A Hybrid Approach for Sentence Similarity: Combining Semantic and Structural Similarity Metrics

Wout Haakman , Pradeep Murukannaiah
TU Delft

Abstract

Predicting similarity between sentence pairs is essential for applications such as recommender systems and plagiarism detection. There have been several categories of approaches for predicting sentence similarity. This paper combines approaches from two categories, semantic and structural similarity, to find a hybrid approach that aligns more closely with how humans determine similarity.

Extensive background research is conducted to understand the scope of the problem, to be able to understand human psychology, as well as the advantages and disadvantages of already existing approaches. Based on the insights from the background research, we propose a novel hybrid approach, combining semantic and structural similarity metrics. The proposed approach is evaluated on the STSS-131 and MSRP datasets and compared with other common approaches and SentenceBERT, a deep learning algorithm. The proposed approach does not perform as well as SentenceBERT but makes up for this by being more explainable and outperforming all other traditional machine learning approaches in its accuracy in predicting sentence similarity. This paper also provides a critical conclusion with recommendations for further improvements.

1 Introduction

Since the beginning of the computerized and highly connected world, there has been a big growth of information. And with that came the need to structure all this information. This was achieved partly due to text similarity, using algorithms [1] like [1] [2] [3], which is to determine how closely related two pieces of text are. Text similarity could be used for applications like web search, navigating big data-sets, and categorization of documents. However, one of the significant downsides of using text similarity approaches is that they rely on the occurrence and count of particular words. This does not work for short texts and sentences, as these do not contain a high degree of words which results in a less precise similarity prediction. Insight in the semantics and structure of a sentence gets lost, which tell a lot more about short text and sentences than counting particular words in the sentence.

These algorithms are called sentence similarity, or short text, algorithms and depend on a different set of features extracted from a piece of text than text similarity algorithms. These algorithms take as input a sentence pair or a short text pair and produce a number that shows how related the two sentences are, often giving a value between 0.0 and 1.0. Sentence similarity algorithms are increasing in relevance and importance, as a growing amount of data is user-generated content [4], which is often shorter text or just one sentence. Applications of such sentence similarity algorithms are academic plagiarism detection, recommender systems in social networks, search in documents, and in specific domains like clustering biological data [5]. Sentence similarity can also improve the efficiency of data analysis on big datasets, serving as a preprocessing step before performing more intensive algorithms like stance detection [6] to make sure sentences are always related before their stance is determined.

Therefore, there is a need for a better understanding of sentence similarity, which is more dependent on the semantics of a specific sentence structure. There have been approaches to topical clustering of tweets [7], sentence embeddings [8] and determining sentence similarity based on word alignment and semantic vector composition [9]. However, these can be improved in accuracy by emphasizing the sentence structure, which was also a result of a survey by Farouk (2019) [10].

We seek to answer the following main question and its subquestions in this paper.

How can an algorithm determine similarity between sentences using a hybrid approach including semantics and structure?

1. How does a human determine the similarity between sentences?
2. What approaches have been tried before and what are their strong and weak points?
3. How to encode the semantics and structure of a sentence?
4. How to determine similarity between sentences using a

[1] In this paper the terms 'algorithm' and 'approach' are used interchangeably, as both go about how to solve a certain problem, with the difference being that they indicate a computational process and a thought process that solve the problem respectively.

given encoding?

5. How well does this similarity algorithm perform compared to human judgement?
6. How well does this similarity algorithm perform compared to other methods?

This paper will present a novel approach to determining sentence similarity inspired by existing approaches, emphasizing the combination of semantic and structure similarity. It will contribute to the existing research on short text and sentence similarity to address some of the shortcomings of existing approaches.

The paper is structured as follows; In Section 2 the context of the project is researched, answering the first two sub-questions. Human psychology and other common approaches are discussed to serve as a foundation for the design. In Section 3 the design of the novel algorithm is presented, taking inspiration from approaches mentioned in Section 2. This section also addresses the 3rd and 4th sub-questions. In Section 4 the designed algorithm is put to practice, being compared to a benchmark performed with humans and compared to existing algorithms to determine its performance. The methodology and approach are reflected upon in Section 5, answering the final two sub-questions. Further improvements and design recommendations are discussed in Section 6, after which the paper is concluded. The ethical implications and responsibility of the research are addressed in Section 7.

2 Problem Description

In this section, first, the aim is to understand the human psychology behind determining sentence similarity, describing the importance of relations in sentences. Then, some common practices for sentence similarity and several approaches are discussed, serving as the basis and theory for the design in the next section.

2.1 Background

Before discussing and understanding how computers determine the similarity between sentences, it is good to understand how a human might determine the similarity between sentences. The evaluation of the precision of similarity computed by a computerized approach should predict the similarity a human might assign to two sentences. Because of this, it makes sense to understand the human psychology on how people find sentences to be similar.

To determine the similarity between words, there has to be some alternative representation that makes calculating the similarity feasible. For instance, determining similarity between points is analogous to computing the distance, as their representation is numerical, and such a calculation would yield a shorter distance for more similar points. This does not work for words. Implying that similarity can be calculated using a distance function must mean that the axioms of the distance representation must hold. But, as discussed by Tversky, 1977 [11], this is not the case. Instead, he proposes a feature-based representation that enables the vectorization of words based on features. This is what is used for Word2Vec, a commonly used model for word embeddings [12].

A more general description of the notion of similarity is described in terms of relations, as discussed by several papers (Gentner (1983) [13]; McRae, Cree, Seidenberg, *et al.* (2005) [14]; Jones and Love (2007) [15]). Gentner discussed the strength of analogical matches to measure the performance of similarity metrics. He found that determining whether two sentences are analogous is characterized by the relations between the sentences instead of the attributes of the sentences. Sentences can be formulated as a series of predicates that structure the analogy and explain the relations. Furthermore, Gentner assessed that "overlap in relations is necessary for any strong perception of similarity between two domains." (Gentner, 1983, p. 161). Overlap in object attributes, meaning similarity between words, is seen as an appearance match. Overlap in relations, as described above, is seen as analogical relatedness. Having overlap in both object-attribute and relations means that two sentences have a literal similarity.

