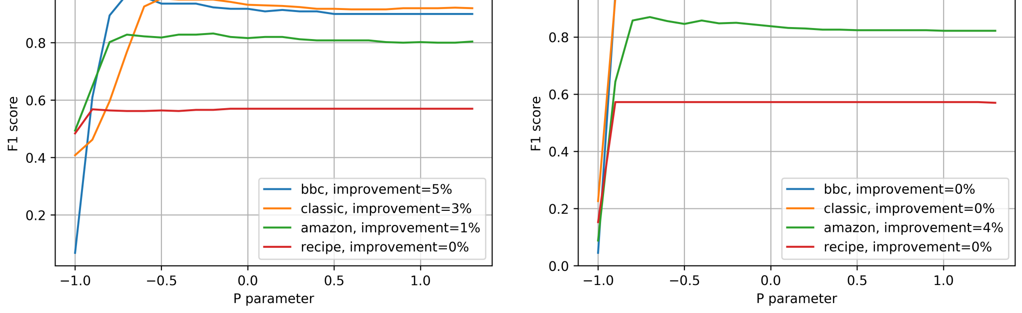

Figure 3. Representation of the GTSM method's *P* parameter behaviour: (**left**) for word2vec; (**right**) for co-ocurrence matrix.

4.2.5. Sigma σ Parameter

We construct a generalized ideal case of word relations (Equation (7)) by deciding if the maximum similarity values of pairs are within 1σ. By putting rigid boundaries in 1σ, we allow some minor error to appear that can be offset by narrowing or expansion of these boundaries depending on a particular dataset and its peculiar properties. However, tuning its boundaries might be insignificant and *without a need to obtain a highest possible result it can be set up automatically to* 1σ.

Table 1 represents the k-NN results of the aforementioned methods on four document classification datasets. Surprisingly, the co-occurrence matrix performs almost as well as word2vec when combined with WMD and outperforms GTSM. The word2vec is a probabilistic model that tries to maximize the probability of words occurring together that shows a resemblance with the co-occurrence matrix. While the word2vec is trained on global information, the co-occurrence represents local information based on the counts of co-occurred word pairs which is the first order co-occurrence relation defined in Zhuang et al. [28]. Thus, the estimation based only on the given corpus might be the reason why co-occurrence vectors have higher results than other embeddings.

Table 1. F1 score for k-NN compared to baseline methods.

	BBC	**Amazon**	**Classic**	**Recipe**
GTSM-cooc	**98.2**	**92.6**	**96.9**	61.0
GTSM-w2v	96.8	91.4	**97.0**	**61.6**
GTSM-BERT	96.4	87.1	95.6	57.3
Cos-BERT	90.5	88.7	94.8	46.9
WMD-w2v	95.4	**92.6**	**97.2**	57.0
WMD-cooc	95.0	91.8	96.4	54.8

There are various ways of extracting the vector representations, so there are several ways to examine the best approach for the particular dataset. Moreover, even a small corpus such as BBC is enough for co-occurrence-based vectors to capture the semantic relationship of words. Regarding the BERT, we intended to show that it can be used not only as a metric itself (see cos+BERT), but rather it can be improved with the GTSM similarity metric as shown on BBC, CLASSIC and RECIPE datasets, even though it is not always the case as for AMAZON. The possible reason why BERT shows slightly lower results than other word representation vectors is that it was not fine-tuned to a specific dataset.

In these results, we withdraw that WMD as the slowest metric to compute, Kusner et al. [12], and rather focus on its quality evaluation metric. In general, we see that WMD performs better than GTSM with a difference of 0.2% on CLASSIC, the same as for AMAZON. On RECIPE, it shows a 1.6% worse result and a considerable 2.8% difference for BBC with respect to GTSM.

4.2.6. Weights

We evaluated three approaches that allow extracting the importance of words in a document described in more detail in Section 3 and examined how each affects the metric performance.

Figure 4 represents the comparison of TF-IDF, LDA, and centroid-based weights. TF-IDF received the highest value in the F1 score with the number of neighbors $k = 7$, and as k increases, its score slowly reduces. Approximately the same results are for centroid-based weights, even though this method averages all words in the document and uses this measure as a centroid to evaluate every word. LDA-based weights have slightly lower evaluation results. Note that we choose only the highest value of the topic assigned to the document, whereas there is a combination of the best topics that can be taken into account.

