Explainable Semantic Textual Similarity via Dissimilar Span Detection

Diego Miguel Lozano[1,†], Daryna Dementieva[1,2], Alexander Fraser[1,2]
[1]School of Computation, Information and Technology
Technical University of Munich
[2]Munich Center for Machine Learning (MCML)
diego@ellisalicante.org, {daryna.dementieva, alexander.fraser}@tum.de

Abstract

Semantic Textual Similarity (STS) is a crucial component of many Natural Language Processing (NLP) applications. However, existing approaches typically reduce semantic nuances to a single score, limiting interpretability. To address this, we introduce the task of Dissimilar Span Detection (DSD), which aims to identify semantically differing spans between pairs of texts. This can help users understand which particular words or tokens negatively affect the similarity score, or be used to improve performance in STS-dependent downstream tasks. Furthermore, we release a new dataset suitable for the task, the Span Similarity Dataset (SSD), developed through a semi-automated pipeline combining large language models (LLMs) with human verification. We propose and evaluate different baseline methods for DSD, both unsupervised—based on LIME, SHAP, LLMs, and our own method—as well as an additional supervised approach. While LLMs and supervised models achieve the highest performance, overall results remain low, highlighting the complexity of the task. Finally, we set up an additional experiment that shows how DSD can lead to increased performance in the specific task of paraphrase detection.

Keywords: semantic textual similarity, dissimilar span detection, explainability

1. Introduction

Semantic Textual Similarity (STS) is a fundamental concept in Natural Language Processing (NLP), being present in a myriad of tasks. For example, STS is the cornerstone of paraphrase identification (Zhou et al., 2022), it is widely used for text classification and clustering tasks (Minaee et al., 2021), it lays the foundation for popular evaluation metrics, such as BERTScore (Zhang et al., 2020) and, more recently, it has enabled the development of Dense Passage Retrieval (DPR) (Karpukhin et al., 2020), as well as some other highly active areas of research such as Retrieval-Augmented Generation (RAG) (Lewis et al., 2020), which has also attracted great attention from the industry.

However, most frequently, a single similarity score is reported (e.g., cosine similarity), and condensing all the semantic nuances into this score hinders interpretability. Popular techniques in the field of Explainable AI (XAI) revolve around natural language explanations (also called rationales, Gurrapu et al., 2023), or visualizations. These also fit well within the context of STS, and in the present work, we focus on providing explanations through textual highlighting. More concretely, we want to identify span pairs that differ in meaning, a task we call Dissimilar Span Detection (DSD).

More formally, given two sequences of word or sub-word tokens $X = (x_1, x_2, ..., x_n)$ and $Y = (y_1, y_2, ..., y_m)$, let $s_j := ((x_a, x_{a+c}), (y_b, y_{b+d}))$ be a token span pair coming from X and Y, with $1 \leq a < n$, $1 \leq b < m$, $1 \leq c \leq n - a$ and $1 \leq d \leq m - b$, the goal is to find all non-overlapping span pairs s_j that, having a common semantic function, differ in meaning. Note that a text pair might also contain no such spans.

As seen in Figure 1, the annotations can directly be shown to users in addition to the overall textual similarity, as well as be used to support other downstream tasks such as: Fake News Detection, in order to detect key mismatching information; Annotation Error Detection (Klie et al., 2023), e.g., to detect errors in STS-related datasets; Text Clustering or Paraphrase Identification, to prevent clustering together or flagging as paraphrase texts that contain small but critical differences; finally, when taken to a multilingual setting, the task of DSD is very similar to that of Critical Error Detection (Zerva et al., 2022), i.e., predicting whether a translation contains a critical error or not, and could also assist in Parallel Corpus Mining, to avoid matching similar texts that contain key discrepancies.

Our main contributions are:

- We introduce the task of Dissimilar Span Detection (DSD): a method that provides interpretability in the context of STS which involves identifying span pairs that differ in meaning. This method can also improve performance in various downstream, STS-dependent tasks.
- We propose a semi-automatic pipeline to create suitable training and evaluation data for the task. A dataset produced in such way, the

[†] Currently affiliated to ELLIS Alicante.

Proceedings of the Fifteenth Language Resources and Evaluation Conference (LREC 2026), pages 2190–2210
11–16 May 2026.

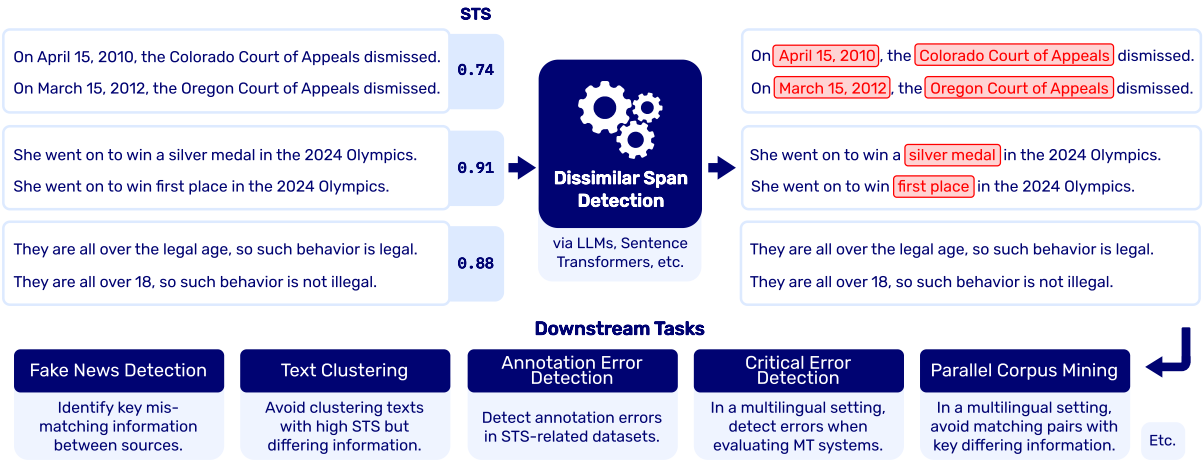

Figure 1: Examples of the *Dissimilar Span Detection* task, i.e., given a pair of texts, identify which spans differ semantically. Here, we show two pairs that contain dissimilar spans and one pair that does not. Cosine similarities are calculated with the Sentence Transformer model `all-MiniLM-L6-v2`. Note that pairs containing dissimilar spans might still yield a high semantic textual similarity, sometimes even higher than sentences that are equivalent in meaning. Therefore, relying on STS alone could potentially lead to erroneous decisions in downstream tasks.

so-called Span Similarity Dataset (SSD), is publicly released, along with a comprehensive evaluation setup. The dataset's correctness is validated through the manual annotation of a subset by 6 different annotators, which we compare against the semi-automatically annotated data.

- We evaluate a number of baseline methods to tackle the task. In particular, two unsupervised, model-agnostic methods based on LIME (Ribeiro et al., 2016) and SHAP (Lundberg and Lee, 2017) are evaluated. Inspired by the latter, we propose a further method that yields better results. We also consider state-of-the-art LLMs, which perform best overall, and evaluate different supervised models fine-tuned on the task, obtaining the best compromise between size and performance. Nevertheless, results stay generally low, highlighting the difficulty of DSD even in our simplified setting. Finally, we apply DSD to the task of paraphrase detection, showing its viability to improve performance in downstream tasks.

The source code and the dataset are publicly available at `https://dmlls.github.io/dissimilar-span-detection`, licensed under GPL v3.0 and CC BY-SA 4.0, respectively.

2. Related Work

The task of annotating tokens or spans of text to provide explanations has already been worked on, albeit not extensively, in the context of Natural Language Inference (NLI). A popular dataset in this regard is e-SNLI (Camburu et al., 2018), which extends the Stanford Natural Language Inference (SNLI) corpus (Bowman et al., 2015) by highlighting the words most relevant for a particular entailment label, and providing natural language explanations for these annotations.

Thorne et al. (2019) explored different methods, namely LIME and Anchor Explanations (Ribeiro et al., 2018), as well as their own method, based on Multiple Instance Learning (MIL), in order to generate token-level explanations that justify the NLI labels. A continuation in this direction is Kim et al. (2020), which expands previous work by annotating not only the tokens/spans relevant for the entailment prediction, but also classifying the detected spans into different categories.

Moving more concretely to the topic of explainability in the context of STS, (Rus et al., 2012) released the SIMILAR corpus in an effort to improve the explainability of word-to-word similarity metrics. It contains 700 manually annotated sentences and considers different types of semantic relations between words.