Relations can be subdivided into intrinsic and extrinsic relations. Intrinsic relations represent features of an item or a word in isolation. Extrinsic relations capture the relation between multiple entities. An example of an intrinsic relationship is a description of a word, such as 'cars have wheels' and 'a week has 7 days'. As Jones and Love (2007) hypothesize, extrinsic relations play an important role in determining similarity. They divide extrinsic relations into three categories.

- **Roles:** If an object of a relation represents the role it plays in the sentence, it can be determined to be similar to another object playing the same role. For instance, a *hammer* and a *baseball bat* can be seen as similar if they both have the role of *hit* in a given context.
- **Relations:** A more general relational notion of similarity then the *Roles* can be made if two objects are often found in the same relation. So even though in a given context they do not have the same role, since they have often been observed having the same role in other situations they can be seen as being similar. For example *hammer* and *baseball bat* will always have the common association of *hit*, therefore they are similar even if in a given example their roles are different.
- **Scenarios:** Objects have a notion of similarity if they occur in the same relation, exemplifying a direct relation. For instance, "The hammer hit the nail", is an example of a scenario based relation where *hammer* and *nail* have an increased similarity because they occur in the same relation.

Now, how can a human determine the similarity between sentences, preferably on a numeric scale, so that it can be compared to some computerized metric? This has been researched by O'Shea, et al. 2013 [16], with a benchmark dataset called STSS-131 that consists of sentence pairs and a human rating. Ultimately, deciding similarity is a subjective task; therefore, formulating the right task to determine similarity by participants was important. STSS-131 captures a representative corpus of the English language sentence pairs with 10-20 words per sentence. They produced human ratings between 0.0 and 4.0 that can be used to compare with algorithms to evaluate the accuracy of an algorithm.

2.2 Related Work

There have been several approaches for measuring sentence similarity, which can be divided into three categories [10]. The first category is word-based similarity, which considers only word similarity and calculates sentence similarity accordingly. Second, the structure-based similarity, which as the name implies relies on the structure of the sentence to determine similarity. The third and last category is vector-based similarity approaches, which depends on the sentence as a vector representation to calculate similarity. A closer look will be taken at a few common practices and approaches for determining sentence similarity here, which will be useful for the design and evaluation further on in the paper.

2.2.1 Preprocessing

To improve accuracy of a sentence similarity approach, several preprocessing steps can be taken to clean up a sentence before it is being evaluated. These steps are categorized by [17] as follows:

- **Basic Operations and Cleaning.** These are the very basic mutations of sentences, and based on the medium these have to be treated differently. These can be simple things like removing non-ASCII characters, line breaks, extra blank spaces, and converting words to all lowercase letters. As exemplified by [17], the use case was tweets thus things like hashtags and mentions also had to be removed.
- **Emoticons.** These are the emoticons made using special characters, like *:)* and *;(*. These can be substituted by *smile_positive* and *smile_negative*.
- **Negation.** The substitution of all negative constructs, like *won't* and *can't* by the word *not*, which by itself is unlikely but because of this substitution negated statements can be generalized and compared for similarity more easily.
- **Dictionary.** As one cannot rely on words to always be perfectly spelled, a spell check can be used to detect and correct misspelled words.
- **Stemming.** The stemming of words will reduce words to their base construct, ensuring that different formulations of verbs are similar and superlatives like *great*, *greater*, *greatest* can be reduced to their smallest form *great*.
- **Stopwords.** Stop words are words that are a common occurrence in sentences but that do not add additional information to the meaning of the sentence such as pronouns and articles. A commonly used list of stopwords is from the NLTK library suite[18]. If a stop word is found in a sentence, it shall be removed to ensure it is not considered in the model.

From [17] it can be concluded that doing the basic operations and stemming as preprocessing steps will result in the best performance, and will be a good way to prepare the sentences before similarity is computed.

2.2.2 Word Embedding (Word2Vec vs GloVe)

A common approach for calculating sentence similarity is from the semantic similarity properties derived from the word embeddings. These approaches are thus word-based similarities. The aim of word embeddings is to encode a word as a numerical or vectorized representation. Some approaches for word embeddings can be described as scoring schemes, like Bag of Words [19] and TF-IDF [20] and do not benefit from the semantic similarity between words. Instead, pre-trained models can generate word vectors that capture the semantic similarity between words. The two most common examples are Word2Vec [12] and GloVe [21]. The biggest distinction between these models is that Word2Vec is a predictive model and GloVe is a count-based model, a difference which is further discussed by Baroni et al. 2014 [22].

After retrieving the individual word vectors, the overall sentence vector can be achieved by taking the mean of all the word vectors [23]. This approach can be found as a simple implementation as well, using Gensim's doc2vec model [24]. After the sentence vectors have been calculated, similarity can be calculated using the cosine similarity, which is a measure of similarity between two vectors that is most commonly used.

2.2.3 Smooth Inverse Frequency

Taking the mean of all the word vectors as mentioned above is an easy way to formulate the sentence embedding but fails to capture the importance of the relevance of words. Even though stop words might be filtered out, some words are still more dominant in deciding the similarity between sentences. Thus, a weight should be attached to each word for its importance in the sentence when calculating the overall sentence embedding. Smooth Inverse Frequency was proposed by Arora, Liang, and Ma (2019) [25] as a simple approach to estimate the importance of words by looking at the frequency of the words in a given corpus.

