Going Beyond Sentence Embeddings: A Token-Level Matching Algorithm for Calculating Semantic Textual Similarity

Hongwei Wang and **Dong Yu**
Tencent AI Lab, Seattle, WA
{hongweiw, dyu}@global.tencent.com

Abstract

Semantic Textual Similarity (STS) measures the degree to which the underlying semantics of paired sentences are equivalent. State-of-the-art methods for STS task use language models to encode sentences into embeddings. However, these embeddings are limited in representing semantics because they mix all the semantic information together in fixed-length vectors, which are difficult to recover and lack explainability. This paper presents a token-level matching inference algorithm, which can be applied on top of any language model to improve its performance on STS task. Our method calculates pairwise token-level similarity and token matching scores, and then aggregates them with pretrained token weights to produce sentence similarity. Experimental results on seven STS datasets show that our method improves the performance of almost all language models, with up to 12.7% gain in Spearman's correlation. We also demonstrate that our method is highly explainable and computationally efficient.

1 Introduction

Measuring the similarity between two sentences is an important task in many natural language processing (NLP) applications. This makes Semantic Textual Similarity (STS) a crucial preliminary step in various domains, such as information retrieval (Wang et al., 2020), machine translation (Castillo and Estrella, 2012), plagiarism detection (Foltỳnek et al., 2019), semantic search (Mangold, 2007), and conversational systems (Santos et al., 2020).

Large pretrained language models (Devlin et al., 2018; Liu et al., 2019) have achieved the state-of-the-art performance on STS task (Reimers and Gurevych, 2019; Gao et al., 2021; Chuang et al., 2022). These approaches typically use language models to encode input sentences into embeddings and then calculate STS using similarity metrics such as the cosine function. However, sentence embeddings have limitations in representing sentences, as all the information of the sentence is aggregated and mixed together in the fixed-length embedding. This problem is especially pronounced for the STS task, which requires fine-grained, low-level semantic understanding and comparison (Majumder et al., 2016). As a result, methods based on sentence embeddings often have difficulty being well-trained and lack explainability for their predicted results.

Going beyond sentence embeddings, we propose a token-level matching algorithm for STS. Our algorithm works in the inference stage, so it can be applied on top of any trained language model to improve its performance. Specifically, given a trained language model (also called base model), we use it to generate token embeddings for the two input sentences and calculate their pairwise token similarity. We then design a novel scoring function to calculate the matching score for each token. The sentence similarity score is calculated by averaging all the token matching scores with token weights, which are learned unsupervisedly from a large corpus. Our method captures fine-grained, token-level information, which is more indicative, robust, and explainable than sentence embeddings.

We conducted experiments on seven standard STS datasets using six language models and their variants as base models. Our method is able to improve the performance of almost all existing language models, especially those "poor" ones (up to 12.7% improvement in Spearman's correlation). Specifically, our model improves SimCSE by 0.8% to 2.2%, and improves ESimCSE by 0.6% to 1.2%, which is the current state-of-the-art model on the STS task. We also demonstrated the explainability of our model by identifying the semantically similar parts between two input sentences.

2 Related Work

Existing work on STS can be broadly divided into two categories: lexicon-based and semantic-based. Lexicon-based approaches (Richardson and

Smeaton, 1995; Niwattanakul et al., 2013; Opitz et al., 2021) calculate the correlation between the character streams of two sentences being compared, which can be applied at the level of characters or words. Semantic-based approaches can be further divided into three categories: word-based methods (Wang et al., 2016), which treat a sentence as a list of words and compare the correlations between words; structure-based methods, which use language tools such as grammar (Lee et al., 2014), part-of-speech (Batanović and Bojić, 2015), and word order (Li et al., 2004) to process sentences and compare their structure; and vector-based methods (Reimers and Gurevych, 2019; Liu et al., 2021; Gao et al., 2021; Wu et al., 2021; Chuang et al., 2022), which calculate sentence embeddings that describe each sentence as a vector and have achieved the state-of-the-art performance on STS.

Our method is conceptually similar to BERTScore (Zhang et al., 2019), a token-level evaluation metric for text generation. However, there are two significant differences between these two approaches: (1) BERTScore is an evaluation metric, while our method is an algorithm for calculating STS; (2) The key designs for token matching score and token weights are also different.