Still working at the word level, Malkiel et al. (2022) introduce an unsupervised technique named BTI (BERT Interpretations) that employs BERT-like models to explain similarities between paragraphs. In this case, individual, matching words that best explain the similarity between pairs of texts are identified. The selected word pairs can then be visualized by the user, helping to establish connections between the two texts.

One of the works most closely related to ours is Lopez-Gazpio et al. (2017), which deals with the concept of *Interpretable Semantic Textual Similarity*, and is framed within the Task 2 of both

Sentence 1	Sentence 2	Span Similarity	Sentence Similarity
It only utilizes {{a few refineries}} in the Northeast.	It only utilizes {{numerous facilities}} in the Northeast.	0	0
It was {{restored}} in the {{1980s}}.	It was {{destroyed}} in the {{eighties}}.	0,1	0
Thank you for {{wasting my money}}.	Thank you for {{misusing my funds}}.	1	1
There are depots at {{Quilpie}} and {{Roma}}.	There are depots at {{Brisbane}} and {{Sydney}}.	0,0	0

Table 1: Examples of shorter sentences from the SSD. Spans are denoted using double curly braces, and then annotated respectively with a 0, in case the span pair differs in meaning, or 1, if they are equivalent. If there exist several annotated spans, these values are separated by a comma. Finally, the sentence similarity column denotes whether the meaning of the sentence pair as a whole is equivalent in meaning (1) or not (0). We have highlighted similar (green) and dissimilar spans (red) for better readability.

SemEval-2015 (Agirre et al., 2015) and SemEval-2016 (Agirre et al., 2016). The authors formalized the interpretability layer as "the alignment between pairs of segments[1] across the two sentences, where the relation between the segments is labeled with a relation type and a similarity score." Granular span labels are defined, such as: `EQUI`, for semantically *equivalent* spans; `SIM`, for *similar* spans; or `OPPO`, for spans with *opposite* meanings. Each of these mutually exclusive labels is also accompanied by a number ranging from 1 to 5, indicating the level of similarity or equivalence. Apart from proposing different methods to solve the task, the authors also implemented a user-facing verbalization system. The dataset they used had been previously released for Task 2 of SemEval-2016 (Agirre et al., 2016), which we also use as an additional resource to evaluate the different baseline methods we consider.

However, to the best of our knowledge, there has been no prior research addressing the utility of detecting semantic *dissimilarities* at the span level. This focus on dissimilarity can be useful not only to provide explanations to end users—an effective way to explain similarity, or the lack thereof, is by pointing out the dissimilarities—but also to improve the performance of systems in downstream tasks that rely on STS.

3. Span Similarity Dataset (SSD)

In this section, we discuss the motivation behind building a new dataset for the task and describe how it was constructed. We also conduct a dataset analysis, reporting several statistics about it.

[1] Segments are what we refer to as *spans*.

3.1. Dataset Motivation

We initially considered NLI-related datasets like e-SNLI. However, pairs labeled as *contradiction* (i.e., those that contain dissimilarities) are often so different that the entire sentence would need to be labeled as dissimilar.[2] Furthermore, the tasks of NLI and STS, although related, are fundamentally different. It is common for NLI datasets to contain sentences that may be logically related in ways that do not necessarily reflect their semantic similarity. As a result, employing these datasets for evaluating DSD would not have been reliable.

On the other hand, we also considered the aforementioned dataset from SemEval-2016, which is specifically crafted for interpretable STS. Unfortunately, it was still too restricted. Even though this dataset annotates different relations between spans, in our case we were only able to use those labeled as *opposite* (`OPPO`), since all other categories include a mix of similar and dissimilar spans (or no dissimilar spans at all). For example, spans such as "a woman" and "a man", or "17 civilians" and "10 police" were annotated as *similar*, when in our context these pairs are considered dissimilar. Consequently, out of 2,929 sentence pairs, there were only 109 relevant to our context.

Due to the scarcity of suitable data, we found it necessary to create our own dataset, the so-called Span Similarity Dataset (SSD). It currently contains 1,000 annotated samples that specifically target the task of DSD. As previously mentioned, it has been publicly released for the benefit of the NLP community.

[2] In fact, the mean cosine similarity of all the *contradiction* pairs in the e-SNLI dataset (considering the *train*, *dev*, and *test* splits) is as low as 0.31, with a standard deviation of 0.20 (calculated using the Sentence Transformer model `all-mpnet-base-v2`).

3.2. Dataset Construction

The goal of our dataset is to provide a first, simplified DSD benchmark to assess where state-of-the-art models, such as LLMs, stand in relation to this task. Thus, we limit ourselves to English and stay at the sentence level.

The sentences of the SSD were sourced from a random subset of the samples from the CANNOT Dataset (Anschütz et al., 2023), which in turn is a compilation of data from different datasets targeting tasks closely related to DSD, such as fact-checking (Sathe et al., 2020), paraphrase detection (Vahtola et al., 2022), or negation identification (Truong et al., 2022).

The main steps involved in the annotation were:

1. Taking the first sentence and altering one or more spans of words, giving result to the second sentence. The modified spans could either be equivalent in meaning to the original one, or be semantically dissimilar.
2. Enclosing each of the altered spans between span annotation markers. In our case, {{ denotes the beginning of a span, and }} its end.
3. Annotating each of the span pairs with either a 1, if they are equivalent in meaning, or 0 otherwise.
4. Annotating whether the entire two sentences are semantically equivalent (1) or not (0).

The annotation was performed in a semi-automatic way through the use of an LLM[3] via a manually engineered prompt, significantly reducing time and effort by allowing the model to replace spans and assign labels. Nevertheless, since the model was unable to consistently annotate the correct labels, we decided to perform the span labeling manually. Finally, each of the automatically generated spans was manually reviewed, and if needed, corrected, before being added to the dataset. For reference, annotating 100 samples this way takes between 1 and 2 hours. An example of the resultant samples can be seen in Table 1.

3.3. Dataset Analysis

The main statistics of the SSD are collected in Table 2. Furthermore, we perform POS tagging on the dataset, obtaining the most frequent POS labels enclosed within spans: nouns (38.12 %), proper nouns (21.72 %), verbs (16.44 %), and adjectives (13.44 %). As for grammatical dependencies, the most common are: objects of a preposition – *pobj* (23.03 %), compound entities (15.72 %), and adjectival modifiers – *amod* (10.69 %).

[3] ChatGPT (OpenAI, 2023) with the GPT-3.5-Turbo backend was employed between November 2023 and April 2024.

Attribute	Value
Sentence pairs	1,000
Span pairs	1,296
Dissimilar span pairs	648 (50 %)
Equivalent span pairs	648 (50 %)
Dissimilar sentence pairs	571 (57 %)
Equivalent sentence pairs	429 (43 %)
Mean sentence word length	10.75 (4.18)
Mean span word length	3.91 (2.71)
Mean spans per sentence	1.30 (0.51)

Table 2: Global statistics of the SSD. For the word counts, simple whitespace tokenization was used, not considering punctuation.

Agreement	Inter-annotator	Annotator-dataset
Span boundaries	0.74 (0.05)	0.67 (0.03)
Span labels	0.87 (0.05)	0.91 (0.04)
Span count	0.66 (0.06)	0.61 (0.07)

Table 3: Mean Cohen's κ scores on different annotation aspects (standard deviations in parenthesis).

To ensure the correctness of the semi-automatically annotated data, we ask 6 different annotators with no previous contact with the task to annotate a subset of 100 random samples from the SSD. We then compute all combinations of paired inter-annotator agreements, and calculate their mean and standard deviation. We also report the mean agreement between each of the annotators and the gold samples proceeding from the SSD. The results, included in Table 3, show a substantial level of agreement, indicating that the task is well defined. After performing an error analysis, we detect that the slightly lower agreement on the span counts is likely derived from spans involving conjunctions, such as expressions connected by "and" either being included in the same span, or split into two different spans.

4. Experimental Setup

Next, we present and explain the different strategies we considered to tackle the problem of DSD. We also introduce the evaluation schema adopted to assess their performance.

4.1. Methods

We propose a total of 5 methods plus 2 baselines. Some of these methods have the advantage of working with smaller models and requiring no fine-tuning at all, and most are, to a significant extent, model agnostic. We also consider two dummy baselines that require no model.