2.2.4 Word Mover's Distance

An approach proposed by M.J. Kusner et al. in 2015 [26] is used as a distance function between text documents. Word Mover's Distance (WMD) uses word embeddings from the Word2Vec model to represent text documents as a point cloud. The distance is calculated by finding the minimum cumulative distance between two text documents for one document to travel to the other point cloud. WMD can be an intriguing approach as it is hyperparameter free, gives an intuitive and easy interpretation of distance between documents, and performs well on text documents.

Its applications are often used for clustering documents, and its performance is compared in [26] with k-nearest neighbours [27]. Implementation of the algorithm has been made easy by Gensim [28], making it easy to work with as well. Given that the distance function, in this case indicating the similarity, is not a number between 0 and 1, its performance is more difficult to compare and combine with other approaches.

2.2.5 Word Order

Going from word-based to structure-based approaches, the Word Order algorithm is a simple approach to take the order of words into account. The motivation for finding the word order is that if two sentences contain the same words, yet in a different order, the sentence could have a different meaning.

A hybrid approach proposed by Y. Li et al. (2004) [29] takes the Word Order as one of the two components of sentence similarity. Their novel approach to calculating Word Order similarity between two sentences t_1 and t_2 is to create the joint word set T that contains all words of t_1 and t_2, and then convert the list of words of each sentence by the index of the word in the set T, creating two integer vectors r_1 and r_2. Then the similarity is computed as: $S_r = 1 - \frac{\|r_1 - r_2\|}{\|r_1 + r_2\|}$.

Word Order can be useful for sentences with similar word usage and sentence length but falls short on sentences with different words and lengths, causing the Word Order similarity to be very low even though the semantic similarity might be high. Y. Li et al. countered this by combining it with a word similarity metric and having a hyperparameter decide what weight each similarity metric has on the total similarity.

2.2.6 Sentence Embedding (SentenceBERT)

The last approach that will be considered in this paper is a vector-based similarity approach and, more specifically, a sentence-based similarity algorithm. Instead of dealing with word-based similarity, the sentence as a whole will be represented as a vector, after which the similarity can again be calculated using the cosine similarity. As outlined by P. Huilgol (2020) [30], the most promosing approach is SentenceBERT. Introduced by N. Reimers and I. Gurevych in 2019 [31], SentenceBERT is a sentence embedding algorithm that uses the Siamese BERT [32] network.

2.2.7 Combined Semantic and Syntactic Measures

Each approach has its problems, and it depends on the task at hand what approach is most well suited for a particular application. Word-based similarity approaches fail to capture the structural information of the sentences, and thus use of relations and order in the sentence does not matter. Structure-based similarity approaches fail to capture the actual semantic similarity, where the use of synonyms and related words are not well addressed. Sentence embedding approaches, as discussed above, is right in between the two. Depending on how a model is trained, it could be more sensitive to the semantics or the structure of a sentence and thus behave more like a word-based or structure-based approach.

3 Design

In this section, a novel algorithm, which is divided into two parts, is proposed. The first part takes advantage of the semantic similarity, called the SIF Semantic Similarity [25], to determine sentence similarity. The second part takes advantage of the structure of a sentence, allowing for comparison between sentences based on the occurrence and order of the words in a sentence. The second part is called the Word Mover's Order Similarity, and it is the novel part of the algorithm, as it combines the Word Order [29] with Word Mover's Distance [26] algorithm.

The implementation of the algorithm with the respective preprocessing steps will be explained to ensure further evaluation is reproducible. Since the algorithm comes with a three hyperparameters (α, ι, and κ), in the final section it will be discussed how those will be tuned to get the best performance for the algorithm. This section will thus cover the different components of the novel algorithm and how it can be implemented and optimized, after which this can be used in the next section to perform evaluation on it to measure the performance.

$$v_s(s) = \frac{1}{n} \sum_{w \in s}^{n} \frac{\alpha}{\alpha + zipf(w)} \cdot w2v(w) \tag{1}$$

$$sim_{sem}(s_1, s_2) = \frac{v_s(s_1) \cdot v_s(s_2)}{||v_s(s_1)|| \cdot ||v_s(s_2)||} \tag{2}$$

3.1 SIF Semantic Similarity

The SIF (Smooth Inverse Frequency) Semantic Similarity determines similarity between two sentences strictly by the cumulative similarity of the words in the sentences. This implementation has been largely inspired by Arora, Liang, and Ma (2019) [25]. The design relies on a word embedding algorithm to translate the words in a sentence to a vector, after which these have to be combined such that one vector can be composed that describes the whole sentence (Equation 1). In this approach, SIF is used, which was explained in Section 2.2.3. With the vector representations of each sentence, the similarity can be calculated using the cosine similarity (Equation 2). For the implementation the Word2Vec model has been used for word embeddings and the zipf_frequency from the wordfreq package is used for weighing words, which gives the frequency of a word on logarithmic scale. Equation 1 and Equation 2 are the formulas to calculate the semantic similarity of a sentence pair using the SIF Semantic Similarity method, which results in a number between 0.0 and 1.0.

3.2 Word Mover's Order Similarity

The second part of the algorithm takes advantage of the structure of a sentence to determine the similarity of a sentence pair. It combines two approaches discussed before, namely Word Mover's Distance from Section 2.2.4 and Word Order from Section 2.2.5. These have been chosen as they both say something about the structure of a sentence and align with the notion that a sentence is comprised of relations, showing the relation of linearity and roles of words in a sentence. However, both approaches have a big constraint that limit their accuracy to determine similarity. Word Mover's Distance's constraint, as explained in Section 2.2.4, is that it gives a distance value which is not between 0.0 and 1.0 and it is therefore not suitable for evaluation. Word Order, as explained in Section 2.2.5, on the other hand is good at embedding the structure of a sentence, but only works for words that are exactly the same and does not take the most similar words into account to compare ordering. For this reason Word Order is often combined with a semantic similarity approach, as in [29]. As both approaches have their strong points, our approach sought out to combine those strong points to be able to take advantage of the structure of the sentence to improve their performance.