3 The Proposed Method

Given a pair of two sentences $s = \langle t_1, t_2, \cdots, t_{|s|} \rangle$ and $\hat{s} = \langle \hat{t}_1, \hat{t}_2, \cdots, \hat{t}_{|\hat{s}|} \rangle$ where t_i ($\hat{t}_i$) is the i-th token in sentence s ($\hat{s}$), our goal is to learn a function $f(s, \hat{s}) \in \mathbb{R}$ that calculates the semantic similarity between s and $\hat{s}$.

Token-Level Similarity Matrix We can calculate token embeddings for s and $\hat{s}$ using any language model, including pretrained language models (Devlin et al., 2018; Liu et al., 2019), or language models specifically finetuned for the STS task (Li et al., 2020; Gao et al., 2021; Chuang et al., 2022). Given sentence s and $\hat{s}$, the language model generates the token embedding matrix $\mathbf{X} \in \mathbb{R}^{|s|\times d}$ and $\hat{\mathbf{X}} \in \mathbb{R}^{|\hat{s}|\times d}$, where each row corresponds to a d-dimension token embedding. The token-level similarity matrix for s and $\hat{s}$ is then calculated as $\mathbf{S} = \mathbf{X}\hat{\mathbf{X}}^\top$, in which the entry $\mathbf{S}_{ij}$ indicates the similarity between token t_i and $\hat{t}_j$.

Token Matching Score The token matching score measures the likelihood that a given token in one sentence can be matched to a token in the other sentence. This score takes into account two aspects: (1) *significance*. Similar to BERTScore (Zhang et al., 2019), we match a token to its most similar token in the other sentence. For example, the significance score of $t_i \in s$ is $sig(t_i) = \max_{\hat{t}_j \in \hat{s}} \mathbf{S}_{ij}$. (2) *uniqueness*. It is important to note that a high score for $sig(t_i)$ does not necessarily mean that t_i can be matched to a certain token in $\hat{s}$, but rather that $\mathbf{S}_{ij}$ is high for all $t_j \in \hat{s}$. To measure how unique $sig(t_i)$ is, we define the uniqueness score of t_i as $uni(t_i) = \max_{\hat{t}_j \in \hat{s}} \mathbf{S}_{ij} - \text{2nd-max}_{\hat{t}_j \in \hat{s}} \mathbf{S}_{ij}$, i.e., the difference between the maximum and the second maximum value of row $\mathbf{S}_{i\cdot}$. We provide an ablation study on the two parts in our experiments.

The token matching score is defined as the sum of the above two scores:

$$\begin{aligned} S(t_i) &= sig(t_i) + uni(t_i) \\ &= 2 \cdot \max_{\hat{t}_j \in \hat{s}} \mathbf{S}_{ij} - \text{2nd-max}_{\hat{t}_j \in \hat{s}} \mathbf{S}_{ij}. \end{aligned} \tag{1}$$

Similarly, for $\hat{t}_j \in \hat{s}$, we have $S(\hat{t}_j) = 2 \cdot \max_{t_i \in s} \mathbf{S}_{ij} - \text{2nd-max}_{t_i \in s} \mathbf{S}_{ij}$.

Token Weighting Tokens typically have different levels of semantic importance. Previous work (Zhang et al., 2019) uses inverse document frequency (IDF) as token weights, as rare words can be more indicative than common words. However, in many cases, high-frequency words can be semantically important (e.g., "not") while low-frequency words may be semantically unimportant (e.g., specific numbers). To address the mismatch between token importance and token frequency, we propose learning token weights from plain texts.

Specifically, we choose unsupervised SimCSE (Gao et al., 2021) as the training model, which takes an input sentence and predicts itself in a contrastive objective with dropout used as noise. During the training stage of SimCSE, instead of using the last-layer embedding of CLS token as the sentence embedding, we assign a trainable weight parameter w_i for each token i in the vocabulary and calculate the weighted average $\sum_{i \in s} w_i \mathbf{t}_i$ as the sentence embedding for s, where $\mathbf{t}_i$ is the last-layer embedding of token $i \in s$. In this way, the token weights w can be trained together with the model parameters of SimCSE on a large unsupervised corpus, which has been shown to be more semantically precise than frequency-based token weights.