SHAP-DSD This approach is powered by the use of SHAP (SHapley Additive exPlanations). SHAP is a popular framework that allows users to obtain explanations of individual model predictions. It is based on game theory principles, building on the concept of Shapley values (Shapley, 1953), where the central idea, when applied to Machine Learning, is to fairly distribute an importance value among a set of input features that contribute to the model's prediction.

SHAP's `Explainer` class is the central "explainer" within the framework. In our case, instead of initializing it directly with a model, we pass a callable (function) *f* that takes in a pair of strings and returns the *dissimilarity score* between their embeddings, calculated as:

$$\text{DissimilarityScore}(a, b) = 1 - \text{CosineSimilarity}(a, b) \quad (1)$$

There exist a myriad of models we can use to encode the text. We decided to choose models within the Sentence Transformers family (Reimers and Gurevych, 2019), since they are suited to work with sentences, matching the nature of our dataset. Nonetheless, the advantage of basing our explanations solely on the dissimilarity score is that we can plug in *any* embedding model, making this method very flexible and easy to update as new advancements in the field are made.

Given the two sentences, we run the explainer and obtain the SHAP values for each of the tokens in the second sentence. If the SHAP value (i.e., dissimilarity value) for a token is above a certain dissimilarity threshold,[4] we consider that token to be dissimilar. Contiguous tokens classified as dissimilar are then included in the same span.

The caveat of this method is that the annotations will only be as good as the model's sensitivity to semantic dissimilarity. This is the case with all the unsupervised methods we propose.

LIME-DSD The core idea behind LIME (*Local Interpretable Model-agnostic Explanations*) is to locally explain a black-box model through a simpler, transparent model that can be interpreted by humans. *Locally* here means that the simpler surrogate model can only provide explanations on the vicinity of the instance being predicted. It achieves this by training the surrogate model on perturbated versions of the instance being explained. In the case of textual data, the perturbations consist in dropping or masking words or characters in the input text.

We proceed in a similar fashion as with SHAP-DSD, using also Sentence Transformer models. Instead of SHAP values, we now work with the explanation weights predicted by LIME for each token. We then enclose within spans those tokens with weights above a threshold.

Embedding-DSD This method constitutes a novel contribution. It bears close similarities to SHAP or LIME in that it aims to explain individual samples by applying certain perturbations on them. However, in our case, we intentionally aim to develop a method that, despite being less generally applicable, works better within the specific context of DSD. For that, instead of relying on masking, or even entirely dropping tokens, our approach considers all possible *n*-gram replacements that can be made from the first sentence to the second sentence, and then calculates the impact each of these replacements has on the overall similarity of the input pair.

Let us illustrate with an example why this might be a more effective perturbation when dealing with DSD. Imagine we are given two sentences that we want to compare:

- `the bird flies fast over the hill`
- `the car rides fast over the hill`

First, we calculate the *base similarity* between the sentences, which is simply their cosine similarity (or any other similarity measure). In our example, let us say that this base similarity is 0.6. We then compute all the unigrams in the first sentence, followed by all the bigrams, trigrams, etc., until we reach an *n*-gram that spans the entirety of the sentence, i.e., when *n* is equal to the number of unigrams in it. Next, we replace *all* the obtained *n*-grams from the previous step in *all* possible positions in the second sentence. If, for example, we were considering the trigram `['the', 'bird', 'flies']` from the first sentence, we would obtain the replacements in the second sentence depicted in Figure 2.

```
1  the bird flies fast over the hill
2  the the bird flies over the hill
3  the car the bird flies the hill
4  the car rides the bird flies hill
5  the car rides fast the bird flies
```

Figure 2: Example of trigram replacements. The considered trigrams proceeding from the first sentence (highlighted) are inserted into the second sentence, replacing the original trigrams.

We then calculate the similarity between each of the replacements and the first, original sentence, and identify which one of the replacements

[4] This threshold is dependent on the underlying model employed.

achieves the highest similarity gain compared to the initial base similarity. Coming back to our example, we can see that the first replacement is identical to the first original sentence, resulting in a similarity of 1.0. Its similarity gain is then 0.4, calculated as 1.0 - 0.6 (base similarity). We then associate this value with each unigram in the trigram (i.e., `'the'`, `'bird'`, `'flies'`). By the end of the iterations, each unigram in the second sentence will have a series of similarity gains from all *n*-gram replacements that include it.

The similarity gains associated to each of the unigrams are combined through an *aggregation function*. In our case, we define a function that assigns smaller weights to gains coming from higher *n*-grams, in order to pay more attention to local differences:

$$AggGain_{unigram} = \frac{1}{n} \cdot \sum_{i=1}^{n} \frac{gains_i}{i}\,, \qquad (2)$$

where $gains$ is an ordered array containing the gains coming from the unigrams, bigrams, ..., *n*-grams in which the unigram appears, and n is the length of this array. This function will assign less weight to gains corresponding to higher *n*-grams (i.e., those that appear later in the $gains$ array). We normalize the aggregated gains by n, since unigrams at the beginning and end of the sentence will appear in less *n*-grams, and therefore will have a smaller amount of gain values associated to them.

Once we have a final, aggregated gain for each unigram, we apply a threshold and those unigrams with gains above it are considered dissimilar, grouping contiguous, dissimilar unigrams in the same span. Appendix C includes a pseudocode version of our algorithm.

This method, once again, relies on a good embedding model to work optimally. In this case, we employed not only Sentence Transformer models, but also commercially available embedding models.[5] This allowed us to draw a comparison between small (former) and large embedding models (latter), and determine whether the performance gain, or lack thereof, is worth the extra cost.

LLM-DSD Recently, the NLP field has experienced notable advancements due to the rapid development of Large Language Models (LLMs). These models have shown outstanding performance in zero-shot or few-shot tasks, so they constituted an evident approach for our task.

When working with LLMs, a carefully designed prompt is crucial to obtain good results, especially when working in zero-shot or few-shot scenarios (Zhao et al., 2023), which is also our case. We formulate our prompt as a 4-shot learning task, providing 2 examples of correct annotations, and 2 examples of incorrect annotations. We employed models from a variety of vendors, namely OpenAI, Meta, Anthropic and DeepSeek.

Token-Classification-DSD This method approaches the problem as a token classification task (also known as sequence labeling), similarly to how Part-of-speech tagging or Named Entity Recognition (NER) are commonly dealt with. In order to transform the data into a suitable format for token classification, we tokenize the training input samples and annotate them following the BIO tagging schema (Ramshaw and Marcus, 1995): the first token within a span is tagged with a "B", the rest of the tokens within the span are tagged with an "I", and any other token (i.e., tokens outside any span) are labeled with an "O". It is important to note that we only attend to spans that are dissimilar, that is, those labeled as 0 in the SSD. Annotated span markers for equivalent spans are removed, and we do not label them in any other way.

Instead of training from scratch, we fine-tuned BERT and RoBERTa (Liu et al., 2019) models, as well as their distilled counterparts, DistilBERT and DistilRoBERTa (Sanh et al., 2019), on the SSD. We use cross-entropy loss, with a learning rate $\eta = 5 \cdot 10^{-5}$, a weight decay of $5 \cdot 10^{-3}$, and a batch size of 8. We adopt a 5-fold cross-validation setting, training the models for 5 epochs on 4 of the folds and evaluating the fine-tuned model on the remaining fold.

Baseline Methods In addition to the previous methods, we propose two simple baselines, No-DSD and Naive-DSD, exploiting certain peculiarities of the SSD. These two techniques have no notion of meaning and are very fast to compute.

No-DSD Here, we just return the second sentence as it is, with no dissimilar spans annotated. This is presumably a strong baseline due to the fact that roughly half of the sentence pairs in the SSD contain no dissimilar spans. Therefore, this method will always annotate such pairs accurately. On the other hand, no pair actually containing dissimilar spans will get correctly annotated.

Naive-DSD In this case, any word in the second sentence that does not appear in the first sentence is considered as dissimilar. This method will fail for those equivalent spans that convey the same meaning but employ a different wording. However, annotations will be fairly accurate for spans that are indeed dissimilar.[6]

4.2. Evaluation

We evaluate the different methods both on our dataset, as well as the subset of samples from the

[5] Models from OpenAI and Google were used.

[6] Note that dissimilar span pairs might still contain common words, such as articles or prepositions.