The Word Mover's Order Similarity metric combines Word Mover's Distance and Word Order. The limitation of Word Order, which is that the words have to be exactly the same, can be remedied by the attribute of Word Mover's Distance that aims to find the most similar pairs of words. Most similar pairs can be found by constructing a similarity matrix and eliminating most similar pairs until all words are considered. If one sentence is longer than the other, the left over words can be paired with the words of the other sentence that they are most similar to as well. After the pairs have been found, Word Order can be applied to calculate the distance between the positions of the words of a pair in their respective sentence. The final similarity can be calculated by taking the average of all minimal pairs, after multiplying the similarity of the words in the pairs with the inverse of the distance of the words (Equation 3).

$$sim_{wmo}(s_1, s_2) = \frac{1}{n} \sum_{p_{min}(w_1, w_2)}^{n} \frac{1}{1 + \kappa \cdot dist_{wo}(w_1, w_2)} \cdot w2v(w_1, w_2) \tag{3}$$

3.3 Total Similarity

The total similarity can now be calculated given the semantic and structural similarity components. The sensitivity to either similarity metrics can be tweaked using the ι as a hyperparameter, which together with the α and κ will be tuned later on to ensure the algorithm works optimally. The total algorithm implements Equation 4.

$$sim(s_1, s_2) = \iota \cdot sim_{sem}(s_1, s_2) + (1 - \iota) \cdot sim_{wmo}(s_1, s_2) \tag{4}$$

3.4 Implementation

The implementation of the novel algorithm proposed in this paper is publicly available on a GitLab repository from the TU Delft[2]. The algorithm has several dependencies and those will be discussed here to ensure that the research is reproducible. The algorithm is implemented using the Python programming language. As mentioned before the algorithm uses Word2Vec [12] for word embeddings, specifically the *GoogleNews-vectors-negative300-SLIM.bin*[3] pretrained model. Zipf_frequency from the wordfreq Python package [33] is used for the SIF Semantic Similarity. The cosine_similarity function from sklearn is used to calculate similarity between vectors in the SIF Semantic Similarity part. All Python package version dependencies can be found in the *requirements.txt* file. Installing all of these dependencies is necessary to be able to run the algorithm.

[2]https://gitlab.ewi.tudelft.nl/cse3000/2020-2021/rp-group-65/rp-group-65-whaakman

[3]urlhttps://github.com/eyaler/word2vec-slim/blob/master/GoogleNews-vectors-negative300-SLIM.bin.gz

Some preprocessing is required before calculating the similarity of a sentence pair. In this implementation three preprocessing steps have been implemented and tested on, which has been inspired from Section 2.2.1 and [17]. First, all characters in the sentences are converted to lowercase letters. Second, all punctuation is removed from the sentence making sure that comma's and periods are not considered. After this, the sentence is tokenized into words by splitting on spaces. The words can then be stemmed to their word stem. Tokenizing a sentence is always necessary, and removing punctuation will make sure that they do not influence the similarity. After some optimization it turned out stemming words worsened the performance and is therefore not used for evaluation but the implementation is still there. Each one of these preprocessing steps can be turned on or off to ensure that either of these steps will not affect the performance negatively.

3.5 Hyperparameter Tuning

As discussed before, the algorithm contains three hyperparameters in total, namely α, ι, and κ. These hyperparameters, just like the preprocessing steps, have to be optimized before evaluation to ensure the performance of the algorithm is optimal. α controls the effect of SIF to the Semantic Similarity. As the zipf_frequency used will give different results than the one used by Arora, Liang, and Ma [25], the $\alpha = 10^{-3}$ mentioned in that paper cannot be reused. κ is used to control the sensitivity of Word Mover's Order to the Word Order distance. ι is used to control the sensitivity of the total similarity to both the semantic and structural similarity. It is bounded between 0.0 and 1.0 as it singifies the percentage one or the other is used, so 0.3 means that the total similarity is comprised for 30% by the SIF Semantic Similarity and 70% by the Word Mover's Order Similarity.

Hyperparameter tuning is the process of finding a value for a given hyperparameter that results in the highest performance for the algorithm. Two tuning algorithms have been tried to find the optimal hyperparameters. In this paper Grid Search is used to optimize the hyperparameters [34]. The optimization was performed on the MSRP sentence pair dataset [35], consisting of 5800 rows of sentence pairs. The parameters were optimized on the training dataset, and the final performance was found using the test dataset. The performance is further discussed in Section 4. The optimized hyperparameters are $\alpha = 0.6876$, $\kappa = 0.09611$ and $\iota = 0.3$.

4 Evaluation

The evaluation will be performed on two different datasets. First the algorithm will be compared against human judgement using the STSS-131 dataset [16]. The algorithm will also be evaluated on the MSRP dataset [35] to see how it performs against a broader set of algorithms. In both evaluations the proposed algorithm is compared against already evaluated algorithms and also against SentenceBERT, which is a sentence embedding approach explained in Section 2.2.6, using the same preprocessing steps. The reason that SentenceBERT is used is because it is a deep learning algorithm, which contrasts with the traditional machine learning algorithm proposed in this paper. Because of this it is expected it

will perform better but it is also less explainable, which will be discussed more in Section 5. At the end of this section it will be clear how well this novel algorithm performs against other common approaches.