We note that the proposed GTSM algorithm is general and is not limited to only the suggested weighting methods.

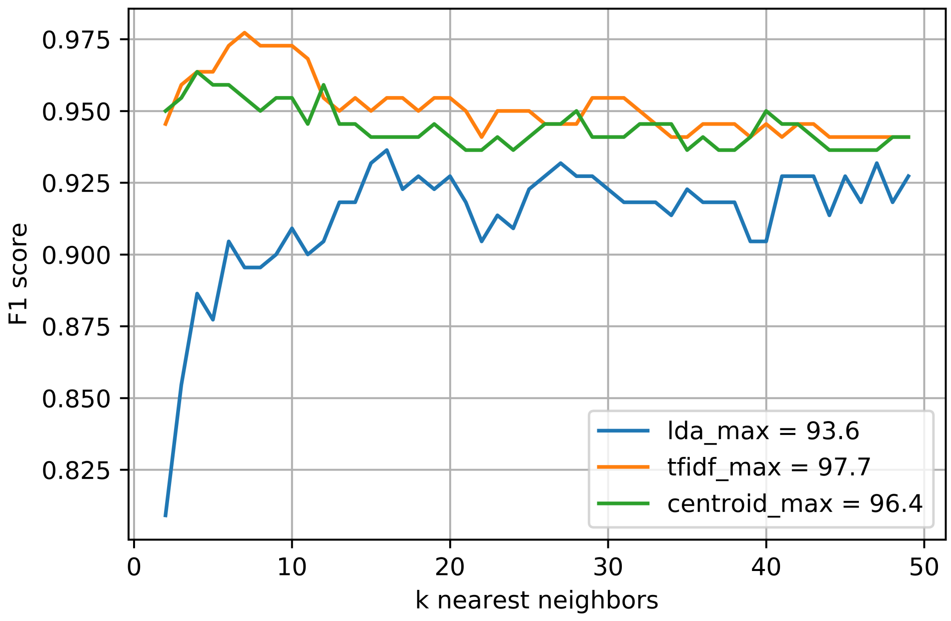

Figure 4. Comparison of weighting methods' F1 scores in *k*-NN task for BBC dataset.

4.3. Correlation Task

For the correlation evaluation task, we considered a collection of student essays at the Hewlett Foundation's Automated Student Assessment Prize that was uploaded on Kaggle (https://www.kaggle.com/c/asap-sas/data accessed on 6 October 2022). We used a set of student responses to Question #3 from the exam where graders assigned responses with scores from 0 to 3. All evaluators had a reference essay to compare and evaluate student-authored essays. The reference essay is available in Appendix A.3.

The basic text preprocessing steps were to remove (i) punctuation, (ii) stopwords, (iii) numeric strings, and (iv) tokens with length = 1, (v) normalize words to lowercase, (vi) apply lemmatization, and (vi) tokenize. The dataset statistics are presented in Appendix A.2.

Table 2 presents the Spearman correlation, Myers and Sirois [29], results of listed benchmarks with the human evaluated scores. The Spearman correlation test assesses the strength of the monotonic relationship between two variables. In order to make it strong, we tested all possible combinations of mentioned similarity metrics in Appendix A.1 and word representations. Note that the purpose of these comparisons was not only to see what vector models are better suited but rather to evaluate how they behave in various similarity metrics. The results are quite interesting, and it can be concluded that not all combinations of vectors and metrics have a sense of fitting together.

However, in a general view, the best results are shown by the GTSM metric that can be suited to all given vector models. The highest correlation with human-graded scores is provided by the cosine-BERT-based model, where the cosine metric averages all word vectors to obtain sentence vectors and then calculates their cosine similarity; its advantage of 0.9% over GTSM is not substantial. At the same time, the cosine metric has the worst results with the co-occurrence-based vectors. In the real case, instead of using a co-occurrence matrix with the cosine metric, it might be better to decompose the word–context matrix into factors. The word–context matrix does not fit with WMS and SMS well. In Clark et al. [14], it is reported that SMS with GloVe has better performance than WMS with GloVe, while we obtained the opposite. The dataset size and preprocessing steps could play a crucial role.