Method	Model Size	SSD				SemEval-2016	
		F1-Global	F1-NoDiff	F1-Diff	Eval. Time	F1-Diff	Eval. Time
LIME `all-mpnet-base-v2` (0.001)	109M	0.463	0.782	0.223	1981.81	0.109	199.89
SHAP `all-MiniLM-L6-v2` (0.010)	22.7M	0.366	0.434	0.131	44.47	0.306	4.52
Embedding `all-mpnet-base-v2` (0.006)	109M	0.469	0.529	0.423	11.28	0.352	1.13
Embedding `text-embedding-004` (0.005)	Billions	0.547	0.666	0.459	611.42	0.338	34.57
LLM `Claude 3.5 Sonnet`	Billions	**0.750**	**0.895**	**0.640**	337.80	0.389	39.22
Token Classification `roberta-base`	125M	0.690	0.838	0.574	**1.23**	**0.441**	**0.10**
No-DSD	N/A	0.429	1.000	0.000	N/A	0.000	N/A
Naive-DSD	N/A	0.311	0.003	0.542	N/A	0.307	N/A

Table 4: Summary of the results. LLMs and supervised methods (Token-Classification-DSD) perform best, followed by our proposed method (Embedding-DSD). For methods that require a threshold, we include it next to the model's name. Evaluation times are reported in minutes. For experiments relying on external APIs (`text-embedding-004` and Claude 3.5 Sonnet), times might vary depending on rate limits, connection speeds, etc. For these models, the exact number of parameters remains undisclosed, so we are unable to report it. The rest of experiments were run on a NVIDIA RTX 3080Ti graphics card.

SemEval-2016 Task 2 dataset with spans annotated as *opposite*. For a more robust evaluation given the relatively small size of the datasets, we adopt a 5-fold cross-validation scheme. For the unsupervised methods, we simply evaluate on the test fold, leaving the remaining 4 folds unused.

Considering that the number of annotated spans can be different from the number of reference spans, we first perform *span alignment*. For that, given that span pairs in the SSD are ordered, we align the reference and annotated spans in a way that the offset between them is minimized. This offset is calculated as follows:

$$\text{Offset}(s_{ref}, s_{ann}) = |s_{ref}.\text{start} - s_{ann}.\text{start}| + |s_{ref}.\text{end} - s_{ann}.\text{end}|, \quad (3)$$

where s_{ref} and s_{ann} are the reference and annotated spans, respectively, and *start* and *end* denote the starting and ending indexes of the span. Since the number of spans per sentence is low, we compute all the possible alignments and choose the one that results in the smallest offset.

For each of the aligned spans, we calculate, at the unigram level, precision, recall, and F1-score in a similar fashion to the METEOR metric (Denkowski and Lavie, 2014). Scores are averaged across all spans in the input pair, giving result to the overall precision, recall and F1-score for that sample. Given that we only get annotations for the second sentence, we then swap them to also evaluate the annotation of the first sentence.

4.2.1. Downstream Task

Additionally, we set up an experiment employing DSD for the task of paraphrase detection. This serves as a plausibility check to ensure the correctness and suitability of DSD in a real-world scenario. The evaluation is performed on the test split of the PAWS-Wiki Labeled (Final) Corpus (Zhang et al., 2019), containing pairs of sentences labeled with either 0 (no paraphrase) or 1 (paraphrase).

First, we use a Sentence Transformer model to perform STS. In order to adapt the model to the task, we finetune it on the training split of the PAWS-Wiki Labeled dataset. At inference time, we label a pair as a paraphrase if the similarity is equal or greater than a specific threshold. We find the optimal threshold for each model by evaluating on the validation set of the dataset. Once found, we run a final evaluation on the test split.

Then, we repeat the experiment, but only label a pair as paraphrase if the pre-established similarity threshold is reached *and* the pair contains no dissimilar spans. These spans are obtained by applying Embedding-DSD on the same model used for STS.

In both setups we calculate the accuracy as the number of correctly classified pairs divided by the total number of pairs. The goal is to determine whether introducing DSD leads to increased accuracies in paraphrase identification.

5. Results and Discussion

In the case of the SSD, in order to account for false positives (i.e., spans labeled as dissimilar when they are not), we report separate metrics for those sentence-pairs that contain no dissimilar spans (*NoDiff*), and those that do (*Diff*). This distinction aims to determine how often evaluated methods treat semantically equivalent spans with lexical differences as dissimilar. We also report results taking into consideration both the *Diff* and *NoDiff* scores (*Global*). As for the SemEval-2016 data, we only report the *Diff* scores, since we only consider samples with dissimilar spans.

Table 4 collects the best results from each of the presented methods. LLMs and supervised methods (i.e., token classification) obtain the highest scores overall. However, the latter is much faster on average, employing models orders of magnitude smaller (ranging around 100 million parameters), at the cost of requiring to be fine-tuned on the task. Our proposed method, Embedding-DSD, achieves better results than the other two considered explainability methods, based on LIME and SHAP, while having considerably lower inference times.

The results for the SemEval-2016 data must be taken as a lower bound, since we only considered spans labeled as *opposite*. As mentioned earlier, other types of spans still remain that are in fact dissimilar, leading to lower scores when compared to the STS results.

Overall, these results prove the difficulty of the task. Specifically, methods that rely on embedding models show the general lack of sensitivity of such models towards localized textual changes. In Appendix B, we include examples of annotations produced by these and the other methods. The annotation schema introduced in this work and used for the SSD could be used to fine-tune STS models that are more sensitive to small lexical variations. This is something that has been proven to work in the case of negations (Anschütz et al., 2023).

Table 5 includes the results from two different models when applying DSD to the concrete task of paraphrase detection. Here, the introduction of DSD improves accuracy up to 8 points. It is worth noting that this improvement is achieved without any modifications to the base model or further fine-tuning on the task.

To conclude this discussion, we would like to mention that, although not the main focus of this work, DSD could also be used to evaluate models' sensitivity to dissimilarity. This evaluation could serve as a complement to STS metrics, or be used as an objective when training or finetuning models for STS related tasks, potentially leading to models that capture semantics in a more complete way.

Model	**STS**	**STS+DSD**
`all-MiniLM-L6-v2` $(0.65, 8{\cdot}10^{-3})$	0.720	**0.808**
`all-mpnet-base-v2` $(0.63, 8{\cdot}10^{-3})$	0.795	**0.868**

Table 5: Comparison of accuracies on the PAWS-Wiki Labeled using uniquely STS or combining STS with DSD. We include the STS and DSD thresholds in parenthesis next to the models.

6. Conclusion

In this work, we present the task of Dissimilar Span Detection (DSD): a method to improve the interpretability and reliability of STS scores. The task consists in, given two texts, identifying spans pairs with a common semantic function, but conveying different meanings. DSD can complement current STS metrics, which typically report a single score, providing interpretability in user-facing scenarios. Additionally, it can potentially enhance the performance of current approaches that rely on STS.

To lay the foundation for the task, we introduce an appropriate annotation and evaluation schema and release the Span Similarity Dataset (SSD). This dataset, developed via a semi-automated pipeline that integrates LLMs with human verification, serves as a valuable starting point for the research community, facilitating further advancements in the field.

We propose and evaluate a number of methods, both unsupervised and model-agnostic, based on LIME, SHAP, LLMs, and our own method, as well as a further supervised approach. These approaches are conceived as a first attempt to tackle the task of DSD, and should therefore be taken as baselines. Regardless, the obtained results reflect the challenging nature of the task. However, even in its current state, DSD can already increase performance in the specific task of paraphrase detection. Nevertheless, there is still ample room for refinement in future work. Immediate improvements can be applied to a number of different fronts: the SSD dataset can be expanded with further samples; approaches that build on the presented methods, or other new approaches, can be explored; finally, DSD can be further integrated and tested on different downstream tasks apart from paraphrase detection.

7. Limitations

The main limitations of our work stem from aspects regarding the annotation of the SSD, namely:

- It currently contains data solely in English. In future work, a multilingual setting could be considered. Still, all the methods presented here would be applicable with no or minor modifications.
- We worked exclusively at the sentence level. An even more challenging setup would work with full paragraphs, or even longer textual units. A baseline approach would consist in 1) splitting the text into sentences, 2) performing sentence alignment, so that the most similar sentences are paired together, and 3) running any of our developed methods.
- In the dataset, corresponding spans always appear in the same order in both the first and the second sentence. In future work, a more complete setup would allow span pairs to come in any order.
- Our methods focused solely on DSD, but a full-fledged system would also be able to identify equivalent spans. Furthermore, it could also be possible to include spans that only appear in one of the sentences. This setting would enable support for further tasks, such as hallucination detection.

These limitations are a byproduct of the approach we adopted towards the task of DSD: establish a common framework, but start with a more restricted and simplified setting in order to lay solid foundations that can be expanded in future work.