4.1 Benchmark STSS-131

To evaluate the proposed algorithm against human performance the STSS-131 dataset has been used [16], which consists of 66 sentence pairs with a continuous value between 0.0 and 4.0. The values are normalized between 0.0 and 1.0. The explanation of the dataset can also be found in Section 2.1. The data provided in the paper has also been used for comparison. To offer another perspective, STSS-131 has also been used to evaluate SentenceBERT. As the result of the similarity algorithm gives a continuous value between 0.0 and 1.0, giving performance measures like the accuracy and F1-score is not possible. However it can be assumed that the actual similarity and the predicted similarity have a linear relationship so the Pearson correlation coefficient can be calculated instead. The r value thus gives an indication of the performance, and higher values are better. The results are shown in Table 1. As shown in the table, SentenceBERT has the highest correlation showing how this sentence embedding approach works exceptionally better than the other approaches, and the proposed novel algorithm is around 3 percent and 9 percent better than LSA and STASIS respectively, which are two approaches not discussed in this paper but used to compare with as they are used by O'shea, Bandar, and Crockett (2014) [16]. These findings will be further discussed in Section 5.

Table 1: Evaluation of proposed algorithm against SentenceBERT, LSA, and STASIS on STSS-131.

Sentence Similarity Measures	STASIS	LSA	Sentence-BERT	Proposed Algorithm
Pearson Correlation Coefficient	0.636	0.693	0.936	0.721

4.2 Benchmark MSRP

The proposed algorithm will also be evaluated against other more common approaches, some of which have been discussed in Section 2.2. The MSRP dataset [4] from Microsoft is used for evaluation as Achananuparp, Hu, and Shen [36] evaluated many common approaches against it. The MSRP dataset consists of 5800 sentence pairs and each sentence pair has one value, namely a 0 for dissimilar or 1 for similar. This means that our proposed algorithm should be reformulated into a classification problem, converting a continuous value to either a 0 or 1. This can be achieved by simply rounding the given similarity value. Given the reformulation, the precision, recall, F1-score, and accuracy can be calculated. From the dataset, specifically the test set has been used for evaluation as the training dataset is used for the hyperparameter tuning. The dataset has also been used to evaluate SentenceBERT. The results can be found in table 2. As seen in the table SentenceBERT has the best F1-score but the proposed novel algorithm has the highest accuracy. These findings will be further discussed in Section 5.

[4] https://www.microsoft.com/en-us/download/details.aspx?id=52398

Table 2: Evaluation of proposed algorithm against SentenceBERT and other common approaches on MSRP.

Sentence Similarity Measures	Prec.	Rec.	F_1	Acc.
$\text{sim}_{jaccard}$	0.835	0.603	0.7	0.657
sim_{sem}	0.674	0.99	0.802	0.675
sim_{wo}	0.681	0.619	0.648	0.554
sim_{sem+wo}	0.674	0.977	0.8	0.671
SentenceBERT	0.687	0.994	0.812	0.694
Proposed Algorithm	0.742	0.856	0.795	0.715

5 Discussion

This section will dive further into the results and an explanation will be provided as to why certain results have been achieved. First an explanation will be provided for why the tuning of the hyperparameters resulted in the values mentioned in this paper, and second the results of the algorithm on the STSS-131 and MSRP will be discussed. It is noteworthy that the proposed algorithm is a traditional machine learning algorithm, not requiring any deep learning, making the results more explainable. This is in contrast with SentenceBERT, which was also used for evaluation, as that acts more like a black box and it is harder to reason on. After this section it should be clear why the algorithm performs the way it does, which makes identifying future improvements easier.

5.1 Effects of Hyperparameters

The optimized hyperparameters used by the algorithm are $\alpha = 0.6876$, $\kappa = 0.09611$, and $\iota = 0.3$ and these values were achieved by performing grid search as discussed in Section 3.5. The question can be posed as to why these values are what they are and how they effect the performance of the algorithm. The α value is used to determine the sensitivity to zipf_frequency. In the original paper from Arora, Liang, and Ma [25] the value of α was 10^{-3}, and therefore the newly found α is considerably higher. The reason for this is because the frequency model used in the original paper is different than zipf_frequency used in this paper. κ controls the sensitivity to the word order and is very low in this paper. This shows that the word order distance used was not very effective in improving the performance of the algorithm, showing that it was maybe not very useful. However, ι got a value of 0.3, meaning that the algorithms total similarity is 30 percent dependent on the SIF Semantic Similarity and 70 percent on the Word Mover's Order similarity. This shows that the total similarity is influenced most by the novel component of the algorithm, namely the combination of Word Order and Word Mover's Distance. This shows how these values affect the performance of the algorithm and sensitivity to certain components in the algorithm.

5.2 Results STSS-131

The results have been shown in the evaluation in Section 4. The algorithm was first evaluated on the STSS-131 dataset, which consists of only 66 sentence pairs. This dataset does however contain a lot of different cases and is a good representation of the different challenges of determining similarity. SentenceBERT performs exceptionally well, with a 0.936 correlation. This is assumed to be because it is a deep learning approach which generalizes better and depending on how it is trained can therefore give much better results than traditional machine learning approaches. The proposed algorithm performs reasonably better than the other approaches mentioned in the original paper [16]. For instance, LSA [5] performs with a correlation of 0.693 and the proposed algorithm with a correlation of 0.721. This increase in performance is probably due to the combination of several common approaches and the hyperparameter tuning which was performed.

Relating the results back to the second to last subquestion posed in Section 1, the evaluation against STSS-131 shows how the proposed algorithm performs against human judgmenet. A higher correlation coefficient thus shows that a given algorithm more closely matches how humans might determine similarity between sentences. As mentioned before, SentenceBERT has the highest correlation meaning it most closely matches the human judgement. Determining how well an algorithm performs compared to humans is subjective however, as a single value like the correlation coefficient does not give context to its usefulness. From the results it can be shown that the proposed algorithm performs best out of all traditional machine learning approaches.