The WMS minimizes the distances between words in two texts; that is, it goes through all the word pairs calculating Euclidean distances. Every vector in word2vec is 300-dimensional, in BERT is 768-dimensional, and in co-occurrence there are 6177 dimensions; let us remember that in high-dimensional space, points become uniformly distant, and the ratio of nearest or farthest points to a given target approaches one Aggarwal et al. [30]; thus, the distinction might be meaningless.

The WMS and SMS are based on the minimization of pure distance measurements, while the GTSM approach is trying to find a suitable mapping by adjusting its algorithm to a particular dataset.

Table 2. Spearman correlation of metrics with human evaluations for different vector representations.

	Cooc	**W2V**	**GloVe**	**BERT**
GTSM	**0.658**	**0.640**	**0.648**	**0.701**
WMS	0.206	0.607	0.559	0.483
SMS	0.338	0.431	0.427	0.160
Cosine	0.097	0.633	0.557	**0.709**

We have also obtained p-values of the Williams test to estimate the significance of vector models in the GTSM metric, where the null hypothesis states that the vectors' scores in the row are not significant over the column vectors. According to these results, there was no significant difference in the choice of vector model.

5. Conclusions

We propose a text similarity measure, GTSM that can be applied in various tasks of different corpus sizes using a preferred vectorization method. The proposed algorithm intends to capture the essential word relations between two texts and evaluate their similarity according to the weightings of words. The attractiveness of this algorithm is its simplicity of implementation, the flexibility of tuning for specific data, relatively cheap computations (average complexity time is $O(n^2)$ regarding the number of unique words in documents), the ability of easy parallelization, and its acceptable F1 score. It is also self-sufficient and does not need additional resources or computationally expensive steps (such as in the preliminary calculation of word2vec or BERT), so it is not bonded to languages with pre-trained vector models and can handle low-resource languages. We showed that a co-occurrence matrix could be a good alternative for other vectorization models such as BERT. As an advantage of the presented algorithm, it is possible to state that it relies on vectorized word representations and, assuming such exist, no further concerns regarding linguistic properties of the analyzed dataset.

The algorithm provides a unified way of using word embeddings and word weighting to find a relation between two bags of words. It is worth noting that this method is flexible enough to be used as a feature generating mechanism, e.g., in graph analysis. Nevertheless, the feature representation and the number of classes strongly impact the classification quality, as it is seen on the RECIPE dataset.

The GTSM algorithm has plenty of room for further expansions: e.g., regarding the selection of advanced heuristic models for computing "ideal" relations or applying alternative vectorizations and weightings (as discussed in Ibrahim and Landa-Silva [31] for TF-IDF) in order to obtain more efficient training and lower error rate results.

Author Contributions: Conceptualization, A.J., A.K., R.M. and I.U.; methodology, A.J., A.K., R.M. and I.U.; software, A.J., A.K., R.M. and I.U.; validation, A.J., A.K., R.M. and I.U.; formal analysis, A.J., A.K., R.M. and I.U.; investigation, A.J., A.K., R.M. and I.U.; resources, A.J., A.K., R.M. and I.U.; data curation, A.J., A.K., R.M. and I.U.; writing—original draft preparation, A.J., A.K., R.M. and I.U.; writing—review and editing, A.J., A.K., R.M. and I.U.; visualization, A.J., A.K., R.M. and I.U.; supervision, A.J., A.K., R.M. and I.U.; project administration, A.J., A.K., R.M. and I.U.; funding acquisition, A.J., A.K., R.M. and I.U. All authors have read and agreed to the published version of the manuscript.

Funding: The work was funded by the Committee of Science of Ministry of Education and Science of the Republic of Kazakhstan under the grants AP08856034, AP09259324, AP09058174.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: All datasets and vectorization models used in experiments are freely available at Kusner et al. [12] and https://github.com/mkusner/wmd (accessed on 6 October 2022).

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

GTSM	Greedy texts similarity mapping
WMD	Word Mover's Distance
SMS	Sentence Mover's Similarity
cooc	Co-occurrence (matrix)

Appendix A

Appendix A.1. List of used vectorization models and similarity metrics.