Ethical Considerations

The main goal of our work is to provide techniques and foster new ideas that help improve the interpretability and reliability of present STS metrics. We are currently unable to envision ways in which our work can be maliciously misused, although we welcome any suggestion and commit to continuously reviewing our methodologies in order to ensure a safe, future development of the task.

We are aware, however, that the presented methods still perform at an unacceptable level for any kind of real-world deployment, and strongly advocate against them being used in production until further research has been conducted and performance and reliability have been improved.

Finally, since the SSD dataset has been crafted from an amalgam of previously released datasets, it may contain biases already present in them. If any of these biases became evident at any point in the future, we remain fully committed to reviewing and rectifying our dataset.

Acknowledgments

The work was funded/co-funded by the European Union (ERC, EPICAL, 101141712). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.

Besides, we would like to thank the six annotators that provided their invaluable time to contribute to this work.

8. Bibliographical References

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Iñigo Lopez-Gazpio, Montse Maritxalar, Rada Mihalcea, German Rigau, Larraitz Uria, and Janyce Wiebe. 2015. SemEval-2015 task 2: Semantic textual similarity, English, Spanish and pilot on interpretability. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 252–263, Denver, Colorado. Association for Computational Linguistics.

Eneko Agirre, Aitor Gonzalez-Agirre, Iñigo Lopez-Gazpio, Montse Maritxalar, German Rigau, and Larraitz Uria. 2016. SemEval-2016 task 2: Interpretable semantic textual similarity. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 512–524, San Diego, California. Association for Computational Linguistics.

Miriam Anschütz, Diego Miguel Lozano, and Georg Groh. 2023. This is not correct! Negation-aware Evaluation of Language Generation Systems. In *Proceedings of the 16th International Natural Language Generation Conference*, pages 163–175, Prague, Czechia. Association for Computational Linguistics.

Michael Denkowski and Alon Lavie. 2014. Meteor Universal: Language Specific Translation Evaluation for Any Target Language. In *Proceedings of the EACL 2014 Workshop on Statistical Machine Translation*.

Sai Gurrapu, Ajay Kulkarni, Lifu Huang, Ismini Lourentzou, and Feras A. Batarseh. 2023. Rationalization for explainable NLP: a survey. *Frontiers in Artificial Intelligence*, 6.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense Passage

Retrieval for Open-Domain Question Answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online. Association for Computational Linguistics.

Youngwoo Kim, Myungha Jang, and James Allan. 2020. Explaining Text Matching on Neural Natural Language Inference. *ACM Trans. Inf. Syst.*, 38(4).

Jan-Christoph Klie, Bonnie Webber, and Iryna Gurevych. 2023. Annotation Error Detection: Analyzing the Past and Present for a More Coherent Future. *Computational Linguistics*, 49(1):157–198.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA. Curran Associates Inc.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *ArXiv*, abs/1907.11692.

I. Lopez-Gazpio, M. Maritxalar, A. Gonzalez-Agirre, G. Rigau, L. Uria, and E. Agirre. 2017. Interpretable semantic textual similarity: Finding and explaining differences between sentences. *Knowledge-Based Systems*, 119:186–199.

Scott M Lundberg and Su-In Lee. 2017. A Unified Approach to Interpreting Model Predictions. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4765–4774. Curran Associates, Inc.

Itzik Malkiel, Dvir Ginzburg, Oren Barkan, Avi Caciularu, Jonathan Weill, and Noam Koenigstein. 2022. Interpreting BERT-based Text Similarity via Activation and Saliency Maps. In *Proceedings of the ACM Web Conference 2022*, WWW '22, page 3259–3268, New York, NY, USA. Association for Computing Machinery.

Shervin Minaee, Nal Kalchbrenner, Erik Cambria, Narjes Nikzad, Meysam Chenaghlu, and Jianfeng Gao. 2021. Deep Learning–based Text Classification: A Comprehensive Review. *ACM Comput. Surv.*, 54(3).

OpenAI. 2023. ChatGPT.

Lance Ramshaw and Mitch Marcus. 1995. Text Chunking using Transformation-Based Learning. In *Third Workshop on Very Large Corpora*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 1135–1144.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2018. Anchors: high-precision model-agnostic explanations. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'18/IAAI'18/EAAI'18. AAAI Press.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *CoRR*, abs/1910.01108.

Aalok Sathe, Salar Ather, Tuan Manh Le, Nathan Perry, and Joonsuk Park. 2020. Automated fact-checking of claims from Wikipedia. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 6874–6882, Marseille, France. European Language Resources Association.

L. S. Shapley. 1953. *17. A Value for n-Person Games*, pages 307–318. Princeton University Press, Princeton.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2019. Generating Token-Level Explanations for Natural Language Inference. 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)*, pages 963–969, Minneapolis, Minnesota. Association for Computational Linguistics.

Thinh Hung Truong, Yulia Otmakhova, Timothy Baldwin, Trevor Cohn, Jey Han Lau, and Karin Verspoor. 2022. Not another negation benchmark: The NaN-NLI test suite for sub-clausal

negation. In *Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 883–894, Online only. Association for Computational Linguistics.

Teemu Vahtola, Mathias Creutz, and Jörg Tiedemann. 2022. It is not easy to detect paraphrases: Analysing semantic similarity with antonyms and negation using the new SemAntoNeg benchmark. In *Proceedings of the Fifth BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 249–262, Abu Dhabi, United Arab Emirates (Hybrid). Association for Computational Linguistics.

Chrysoula Zerva, Frédéric Blain, Ricardo Rei, Piyawat Lertvittayakumjorn, José G. C. de Souza, Steffen Eger, Diptesh Kanojia, Duarte Alves, Constantin Orăsan, Marina Fomicheva, André F. T. Martins, and Lucia Specia. 2022. Findings of the WMT 2022 Shared Task on Quality Estimation. In *Proceedings of the Seventh Conference on Machine Translation (WMT)*, pages 69–99, Abu Dhabi, United Arab Emirates (Hybrid). Association for Computational Linguistics.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. BERTScore: Evaluating Text Generation with BERT. In *International Conference on Learning Representations*.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. 2023. A Survey of Large Language Models.

Chao Zhou, Cheng Qiu, and Daniel E. Acuna. 2022. Paraphrase Identification with Deep Learning: A Review of Datasets and Methods.

9. Language Resource References

Agirre, Eneko and Gonzalez-Agirre, Aitor and Lopez-Gazpio, Iñigo and Maritxalar, Montse and Rigau, German and Uria, Larraitz. 2016. *SemEval-2016 Task 2: Interpretable Semantic Textual Similarity*. Association for Computational Linguistics.

Miriam Anschütz, Diego Miguel Lozano, and Georg Groh. 2023. This is not correct! Negation-aware Evaluation of Language Generation Systems. In *Proceedings of the 16th International Natural Language Generation Conference*, pages 163–175, Prague, Czechia. Association for Computational Linguistics.

Bowman, Samuel R. and Angeli, Gabor and Potts, Christopher and Manning, Christopher D. 2015. *A large annotated corpus for learning natural language inference*. Association for Computational Linguistics.

Camburu, Oana-Maria and Rocktäschel, Tim and Lukasiewicz, Thomas and Blunsom, Phil. 2018. *e-SNLI: Natural Language Inference with Natural Language Explanations*. Curran Associates, Inc.

Vasile Rus and Mihai C. Lintean and Cristian Moldovan and William Baggett and Nobal B. Niraula and Brent and Morgan. 2012. *The SIMILAR Corpus: A Resource To Foster The Qualitative Understanding of Semantic Similarity of Texts*. European Language Resources Association (ELRA).

Zhang, Yuan and Baldridge, Jason and He, Luheng. 2019. *PAWS: Paraphrase Adversaries from Word Scrambling*. Google AI Language. Proc. of NAACL.

A. Detailed Results

Table 6 and Table 7 include a detailed collection of the results obtained by each of the different proposed DSD methods.

For the SSD, we report precision, recall and F1 scores evaluated on both sentence-pairs containing no differing spans (i.e., *NoDiff*), as well as on pairs including such spans (i.e., *Diff*). We also report the global performance considering both scenarios (i.e., *Global*).

For the SemEval-2016 Task 2 data, we only report the *NoDiff* scores, since we only consider sentences that contain *opposite* spans.