5.3 Results MSRP

The proposed algorithm was also evaluated on the MSRP dataset, consisting of 5800 sentence pairs. As the dataset is formulated as a classification problem, several performance metrics could be calculated to easily investigate and compare the performance of several algorithms. The performance of SentenceBERT was not much better than other approaches, scoring only an accuracy of 0.694 and an F_1 score of 0.812. The performance of the proposed algorithm was reasonably better than the other algorithms. The algorithm has an F_1 score of 0.795 similar to some of the other higher scores and the highest accuracy of 0.715. It is also noticeable how similar all the metrics are to the other approaches like sim_{sem+wo}, which makes sense as this approach uses many of the same components combining semantic similarity and the Word Order algorithm. After discussing the results of the proposed algorithm on the STSS-131 and MSRP dataset it can be seen that the algorithm performs better than other traditional approaches with a margin, and depending on the application it can perform better than SentenceBERT although less likely.

Relating the results back to the last subquestion posed in Section 1, the evaluation against MSRP shows how the proposed algorithm performs against other common approaches. As shown from the results discussed above the proposed algorithm performed reasonably better in terms of accuracy and relatively equal in F_1 score, indicating that the proposed algorithm performs similar if not better than other approaches.

6 Conclusion and Future Work

This paper proposed a novel algorithm to determine sentence similarity, combining semantic and structural similarity metrics. This paper also presented its implementation, optimization, and evaluation, comparing it to several other approaches showing its performance to be better than other approaches within a certain degree. Several sub questions were stated in the introduction in Section 1, which have all been addressed throughout the sections. An interesting connection has been made between human psychology and existing similarity algorithms, showing how differently similarity is perceived yet how human psychology can possibly translate to a computerized approach. This paper showed that the novel algorithm performs relatively similar if not better than other approaches, yet it still does not outperform deep learning. The question on how an algorithm can determine similarity using a hybrid approach has been answered following the design, and this can be reproduced by the reader using the resources provided in this paper.

6.1 Improvements and Recommendations

A few recommendations can be made, after having worked on this problem and discovering flaws in the proposed algorithm, to further improve the algorithm. The hyperparameter ι used to control the influence of semantic and structural similarity for the total similarity showed that the optimized value is 0.3 meaning that structural similarity is favored. However, κ was a low value of 0.09611, showing that Word Order was not used effectively. The way Word Order will be expressed in a future implementation could be improved as in the current implementation it's nearly nullified.

As Grid Search was used to optimize the hyperparameters it is possible that these values are not the absolute optimum. For this it is recommended to still try out a better hyperparameter tuning algorithm, like Bayesian Optimization [37]. For anyone interested in implementing the proposed algorithm, the question can be posed as to whether it makes sense to use a traditional machine learning approach as opposed to a deep learning approach like SentenceBERT. The advantage of using the proposed algorithm, or other traditional algorithms is its explainability, compromising on the performance of the algorithm compared to deep learning approaches. However, making a conclusive statement on the comparison of traditional machine learning with deep learning is outside of the scope of this paper.

7 Responsible Research

The relevance of responsible research is of much importance. To follow the scientific method [6] it is important that the conducted research is reproducible and verifiable. Besides that, from the paper it should be clear that a hypothesis is being tested, and through conducting an experiment, data can be

[5] https://towardsdatascience.com/latent-semantic-analysis-distributional-semantics-in-nlp-ea84bf686b50

[6] https://upload.wikimedia.org/wikipedia/commons/thumb/8/82/The_Scientific_Method.svg/1200px-The_Scientific_Method.svg.png

analyzed which can then support or refute the posed hypothesis. From the structure of this paper it should be clear that the scientific method was followed. In the introduction 1 the motivation and research question were introduced. Based on the background research in the problem description 2, and the novel algorithm proposed in the design Section 3, an experiment has been conducted in the evaluation stage 4. The results will be communicated and explained in the discussion 5 and the research will be concluded on in the last Section 6. The following three sections show the consideration of ethical aspects on the bias of the algorithm, the reason STSS-131 [16] and MSRP [35] were picked as datasets, and the reproducibility of the research.

7.1 Bias

The performance of the algorithm relies not only on the design but also on a couple pretrained models and certain hyperparameters. It is of importance, for fair and ethical AI, to be aware of any bias in an algorithm, also discussed by K. Escherich [38], and therefore an overview will be given of possible bias in the proposed algorithm. In Section 3.4 the dependencies of the algorithm have been discussed, and two pretrained models. For word embeddings the model *GoogleNews-vectors-negative300.bin.gz* has been used, whose data has been obtained from Google News and consists of 100 billion words. To get the word frequencies used for SIF the library wordfreq [33] is used. The data was gathered using *Exquisite Corpus*[7], "whose goal is to download good, varied, multilingual corpus data, process it appropriately, and combine it into unified resources such as wordfreq" [39]. This shows that the data used is relatively general and there was an effort to minimize bias.

In Section 3.5 the process of tuning the three hyperparameters involved in the algorithm is shown. These hyperparameters are trained on the training dataset of MSRP, making sure they are not overfitting on the test dataset of MSRP. It is clear what hyperparameters have been used for the evaluation and how these values were found. However, it should also be clear that these chosen hyperparameters directly affect the algorithms sensitivity to certain components. For instance, a higher ι means that the algorithm is more sensitive to the semantic similarity which could introduce more bias if the word embedding model used is more biased. Fortunately the algorithm is not a decisive algorithm and will therefore not directly affect any big decisions made in a system. The ethical implications depend heavily on how it is used.

7.2 Datasets

In the training of the hyperparameters and the evaluation two datasets were picked and the results of those were presentend in 4. Whenever one does research it is important to note the validity of the results, also showing that the provided results are not cherry picked to make the algorithm seem to perform better than other algorithms. The motivation for picking the STSS-131 dataset was to compare it with human judgment. The STSS collection of datasets contains sentence pairs that have been picked and provided with a lot of consideration for the fairness and correct representation of how humans determine similarity. The hyperparameters were trained on the training set of sentence pairs of MSRP. MSRP has been chosen because many other common approaches were evaluated on this dataset [36]. This makes sure that our evaluation can be compared with a representative set of other algorithms. There are not a lot of datasets containing similarities of sentence pairs, and MSRP was by far the biggest one.