Table A1. Baselines.

Task	Metric	Vector Form
Classification	GTSM	Cooc/W2V/BERT
	WMD	Cooc/W2V/BERT
	Cosine	BERT
Correlation	GTSM	Cooc/W2V/GloVe/ BERT
	WMD	Cooc/W2V/GloVe/ BERT
	SMS	Cooc/W2V/GloVe/ BERT
	Cosine	Cooc/W2V/GloVe/ BERT

Appendix A.2. Datasets and Results

Table A2. Classification task dataset statistics.

	Train Size	Test Size	Vocab Size	Avg Document Size	Num Classes
BBCSPORT	517	220	10,076	116.5	5
CLASSIC	4965	2128	18,070	38.7	4
AMAZON	5600	2400	30,183	197	4
RECIPE	3059	1311	5202	48.3	15

Table A3. Correlation task dataset statistics.

	Essays
Corpus size	1727
Vocabulary size	6177
Avg # tokens per sentence	8.6
Avg # tokens per document	52.4
Avg # sentences per document	5.9

Table A4. Parameters of GTSM obtained by Bayesian Optimization.

	ϵ	P	$x \times \sigma$	**Result**
Cooc				
BBC	0.9803	−0.6	0.8226	98.2
AMAZON	0.8637	−0.6	0.9195	92.6
CLASSIC	0.92	−0.7	0.8954	96.9
RECIPE	0.9999	0.0	0.9801	60.9
w2v				
BBC	0.9049	−0.7	1.438	96.8
AMAZON	0.7024	−0.7	0.7757	91.4
CLASSIC	0.4858	−0.5	0.8994	97.0
RECIPE	0.9994	−0.6	2	61.6
BERT				
BBC	0.8059	−0.5	0.1462	96.4
AMAZON	0.81	−0.5	0.6235	87.1
CLASSIC	0.7981	−0.8	0.6997	95.6
RECIPE	0.84	0	0.99	57.3

Appendix A.3. Reference Essay

The setting seems to be as formidable an opponent as the actual workout. It seems as if everything is against the cyclist, including nature. As the day progresses, and the cyclist's journey continues, the setting becomes harsher and harsher. After passing the first "town", the "sun was beginning to beat down." In need of water, all a cruel pump gives him is "a tarlike substance." His sufferings continue, increasingly pummeled by his surroundings and his thirst for water. If dehydration was not enough, the flat terrain gave way to "rolling hills", which would only punish his legs more. Reaching possible salvation, his hopes are crushed when the "Welch's Grape Juice Factory" turns out to be abandoned. All these events are enough to destroy anyone's spirit. The cyclist almost gives up hope to accept certain death. He has become ferociously beaten by his very surroundings. It appears as if he is fated to die alone in the blistering heat. Although he hangs his head in despair, he still continues on the path of disappointment. In a twist of fate, he encounters a thriving store where he halts and drinks. Finally encountering his salvation, this particular setting brings new hope and relief to the cyclist who has finally survives his trek through nature.