Method	Model	NoDiff			Time (minutes)
		Precision	Recall	F1	
LIME	`all-MiniLM-L6-v2` (0.010)	**0.253** (0.010)	**0.395** (0.029)	**0.269** (0.013)	**120.88**
	`all-mpnet-base-v2` (0.001)	0.154 (0.017)	0.093 (0.011)	0.109 (0.011)	199.89
SHAP	`all-MiniLM-L6-v2` (0.005)	**0.247** (0.012)	**0.529** (0.025)	**0.306** (0.016)	**4.52**
	`all-mpnet-base-v2` (0.001)	0.243 (0.016)	0.480 (0.022)	0.293 (0.019)	21.58
Embedding	`all-MiniLM-L6-v2` (0.005)	**0.249** (0.015)	**0.842** (0.051)	**0.362** (0.019)	**0.38**
	`all-mpnet-base-v2` (0.005)	0.243 (0.013)	0.823 (0.064)	0.352 (0.017)	1.13
	`text-embedding-3-large` (0.005)	0.248 (0.015)	0.829 (0.056)	0.355 (0.020)	11.16
	`text-embedding-004` (0.005)	0.239 (0.012)	0.778 (0.007)	0.338 (0.018)	34.57
LLM	GPT-4o `2024-11-20`	0.376 (0.031)	**0.609** (0.033)	**0.416** (0.031)	45.22
	Llama 3.3 70B Instruct `2024-12-06`	**0.409** (0.045)	0.530 (0.054)	0.413 (0.047)	92.86
	Claude 3.5 Sonnet `2024-10-22`	0.344 (0.040)	0.593 (0.041)	0.389 (0.040)	**39.22**
	DeepSeek R1 `2025-01-20`	0.379 (0.048)	0.471 (0.049)	0.390 (0.050)	1585.06
Token Classification	`distilbert-base-uncased`	0.379 (0.063)	0.517 (0.083)	0.424 (0.069)	**0.06**
	`bert-base-uncased`	**0.390** (0.023)	0.559 (0.019)	**0.446** (0.022)	0.11
	`distilroberta-base`	0.374 (0.033)	**0.586** (0.024)	0.444 (0.029)	0.62
	`roberta-base`	0.369 (0.022)	0.579 (0.018)	0.441 (0.020)	0.10
Baselines	No-DSD	0.000 (0.000)	0.000 (0.000)	0.000 (0.000)	N/A
	Naive-DSD	**0.317** (0.023)	**0.389** (0.039)	**0.307** (0.028)	N/A

Table 6: Results of the different methods on the SemEval-2016 Task 2 data. As previously mentioned, we only consider the sentences that contain spans annotated as *opposite*. That is why we only report the *NoDiff* scores. We include standard errors within parenthesis. For methods that require a threshold, its value is included in parenthesis next to the model's name. Best results from each method are shown in bold, while best results overall in a column are also underlined. The time column reflects the total time to run the entire evaluation (i.e., 5 folds). Note that for methods that rely on external APIs (e.g., LLM-DSD), times might fluctuate depending on rate limits, connection speeds, etc.

Method	Model	Precision			Recall			F1			Time (minutes)
		Global	NoDiff	Diff	Global	NoDiff	Diff	Global	NoDiff	Diff	
LIME	all-MiniLM-L6-v2 (0.030)	0.355 (0.003)	0.371 (0.012)	0.342 (0.011)	0.287 (0.005)	0.371 (0.012)	**0.224** (0.009)	0.294 (0.004)	0.371 (0.012)	**0.236** (0.010)	**1018.70**
	all-mpnet-base-v2 (0.001)	**0.563** (0.009)	**0.782** (0.014)	**0.397** (0.016)	**0.436** (0.014)	**0.782** (0.014)	0.177 (0.013)	**0.463** (0.013)	**0.782** (0.014)	0.223 (0.014)	1981.81
SHAP	all-MiniLM-L6-v2 (0.010)	**0.394** (0.003)	**0.434** (0.011)	**0.362** (0.011)	**0.370** (0.004)	**0.434** (0.011)	**0.320** (0.011)	**0.366** (0.002)	**0.434** (0.011)	**0.313** (0.010)	**44.47**
	all-mpnet-base-v2 (0.030)	0.321 (0.003)	0.356 (0.011)	0.293 (0.008)	0.256 (0.007)	0.356 (0.011)	0.185 (0.010)	0.266 (0.005)	0.356 (0.356)	0.198 (0.008)	216.00
Embedding	all-MiniLM-L6-v2 (0.010)	0.434 (0.008)	0.564 (0.014)	0.333 (0.015)	0.506 (0.008)	0.564 (0.014)	0.460 (0.015)	0.438 (0.009)	0.564 (0.014)	0.341 (0.013)	**3.75**
	all-mpnet-base-v2 (0.006)	0.445 (0.008)	0.529 (0.019)	0.380 (0.012)	0.585 (0.006)	0.529 (0.019)	0.625 (0.015)	0.469 (0.005)	0.529 (0.019)	0.423 (0.009)	11.28
	text-embedding-3-large (0.005)	0.454 (0.025)	**0.699** (0.040)	0.334 (0.033)	0.632 (0.023)	**0.699** (0.040)	0.599 (0.027)	0.486 (0.028)	**0.699** (0.040)	0.381 (0.035)	111.52
	text-embedding-004 (0.005)	**0.527** (0.008)	0.666 (0.013)	**0.421** (0.007)	**0.650** (0.005)	0.666 (0.013)	**0.635** (0.011)	**0.547** (0.008)	0.666 (0.013)	**0.456** (0.006)	611.42
LLM	GPT-4o 2024-11-20	0.854 (0.009)	0.833 (0.010)	0.870 (0.013)	0.709 (0.004)	0.833 (0.010)	**0.615** (0.012)	0.740 (0.004)	0.833 (0.010)	**0.670** (0.010)	**234.93**
	Llama 3.3 70B Instruct 2024-12-06	0.841 (0.008)	0.831 (0.012)	0.847 (0.013)	0.692 (0.006)	0.831 (0.012)	0.585 (0.011)	0.723 (0.007)	0.831 (0.012)	0.640 (0.011)	584.77
	Claude 3.5 Sonnet 2024-10-22	**0.886** (0.005)	**0.895** (0.009)	**0.878** (0.007)	**0.711** (0.006)	**0.895** (0.009)	0.571 (0.007)	**0.750** (0.005)	**0.895** (0.009)	0.640 (0.007)	337.80
	DeepSeek R1 2025-01-20	0.622 (0.011)	0.302 (0.014)	0.861 (0.007)	0.452 (0.009)	0.302 (0.014)	0.564 (0.006)	0.492 (0.010)	0.302 (0.014)	0.634 (0.005)	2739.06
Token Cl.	distilbert-base-uncased	0.553 (0.021)	0.595 (0.019)	0.521 (0.030)	0.513 (0.020)	0.595 (0.019)	0.452 (0.030)	0.513 (0.020)	0.595 (0.019)	0.451 (0.029)	0.62
	bert-base-uncased	0.546 (0.017)	0.587 (0.019)	0.516 (0.022)	0.506 (0.015)	0.587 (0.019)	0.448 (0.021)	0.506 (0.015)	0.587 (0.019)	0.446 (0.021)	1.11
	distilroberta-base	0.638 (0.014)	0.747 (0.012)	0.552 (0.025)	0.592 (0.013)	0.747 (0.012)	0.470 (0.027)	0.592 (0.012)	0.747 (0.012)	0.471 (0.024)	**0.57**
	roberta-base	**0.738** (0.010)	**0.838** (0.018)	**0.657** (0.022)	**0.697** (0.008)	**0.838** (0.018)	**0.585** (0.023)	**0.690** (0.008)	**0.838** (0.018)	**0.574** (0.022)	1.23
Baselines	No-DSD	0.429 (0.020)	**1.000** (0.000)	0.000 (0.000)	**0.429** (0.020)	**1.000** (0.000)	0.000 (0.000)	**0.429** (0.020)	**1.000** (0.000)	0.000 (0.000)	N/A
	Naive-DSD	**0.455** (0.018)	0.003 (0.002)	**0.793** (0.014)	0.266 (0.011)	0.003 (0.002)	**0.462** (0.008)	0.311 (0.013)	0.003 (0.002)	**0.542** (0.010)	N/A

Table 7: Mean results achieved on the SSD by the various DSD methods, evaluated via 5-fold cross-validation. We include standard errors within parenthesis. For methods that require a threshold, its value is included in parenthesis next to the model's name. Best results from each method are shown in bold, while best results overall in a column are also underlined. The time column reflects the total time to run the entire evaluation (i.e., 5 folds). As already mentioned, for methods that rely on external APIs (e.g., LLM-DSD), times might fluctuate depending on rate limits, connection speeds, etc. The elevated time for the DeepSeek R1 model was due to the fact that the model was often formatting the output incorrectly, and therefore a lot of retries were needed.