7.3 Reproducibility

As the aim for this paper was to propose a novel sentence similarity algorithm, its usefulness depends on how easy its implementation is for experimenting and applying in other projects. The aim for Section 3 was to make the algorithm explainable and reproducible. The code can also be found online [8]. From the evaluation in Section 4 it is also clear what datasets have been used and how the evaluation metrics were achieved.

Acknowledgements

I wish to express my sincere gratitude to the supervisor and responsible professor of our research project for the course CSE3000 Research Project, Asst. Prof. Pradeep Murukannaiah. Our research group proposed our own research topic and he decided to supervise us because of his own interest in the topic. Thanks to his effort to supervise us, the weekly meetings, and the willingness to always answer questions via email, I was able to successfully finish my research with satisfactory results.

I am also thankful to my fellow students Abel van Steenweghen, Jacob Roeters van Lennep, Kristóf Vass, and Simon Mariën. Together we formed a research group and we proposed our own research topic. With their support and cooperation we were able to elavate each others' work. Working together was also very motivating to me personally.

[7] https://github.com/LuminosoInsight/exquisite-corpus

[8] https://gitlab.ewi.tudelft.nl/cse3000/2020-2021/rp-group-65/rp-group-65-whaakman

References

[1] A. Strehl, J. Ghosh, and R. Mooney, "Impact of similarity measures on web-page clustering," in *Proceedings of the AAAI Workshop on AI for Web Search (AAAI 2000)*, Austin, TX, USA, 2000, pp. 58–64.

[2] S. Tata and J. M. Patel, "Estimating the selectivity of tf-idf based cosine similarity predicates," *SIGMOD Rec.*, vol. 36, no. 2, pp. 7–12, Jun. 2007, ISSN: 0163-5808. DOI: 10.1145/1328854.1328855. [Online]. Available: https://doi.org/10.1145/1328854.1328855.

[3] S. Albitar, S. Fournier, and B. Espinasse, "An effective tf/idf-based text-to-text semantic similarity measure for text classification," in *Web Information Systems Engineering – WISE 2014*, B. Benatallah, A. Bestavros, Y. Manolopoulos, A. Vakali, and Y. Zhang, Eds., Cham: Springer International Publishing, 2014, pp. 105–114, ISBN: 978-3-319-11749-2.

[4] S. Melumad, J. J. Inman, and M. T. Pham, "Selectively emotional: How smartphone use changes user-generated content," *Journal of Marketing Research*, vol. 56, no. 2, pp. 259–275, 2019. DOI: 10.1177/0022243718815429. eprint: https://doi.org/10.1177/0022243718815429. [Online]. Available: https://doi.org/10.1177/0022243718815429.

[5] M. Han, X. Zhang, X. Yuan, J. Jiang, W. Yun, and C. Gao, "A survey on the techniques, applications, and performance of short text semantic similarity," *Concurrency and Computation: Practice and Experience*, vol. 33, no. 5, e5971, 2021. DOI: https://doi.org/10.1002/cpe.5971. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/cpe.5971. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.5971.

[6] D. Küçük and F. Can, "Stance detection: A survey," *ACM Comput. Surv.*, vol. 53, no. 1, Feb. 2020, ISSN: 0360-0300. DOI: 10.1145/3369026. [Online]. Available: https://doi.org/10.1145/3369026.

[7] K. Dela Rosa, R. Shah, B. Lin, A. Gershman, and R. Frederking, "Topical clustering of tweets," *3Rd Workshop on Social Web Search and Mining*, Jan. 2011, (accessed: 19.04.2021).

[8] D. Cer, Y. Yang, S.-y. Kong, N. Hua, N. Limtiaco, R. St. John, N. Constant, M. Guajardo-Cespedes, S. Yuan, C. Tar, B. Strope, and R. Kurzweil, "Universal sentence encoder for English," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, Brussels, Belgium: Association for Computational Linguistics, Nov. 2018, pp. 169–174. DOI: 10.18653/v1/D18-2029. [Online]. Available: https://www.aclweb.org/anthology/D18-2029.

[9] M. A. Sultan, S. Bethard, and T. Sumner, "DLS@CU: Sentence similarity from word alignment and semantic vector composition," in *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, Denver, Colorado: Association for Computational Linguistics, Jun. 2015, pp. 148–153. DOI: 10.18653/v1/S15-2027. [Online]. Available: https://www.aclweb.org/anthology/S15-2027.

[10] M. Farouk, "Measuring sentences similarity: A survey," vol. 12, pp. 1–11, 25 2019. DOI: https://dx.doi.org/10.17485/ijst/2019/v12i25/143977. [Online]. Available: https://indjst.org/articles/measuring-sentences-similarity-a-survey.

[11] A. Tversky, "Features of similarity," *Psychological Review*, vol. 84, no. 4, pp. 327–352, 1977. DOI: 10.1037/0033-295X.84.4.327.

[12] J. Gilyadov. (Mar. 2017). "Word2vec explained," [Online]. Available: https://israelg99.github.io/2017-03-23-Word2Vec-Explained/.

[13] D. Gentner, "Structure-mapping: A theoretical framework for analogy," *Cognitive Science*, vol. 7, no. 2, pp. 155–170, 1983, ISSN: 0364-0213. DOI: https://doi.org/10.1016/S0364-0213(83)80009-3. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0364021383800093.

[14] K. McRae, G. Cree, M. Seidenberg, and C. Mcnorgan, "Semantic feature production norms for a large set of living and nonliving things," *Behavior research methods*, vol. 37, pp. 547–59, Dec. 2005. DOI: 10.3758/BF03192726.