Appendix A.4. GTSM illustration

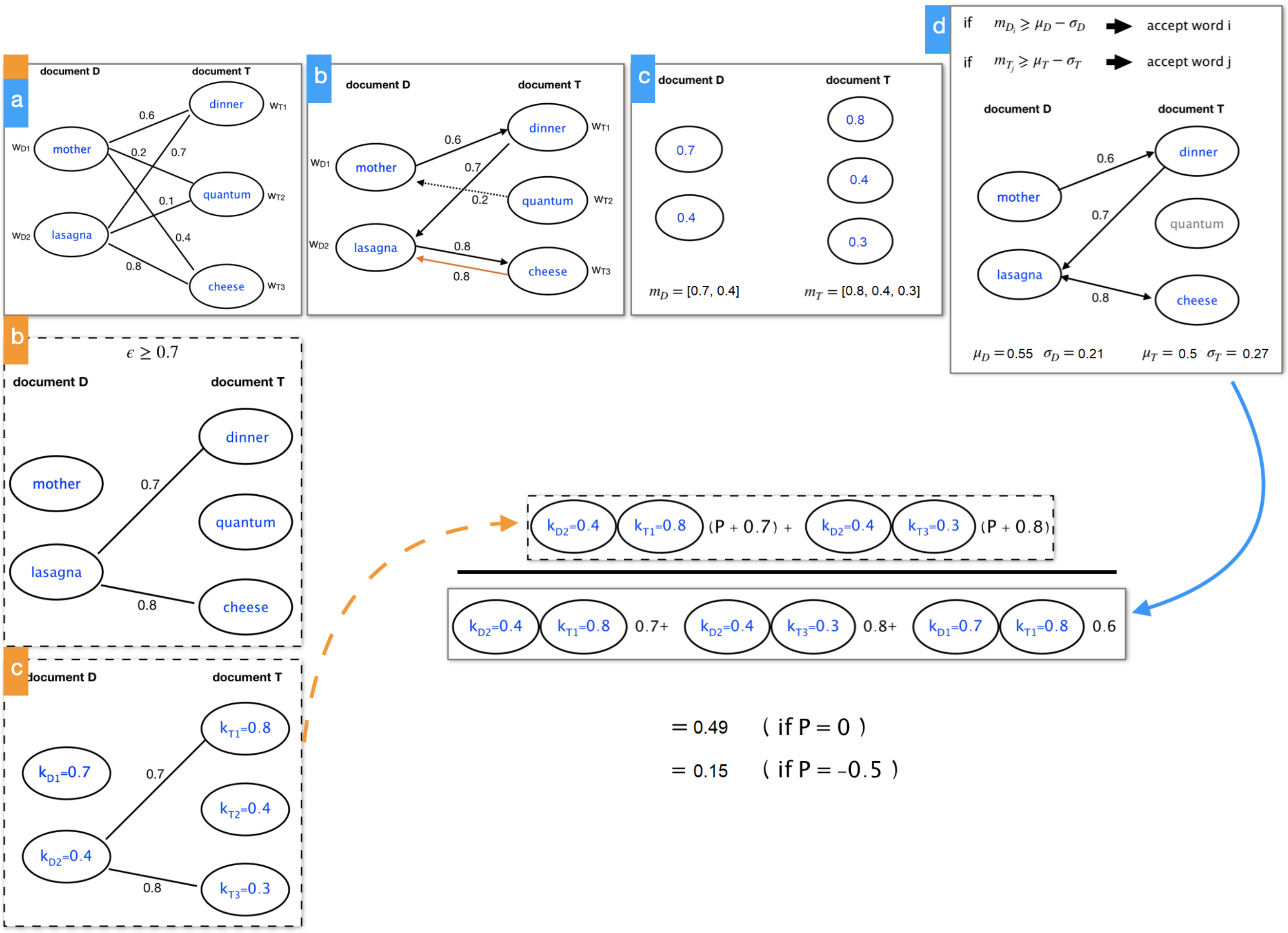
a
document D
document T
mother
lasagna
dinner
quantum
cheese
b
c
d
$\epsilon \geq 0.7$
m_D = [0.7, 0.4]
m_T = [0.8, 0.4, 0.3]
if $m_{D_i} \geq \mu_D - \sigma_D$ accept word i
if $m_{T_j} \geq \mu_T - \sigma_T$ accept word j
μ_D = 0.55 σ_D = 0.21
μ_T = 0.5 σ_T = 0.27
k_{D2}=0.4 k_{T1}=0.8 (P + 0.7) + k_{D2}=0.4 k_{T3}=0.3 (P + 0.8)
k_{D2}=0.4 k_{T1}=0.8 0.7+ k_{D2}=0.4 k_{T3}=0.3 0.8+ k_{D1}=0.7 k_{T1}=0.8 0.6
= 0.49 (if P = 0)
= 0.15 (if P = -0.5)