B. Examples of Annotated Outputs

Table 8 collects four different examples of annotations outputted by the different methods that we have presented, together with the reference annotation for comparison.

Sentence 1	Sentence 2	Method	Annotation
Following the Civil War, the plantation house was destroyed by hurricanes.	Before the Civil War, hurricanes destroyed the plantation house.	Reference Annotation	{{Before the Civil War}}, hurricanes destroyed the plantation house.
		LIME all-MiniLM-L6-v2 $\alpha = 0.030$	{{Before the}} Civil {{War}}, hurricanes destroyed the {{plantation house.}}
		SHAP all-MiniLM-L6-v2 $\alpha = 0.010$	Before the Civil War{{,}} hurricanes destroyed the plantation house{{.}}
		Embedding all-MiniLM-L6-v2 $\alpha = 0.010$	Before the Civil War, hurricanes destroyed the plantation house.
		Embedding text-embedding-3-large $\alpha = 0.005$	{{Before}} the Civil War, hurricanes destroyed the plantation house.
		LLM GPT-4o	Before {{the Civil War}}, hurricanes destroyed the plantation house.
		Token Classification roberta-base	{{Before the}} Civil War, hurricanes destroyed the plantation house.
		Baseline Naive-DSD	{{Before}} the Civil War, hurricanes destroyed the plantation house.
Retail version supports RAM disk size between 5MiB to 64GiB.	Enterprise version supports RAM disk size between 1GiB to 128GiB.	Reference Annotation	{{Enterprise version}} supports RAM disk size {{between 1GiB to 128GiB}}.
		LIME all-mpnet-base-v2 $\alpha = 0.015$	Enterprise version supports RAM disk size between 1GiB to 128GiB.
		SHAP all-mpnet-base-v2 $\alpha = 0.030$	{{Enterprise}} version supports RAM disk size between 1GiB {{to}} 128GiB.
		Embedding all-mpnet-base-v2 $\alpha = 0.006$	{{Enterprise version}} supports RAM disk size between 1GiB to 128GiB{{.}}

		Embedding text-embedding-004 $\alpha = 0.005$	{{Enterprise version supports}} RAM disk size {{between 1GiB to 128GiB.}}
		LLM Claude 3.5 Sonnet	{{Enterprise version}} supports RAM disk size {{between 1GiB to 128GiB.}}
		Token Classification distilroberta-base	{{Enterprise version}} supports RAM disk size between {{1GiB}} to {{128GiB}}.
		Baseline Naive-DSD	{{Enterprise}} version supports RAM disk size between {{1GiB}} to {{128GiB}}.
China stock index futures close higher – Dec. 4	China stock index futures close lower – Jan. 24	Reference Annotation	China stock index futures close {{lower}} – Jan. 24
		LIME all-MiniLM-L6-v2 $\alpha = 0.010$	China {{stock index}} futures {{close lower}} – {{Jan}}. {{24}}
		SHAP all-MiniLM-L6-v2 $\alpha = 0.005$	China {{stock}} index futures close lower {{– Jan. 24}}
		Embedding all-MiniLM-L6-v2 $\alpha = 0.005$	China stock index futures close lower – {{Jan. 24}}
		Embedding text-embedding-3-large $\alpha = 0.005$	{{China stock index futures close lower – Jan. 24}}
		LLM Llama 3.3 70B Instruct	China stock index futures {{close lower}} – {{Jan. 24}}
		Token Classification bert-base-uncased	China stock index futures close {{lower –}} {{Jan. 24}}
		Baseline Naive-DSD	China stock index futures close {{lower}} – {{Jan. 24}}

Table 8: Examples of annotated outputs. The first two sentence pairs are coming from the SSD, while the third proceeds from the SemEval-2016 Task 2 data. In the latter, it can be seen that the differing dates “Dec. 4” and “Jan. 24” have not been included in the reference annotations (they were annotated as *similar*). This is one of the caveats with the SemEval-2016 Task dataset: there are no unambiguous annotations of dissimilarity beyond the *opposite* spans.

C. Embedding-DSD Algorithm

Algorithm 1 and Algorithm 2 describe the steps involved in the Embedding-DSD method. Algorithm 1 returns a map (dictionary) specifying the dissimilarity score for each of the unigrams in the second sentence. After that, it still remains applying the threshold over these scores in order to determine which unigrams will be included within a dissimilar span.

Algorithm 1 Embedding-DSD.

procedure EMBEDDING-DSD($sentence1, sentence2$)
  $baseSimilarity \leftarrow$ cosSim($sentence1, sentence2$)
  $replacements \leftarrow$ getReplacements($sentence1, sentence2$)
  $similarityGains \leftarrow \{\}$
  for $indices, repl \in replacements$ **do** ▷ Replaced unigram indices, and resulting string
    $gain \leftarrow$ cosSim($sentence1, repl$) $- baseSimilarity$ ▷ Similarity gain yielded by the replacement
    $similarityGains[indices] \leftarrow gain$
  end for
  $unigrams \leftarrow$ getUnigrams($sentence2$)
  $unigramDiff \leftarrow \{\}$ ▷ Map from unigram index to array of gains
  for $uniIndex \leftarrow 0$, len($unigrams$) $- 1$ **do** ▷ Assign *n*-gram gains to individual unigrams
    for $gainIndices, gain \in similarityGains$ **do**
      if $uniIndex \in gainIndices$ **then**
        $unigramDiff[uniIndex]$.insert($gain$)
      end if
    end for
  end for
  for $uniIndex \in unigramDiff$ **do** ▷ Aggregate gains
    $unigramDiff[uniIndex] \leftarrow$ aggregateGains($unigramDiff[uniIndex]$)
  end for
  return $unigramDiff$
end procedure

D. LLM Prompts

Here, we include the prompts that we used for 1) generating the differing spans in the SSD, and 2) evaluating different LLMs on the SSD.

D.1. Prompt used for the SSD creation

As mentioned earlier, we initially tried to prompt the LLM to generate both the modified spans, as well as their labels (i.e., 0 if the modified span pair was semantically dissimilar, or 1 if it was equivalent in meaning). However, since the labels were often incorrectly annotated, we resorted to labeling them ourselves. To get the annotated sentence, we used the following prompt:

Algorithm 2 Calculate all possible *n*-gram replacements.

procedure GETREPLACEMENTS($sentence1, sentence2$)
  $sent2Unigrams \leftarrow$ getUnigrams($sentence2$)
  $sent2Replacements \leftarrow \{\}$ ▷ Map from replaced indices to resultant string
  for $nGramSize \leftarrow 1$, len($sent2Unigrams$) **do** ▷ Consider all possible *n*-gram sizes
    $sent1NGrams \leftarrow$ getNGrams($sentence1, nGramSize$)
    for $nGram \in sent1NGrams$ **do** ▷ Generate replacements for each retrieved *n*-gram
      for $i \leftarrow 0$, len($sent2Unigrams$) $-$ len($nGram$) $+ 1$ **do** ▷ Positions that allow a replacement
        $replacedIndices \leftarrow [\,]$
        $replaced \leftarrow sent2Unigrams$.copy()
        for $j \leftarrow 0$, len($nGram$) **do** ▷ Replace the unigrams present in the *n*-gram
          $replacedIndices$.insert($i + j$)
          $replaced[i + j] = nGram[j]$
        end for
        $sent2Replacements[replacedIndices] \leftarrow replaced$.toString()
      end for
    end for
  end for
  return $sent2Replacements$
end procedure

```
I will give you some sentences below, each of them separated by a newline. Your task
is to modify spans of text. The modified spans can either the same meaning as the
original, or an opposite meaning. Please, annotate your spans in both the original
and the modified sentence with the characters `{{`for the beginning of a span, and
`}}`for its ending. Also, try to replace spans containing more than one word. For
example:

Input 1:

“Thinking about it, the issue might be more critical than what we first thought.”

Output 1:

“Thinking about it, the issue {{might be more critical}} than what we {{first}}
thought.”  => “Thinking about it, the issue {{is less relevant}} than what we
{{initially}} thought.”


Input 2:

“She gazed out of the window, contemplating the distant city lights twinkling in the
evening darkness.”

Output 2:

“She gazed {{out of the window}}, {{contemplating}} the distant city lights twinkling
in the evening {{darkness}}.” => “She gazed {{inside her heart}}, {{pondering}} the
distant city lights twinkling in the evening {{obscurity}}.”


Here are the sentences to annotate. Give me only the output as provided in the example
above, and don’t leave blank lines between outputs:

“It has been reported from a number of states.”
```

“It has a W10 loading gauge but W9 rolling stock is excluded.”