[15] M. Jones and B. C. Love, "Beyond common features: The role of roles in determining similarity," *Cognitive Psychology*, vol. 55, no. 3, pp. 196–231, 2007, ISSN: 0010-0285. DOI: https://doi.org/10.1016/j.cogpsych.2006.09.004. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0010028506000715.

[16] J. O'shea, Z. Bandar, and K. Crockett, "A new benchmark dataset with production methodology for short text semantic similarity algorithms," *ACM Trans. Speech Lang. Process.*, vol. 10, no. 4, Jan. 2014, ISSN: 1550-4875. DOI: 10.1145/2537046. [Online]. Available: https://doi.org/10.1145/2537046.

[17] F. Magliani, T. Fontanini, P. Fornacciari, S. Manicardi, and E. Iotti, "A comparison between preprocessing techniques for sentiment analysis in twitter," Dec. 2016.

[18] sebleier. (2010). "Nltk's list of english stopwords," [Online]. Available: https://gist.github.com/sebleier/554280. (accessed: 03.05.2021).

[19] J. Brownlee. (2019). "A gentle introduction to the bag-of-words model," [Online]. Available: https://machinelearningmastery.com/gentle-introduction-bag-words-model/. (accessed: 03.05.2021).

[20] R. Madan. (2019). "Tf-idf/term frequency technique: Easiest explanation for text classification in nlp using python (chatbot training on words)," [Online]. Available: https://medium.com/analytics-vidhya/tf-idf-term-frequency-technique-easiest-explanation-for-text-classification-in-nlp-with-code-8ca3912e58c3. (accessed: 03.05.2021).

[21] Sciforce. (Aug. 2018). "Word vectors in natural language processing: Global vectors (glove)," [Online]. Available: https://medium.com/sciforce/word-vectors-in-natural-language-processing-global-vectors-glove-51339db89639.

[22] M. Baroni, G. Dinu, and G. Kruszewski, "Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Baltimore, Maryland: Association for Computational Linguistics, Jun. 2014, pp. 238–247. DOI: 10.3115/v1/P14-1023. [Online]. Available: https://www.aclweb.org/anthology/P14-1023.

[23] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ser. ICML'14, Beijing, China: JMLR.org, 2014, II–1188–II–1196.

[24] Gensim. (2021). "Doc2vec paragraph embeddings," [Online]. Available: https://radimrehurek.com/gensim/models/doc2vec.html. (accessed: 03.05.2021).

[25] S. Arora, Y. Liang, and T. Ma, "A simple but tough-to-beat baseline for sentence embeddings," English (US), 5th International Conference on Learning Representations, ICLR 2017 ; Conference date: 24-04-2017 Through 26-04-2017, Jan. 2019. [Online]. Available: https://openreview.net/pdf?id=SyK00v5xx.

[26] M. J. Kusner, Y. Sun, N. I. Kolkin, and K. Q. Weinberger, "From word embeddings to document distances," in *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ser. ICML'15, Lille, France: JMLR.org, 2015, pp. 957–966.

[27] O. Harrison. (2018). "Machine learning basics with the k-nearest neighbors algorithm," [Online]. Available: https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761. (accessed: 03.05.2021).

[28] Gensim. (2009). "Word mover's distance," [Online]. Available: https://radimrehurek.com/gensim/auto_examples/tutorials/run_wmd.html. (accessed: 03.05.2021).

[29] Y. Li, Z. Bandar, D. McLean, and J. O'Shea, "A method for measuring sentence similarity and its application to conversational agents.," Jan. 2004.

[30] P. Huilgol. (2020). "Top 4 sentence embedding techniques using python!" [Online]. Available: https://www.analyticsvidhya.com/blog/2020/08/top-4-sentence-embedding-techniques-using-python/. (accessed: 05.05.2021).

[31] N. Reimers and I. Gurevych. (2019). "Sentence-bert: Sentence embeddings using siamese bert-networks." arXiv: 1908.10084 [cs.CL].

[32] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. (2019). "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv: 1810.04805 [cs.CL].

[33] R. Speer, J. Chin, A. Lin, S. Jewett, and L. Nathan, *Luminosoinsight/wordfreq: V2.2*, Oct. 2018. DOI: 10.5281/zenodo.1443582. [Online]. Available: https://doi.org/10.5281/zenodo.1443582.

[34] R. Joseph. (2018). "Grid search for model tuning," [Online]. Available: https://towardsdatascience.com/grid-search-for-model-tuning-3319b259367e. (accessed: 02.06.2021).

[35] W. B. Dolan and C. Brockett, "Automatically constructing a corpus of sentential paraphrases," in *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*, 2005. [Online]. Available: https://www.aclweb.org/anthology/I05-5002.

[36] P. Achananuparp, X. Hu, and X. Shen, "The evaluation of sentence similarity measures," vol. 5182, Sep. 2008, pp. 305–316, ISBN: 978-3-540-85835-5. DOI: 10.1007/978-3-540-85836-2_29.

[37] W. Koehrsen. (2018). "A conceptual explanation of bayesian hyperparameter optimization for machine learning," [Online]. Available: https://towardsdatascience.com/a-conceptual-explanation-of-bayesian-model-based-hyperparameter-optimization-for-machine-learning-b8172278050f. (accessed: 02.06.2021).

[38] K. Escherich. (2019). "Why do we need to talk about ethics and bias in ai?" [Online]. Available: https://www.ibm.com/blogs/nordic-msp/ethics-and-bias-in-ai/. (accessed: 06.06.2021).

[39] R. Speer. (2018). "Luminosoinsight/wordfreq: Access a database of word frequencies, in various natural languages.," [Online]. Available: https://github.com/LuminosoInsight/wordfreq/#sources-and-supported-languages. (accessed: 06.06.2021).