Figure A1. Illustrative representation of the GTSM algorithm. It is assumed that we already performed vectorization of words at this stage, obtained weights, and calculated similarities between words. ***Orange dashed line*** (denominator): (**a**) relations between all words in two documents, (**b**) set up a threshold ϵ and ignore all relations that are less than ϵ, (**c**) consider weights of words. Then multiply similarities to the corresponding weights of words. ***Blue solid line*** (numerator): (**a**) relations between all words in two documents, (**b**) leave only those similarities that are maximum in respect to each word where dashed line denotes to ignore the link and orange line to consider that there is a repeated relation, (**c**) record them in lists m_D and m_T, (**d**) calculate mean and standard deviation for each list, estimate values of which words are out 1σ and ignore those words and their relations. We note, that in case ϵ is selected such that the majority of relations between w_{D_1} and w_{T_1} are included (e.g., $\epsilon = 0.1$), then GTSM algorithm takes into account even non-essential word relations. The evaluated similarity score can be greater than one. In order to normalize the obtained value to (0, 1)-range, it is possible to use elementary mapping, such as $f(s) = 1 - \exp(-s)$. However, since obtained similarity score is used in k-NN (or other evaluation methods) and only relative values matter(i.e., whether two documents are more similar than the others), such normalization can be omitted. From this example it is clear that ϵ parameter (as well as P parameter) of the GTSM can be used as rigging parameters for specific problem domains.

References

1. Veisi, H.; Golchinpour, M.; Salehi, M.; Gharavi, E. Multi-level text document similarity estimation and its application for plagiarism detection. *Iran J. Comput. Sci.* **2022**, *5*, 143–155. [CrossRef]
2. Arabi, H.; Akbari, M. Improving plagiarism detection in text document using hybrid weighted similarity. *Expert Syst. Appl.* **2022**, *207*, 118034. [CrossRef]
3. Mishra, S.; Panda, S.K. Asymmetrically weighted cosine similarity measure for recommendation systems. In *Proceedings of the Advances in Distributed Computing and Machine Learning*; Rout, R.R., Ghosh, S.K., Jana, P.K., Tripathy, A.K., Sahoo, J.P., Li, K.C., Eds.; Springer Nature Singapore: Singapore, 2022; pp. 489–500.
4. Harispe, S.; Ranwez, S.; Janaqi, S.; Montmain, J. Semantic similarity from natural language and ontology analysis. In *Synthesis Lectures on Human Language Technologies*; Morgan & Claypool: San Rafael, CA, USA, 2015; Volume 8, p. 433. [CrossRef]
5. Wang, J.; Dong, Y. Measurement of Text Similarity: A Survey. *Information* **2020**, *11*, 421. [CrossRef]
6. Mudgil, P.; Gupta, P.; Mathur, I.; Joshi, N. A novel similarity measure for context-based search engine. In Proceedings of the International Conference on Innovative Computing and Communications, New Delhi, India, 17–18 February 2023; Gupta, D., Khanna, A., Bhattacharyya, S., Hassanien, A.E., Anand, S., Jaiswal, A., Eds.; Springer Nature Singapore: Singapore, 2023; pp. 791–808.
7. Zhang, Y.; Jin, R.; Zhou, Z.H. Understanding bag-of-words model: A statistical framework. *Int. J. Mach. Learn. Cybern.* **2010**, *1*, 43–52. [CrossRef]
8. Ramos, J. Using tf-idf to determine word relevance in document queries. In Proceedings of the First Instructional Conference on Machine Learning, Piscataway, NJ, USA, 21–24 August 2003; Volume 242, pp. 133–142. Available online: https://www.researchgate.net/file.PostFileLoader.html?id=587340a5dc332da8fc3aaae3&assetKey=AS%3A448525403201536%401483948197307 (accessed on 1 October 2022).
9. Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. Efficient estimation of word representations in vector space. *arXiv* **2013**, arXiv:1301.3781.
10. Pennington, J.; Socher, R.; Manning, C. Glove: Global vectors for word representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, 26–28 October 2014; Association for Computational Linguistics: Doha, Qatar, 2014; pp. 1532–1543. [CrossRef]
11. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. 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), Minneapolis, MN, USA, 2–7 June 2019; Association for Computational Linguistics: Minneapolis, MN, USA, 2019; pp. 4171–4186. [CrossRef]
12. Kusner, M.; Sun, Y.; Kolkin, N.; Weinberger, K. From word embedding to document distances. In Proceedings of the 32nd International Conference on Machine Learning, Lille, France, 6–11 July 2015; pp. 957–966.
13. Wei, C.; Wang, B.; Kuo, C.C.J. SynWMD: Syntax-aware Word Mover's Distance for Sentence Similarity Evaluation. *arXiv* **2022**, arXiv:2206.10029. [CrossRef]
14. Clark, E.; Celikyilmaz, A.; Smith, N.A. Sentence mover's similarity: Automatic evaluation for multi-sentence texts. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, Florence, Italy, 28 July–2 August 2019; Association for Computational Linguistics: Florence, Italy, 2019; pp. 2748–2760. [CrossRef]
15. Rubner, Y.; Tomasi, C.; Guibas, L.J. The earth mover's distance as a metric for image retrieval. *Int. J. Comput. Vis.* **2000**, *40*, 99–121. [CrossRef]
16. Ling, H.; Okada, K. An efficient earth mover's distance algorithm for robust histogram comparison. *IEEE Trans. Pattern Anal. Mach. Intell.* **2007**, *29*, 840–853. [CrossRef] [PubMed]
17. Fu, A.Y.; Wenyin, L.; Deng, X. Detecting phishing web pages with visual similarity assessment based on earth mover's distance (EMD). *IEEE Trans. Dependable Secur. Comput.* **2006**, *3*, 301–311. [CrossRef]
18. Kilickaya, M.; Erdem, A.; Ikizler-Cinbis, N.; Erdem, E. Re-evaluating automatic metrics for image captioning. *arXiv* **2016**, arXiv:1612.07600.
19. Leydesdorff, L.; Vaughan, L. Co-occurrence matrices and their applications in information science: Extending ACA to the Web environment. *J. Am. Soc. Inf. Sci. Technol.* **2006**, *57*, 1616–1628. [CrossRef]
20. Levy, O.; Goldberg, Y.; Dagan, I. Improving Distributional Similarity with Lessons Learned from Word Embeddings. *Trans. Assoc. Comput. Linguist.* **2015**, *3*, 211–225. [CrossRef]
21. Radev, D.R.; Jing, H.; Styś, M.; Tam, D. Centroid-based summarization of multiple documents. *Inf. Process. Manag.* **2004**, *40*, 919–938. [CrossRef]
22. Salton, G.; Buckley, C. Term-weighting approaches in automatic text retrieval. *Inf. Process. Manag.* **1988**, *24*, 513–523. [CrossRef]
23. Beel, J.; Gipp, B.; Langer, S.; Breitinger, C. Paper recommender systems: A literature survey. *Int. J. Digit. Libr.* **2016**, *17*, 305–338. [CrossRef]
24. Blei, D.M.; Ng, A.Y.; Jordan, M.I. Latent dirichlet allocation. *J. Mach. Learn. Res.* **2003**, *3*, 993–1022.
25. Martinez-Cantin, R. Bayesopt: A bayesian optimization library for nonlinear optimization, experimental design and bandits. *J. Mach. Learn. Res.* **2014**, *15*, 3735–3739.
26. Alammar, J. The Illustrated Transformer. 2019. Available online: http://jalammar.github.io/illustrated-bert/ (accessed on 1 October 2022).

27. Greene, D.; Cunningham, P. Practical solutions to the problem of diagonal dominance in kernel document clustering. In Proceedings of the 23rd International Conference on Machine Learning (ICML'06), Baltimore, MD, USA, 25–29 June 2006; ACM Press: New York, NY, USA, 2006; pp. 377–384.
28. Zhuang, Y.; Xie, J.; Zheng, Y.; Zhu, X. Quantifying context overlap for training word embeddings. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, 31 October–4 November 2018; pp. 587–593.
29. Myers, L.; Sirois, M.J. Spearman correlation coefficients, differences between. *Encycl. Stat. Sci.* **2004**, *12*, ess5050.
30. Aggarwal, C.C.; Hinneburg, A.; Keim, D.A. On the surprising behavior of distance metrics in high dimensional space. In Proceedings of the International Conference on Database Theory, London, UK, 4–6 January 2001; Springer: Berlin/Heidelberg, Germany, 2001; pp. 420–434.
31. Ibrahim, O.A.; Landa-Silva, D. A new weighting scheme and discriminative approach for information retrieval in static and dynamic document collections. In Proceedings of the 2014 14th UK Workshop on Computational Intelligence (UKCI), Bradford, UK, 8–10 September 2014; pp. 1–8.