[...]

“Timberlacing includes finely webbed Dhajji.”

“Other programs exclude the famous CUNY College Now and GEAR UP.”

Don’t forget to include spans that have equivalent meaning.

In our prompts, we included around 35 sentences (above omitted with “[...]” for brevity). The last instruction was added due to the model’s tendency to output uniquely dissimilar spans. Some example outputs for the previous input would be:

“It has {{been reported from}} a number of {{states}}.” => "It has {{come in from}} a number of {{regions}}.”

“It has a {{W10 loading gauge}} but {{W9 rolling stock}} is excluded.” => "It has a {{high capacity gauge}} but {{smaller stock}} is excluded.”

[...]

“{{Timberlacing includes}} finely webbed {{Dhajji}}.” => "{{Wooden reinforcements feature}} finely webbed {{timber frames}}.”

“Other programs {{exclude the famous}} CUNY College Now and {{GEAR UP}}.” => "Other programs {{leave out the well-known}} CUNY College Now and {{the GEAR UP initiative}}.”

D.2. Prompt used in LLM-DSD

In order to evaluate LLMs on the SSD, we adopt an in-context learning setting, providing within the prompt four examples with the expected inputs and outputs.

First, the following message is passed to the model as *system prompt*:

You are an NLP model able to detect differences in meaning in textual pairs. More concretely, given a premise and a hypothesis, you are able to compare them and annotate in the hypothesis the spans that are differing in meaning to the information included in the premise.

Here is an example of an input and the expected output:

````
# INPUT 1

```json
{
  "premise": "There was international outrage for the decision.",
  "hypothesis": "There was no reaction to the decision."
}
```
````

````
# OUTPUT 1

```
There was {{no reaction}} to the decision.
```
````

As you can see, the inputs are formatted as a JSON blob containing the premise and hypothesis. The response is a code block containing the hypothesis with the differing spans enclosed within the markers "{{" and "}}". Note that without these markers, both the input hypothesis and the annotated hypothesis are identical.

Here is another example:

````
# INPUT 2

```json
{
````
````

````
  "premise": "Microorganisms are too small to be seen by the naked eye.",
  "hypothesis": "Microorganisms have considerable size and can be seen with your eyes."
}
```
````

````
# OUTPUT 2

```
Microorganisms {{have considerable size}} and {{can be seen with your eyes}}.
```
````

On the other hand, here goes an example with a incorrect output, since the annotated hypothesis includes words not present in the input hypothesis:

````
# INPUT 3

```json
{
  "premise": "It is much warmer here than it used to be.",
  "hypothesis": "It is way colder here than it used to be."
}
```
````

````
# (INCORRECT) OUTPUT 3

```
I believe it is {{way colder}} here than it used to be.
```
````

Let me show you one last example of an erroneous output, in this case because the annotation appears at the end and not within the hypothesis:

````
# INPUT 4

```json
{
  "premise": "The deputy was urged to provide an immediate apology for his controversial comments.",
  "hypothesis": "The deputy was urged to resign for his controversial comments."
}
```
````

````
# (INCORRECT) OUTPUT 4

```
The deputy was urged to resign for his controversial comments. {{resign}}
```
````

Then, the specific instance to be annotated was passed as a *user message*:

````
You are now given the following JSON input:

```json
{
  "premise": "He worked as a curator, editor and bibliographer.",
  "hypothesis": "He worked as a gardener, writer, and librarian."
}
```

Please, provide the annotated hypothesis using the start marker "{{" and the end marker "}}". Enclose the annotated hypothesis within a code block using "```" so I can easily identify it. Please, reason your answer.
````
````

E. Instructions for Annotators

In this section, we present the specific instructions that were provided to the annotators, which were intended to guide them in the validation process and ensure the accuracy and correctness of the SSD.

Span Similarity Dataset – Annotation Instructions

The data to be annotated is provided as a `.tsv` file called `annotate.tsv`. In each line, there are two sentences separated by a tab character (`\t`). These sentences are very similar, except for certain word spans. Here we define a "span" as a group of contiguous words. In some cases, the differing spans are equivalent in meaning. In others, they have different meaning. Your task is to annotate the differing spans and determine whether they have equivalent meaning or not. More specifically:

1. Identify the differing spans and enclose them within double curly braces, i.e., `{{` to signal the beginning of a span, and `}}` to signal its end. Try to annotate the spans in such way that the spans have enough context on its own (e.g., in example 2, "food" and "portions" are included in spans).
   (a) Example:
   - Sentence 1: `I'm not a bad talker either.`
   - Sentence 2: `I'm not an eloquent speaker either.`
   - Annotated Sentence 1: `I'm not {{a bad talker}} either.`
   - Annotated Sentence 2: `I'm not {{an eloquent speaker}} either.`

   (b) Example:
   - Sentence 1: `Food is way overpriced and portions are very small.`
   - Sentence 2: `Food is reasonably priced and portions are generous.`
   - Annotated Sentence 1: `{{Food is way overpriced}} and {{portions are very small}}.`
   - Annotated Sentence 2: `{{Food is reasonably priced}} and {{portions are generous}}.`
2. Add a space to the end of the line. Then write a `1` if the annotated span is equivalent, or a `0` if it is dissimilar. In case there are several spans, separate the numbers with a comma (leave no space in between the numbers), e.g., `0,1`. Span pairs will always appear in the same order in both sentences.
   (a) Example:
   - Original Line:
     `I'm not a bad talker either. I'm not an eloquent speaker either.`
   - Annotated Line:
     `I'm not {{a bad talker}} either. I'm not {{an eloquent speaker}} either. 0`

   (b) Example:
   - Original Line:
     `Food is way overpriced and portions are very small. Food is reasonably priced and portions are generous.`
   - Annotated Line:
     `{{Food is way overpriced}} and {{portions are very small}}. {{Food is reasonably priced}} and {{portions are generous}}. 0,0`

You can find some examples of the resultant annotation in *examples.tsv*. Please, **do not use any assistance from a Large Language Model (LLM)** or any other automatic annotation methods, since the goal is to evaluate human performance on the task. Save your annotations as `annotated_your-first-name.tsv`. Thank you!

F. Licenses of the Resources Employed

Reporting the licenses of resources used is essential for ensuring transparency and reproducibility, allowing others to verify and build upon previous work without legal ambiguities. In Table 9, we collect the licenses of the datasets we used. Table 10 is an analogous table collecting the model licenses.

Dataset	License	Homepage
Span Similarity Dataset (SSD)	CC BY-SA 4.0	https://dmlls.github.io/dissimilar-span-detection
SemEval-2016 Task 2 Dataset	CC BY-SA 4.0	https://alt.qcri.org/semeval2016/task2

Table 9: Licenses of the datasets we used in our work.

Model	License	Homepage
`all-MiniLM-L6-v2`	Apache License 2.0	https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2
`all-mpnet-base-v2`	Apache License 2.0	https://huggingface.co/sentence-transformers/all-mpnet-base-v2
`text-embedding-3-large`	Proprietary	https://platform.openai.com/docs/guides/embeddings/#embedding-models
`text-embedding-004`	Proprietary	https://ai.google.dev/gemini-api/docs/embeddings
`GPT-4o`	Proprietary	https://openai.com/index/hello-gpt-4o
`Llama 3.3 70B Instruct`	Llama 3.3 Community License Agreement	https://www.llama.com/docs/model-cards-and-prompt-formats/llama3_3/
`Claude 3.5 Sonnet`	Proprietary	https://www.anthropic.com/claude/sonnet
`DeepSeek R1`	MIT	https://github.com/deepseek-ai/DeepSeek-R1
`bert-base-uncased`	Apache License 2.0	https://huggingface.co/google-bert/bert-base-uncased
`roberta-base`	MIT	https://huggingface.co/FacebookAI/roberta-base
`distilbert-base-uncased`	Apache License 2.0	https://huggingface.co/distilbert/distilbert-base-uncased
`distilroberta-base`	Apache License 2.0	https://huggingface.co/distilbert/distilroberta-base

Table 10: Licenses of the models we used in our work.