The final STS score of the input sentences $(s, \hat{s})$ is the weighted average of all token scores:

$$f(s, \hat{s}) = \frac{\sum_{t_i \in s} S(t_i) w_{t_i}}{2 \sum_{t_i \in s} w_{t_i}} + \frac{\sum_{\hat{t}_i \in \hat{s}} S(\hat{t}_i) w_{\hat{t}_i}}{2 \sum_{\hat{t}_i \in \hat{s}} w_{\hat{t}_i}}. \tag{2}$$

Model	**STS12**	**STS13**	**STS14**	**STS15**	**STS16**	**STS-B**	**SICK-R**	**Avg. gain**
Sentence-BERT$_{\text{base}}$	67.6/**70.3**	70.0/**77.4**	67.4/**74.5**	75.8/**80.4**	69.8/**76.0**	73.1/**78.2**	70.4/**73.8**	+5.2
Sentence-BERT$_{\text{large}}$	71.1/**73.7**	70.1/**78.1**	70.2/**76.3**	77.9/**82.4**	73.8/**79.2**	77.2/**81.1**	71.4/**75.5**	+4.9
ConSERT$_{\text{base}}$	**65.9**/65.7	73.6/**81.6**	65.9/**72.7**	75.5/**81.4**	74.0/**80.3**	73.0/**78.0**	65.9/**68.2**	+4.9
ConSERT$_{\text{large}}$	69.9/**71.6**	82.1/**85.6**	71.4/**76.2**	82.1/**84.2**	77.0/**80.8**	76.9/**81.5**	71.1/**71.7**	+3.0
Mirror-BERT$_{\text{base}}$	59.7/**63.7**	59.3/**80.0**	53.4/**71.1**	68.8/**79.9**	62.5/**78.4**	57.9/**76.1**	67.7/**69.3**	+12.7
Mirror-RoBERTa$_{\text{base}}$	65.3/**67.8**	80.5/**81.8**	72.0/**73.7**	79.7/**80.9**	77.4/**79.1**	77.6/**79.7**	70.0/**70.5**	+1.6
SimCSE-BERT$_{\text{base}}$	**68.4**/67.4	82.4/**84.9**	74.4/**76.7**	80.9/**83.3**	78.6/**82.4**	76.9/**82.5**	72.2/**72.3**	+2.2
SimCSE-BERT$_{\text{large}}$	**70.9**/70.8	84.2/**86.8**	76.4/**79.1**	84.5/**85.9**	79.8/**83.4**	79.3/**84.4**	**73.9**/72.8	+2.0
SimCSE-RoBERTa$_{\text{base}}$	70.2/**71.1**	81.8/**82.5**	73.2/**74.8**	81.4/**82.2**	80.7/**81.7**	80.2/**82.0**	68.6/**69.6**	+1.1
SimCSE-RoBERTa$_{\text{large}}$	72.9/**73.5**	84.0/**84.8**	75.6/**76.8**	84.8/**85.2**	81.8/**82.5**	82.0/**82.9**	71.3/**72.0**	+0.8
ESimCSE-BERT$_{\text{base}}$	**73.4**/69.9	83.3/**85.7**	77.3/**77.8**	82.7/**84.2**	78.8/**82.4**	80.2/**82.9**	72.3/**72.8**	+1.1
ESimCSE-BERT$_{\text{large}}$	**73.2**/72.6	85.4/**86.8**	77.7/**79.5**	84.3/**85.5**	78.9/**82.2**	80.7/**84.0**	**74.9**/73.1	+1.2
ESimCSE-RoBERTa$_{\text{base}}$	69.9/**71.2**	82.5/**83.0**	74.7/**76.0**	83.2/**83.7**	80.3/**81.6**	81.1/**82.7**	70.6/**71.5**	+1.1
ESimCSE-RoBERTa$_{\text{large}}$	73.2/**73.8**	84.9/**85.4**	76.9/**77.8**	84.9/**85.4**	81.2/**81.9**	82.8/**83.4**	72.3/**72.9**	+0.6
DiffCSE-BERT$_{\text{base}}$	**72.3**/66.5	**84.4**/83.7	**76.5**/75.5	**83.9**/83.0	80.5/**80.6**	80.6/**80.7**	**71.2**/70.0	-1.3
DiffCSE-RoBERTa$_{\text{base}}$	70.1/**70.9**	**83.4**/83.1	75.5/**76.0**	**82.8**/82.6	82.1/**82.8**	82.4/**83.6**	71.2/**72.2**	+0.5

Table 1: Spearman's correlation results (in %) on seven STS datasets. The numbers before "/" are the results of the original models, and the numbers after "/" are the results of applying our method on top of the original model. The higher number is highlighted.

4 Experiments

4.1 Evaluation Setup

We evaluate our method on seven STS datasets: STS 2012-2016 (Agirre et al., 2012, 2013, 2014, 2015, 2016), STS Benchmark (Cer et al., 2017), and SICK-Relatedness (Marelli et al., 2014). Each dataset consists of sentence pairs and their corresponding ground-truth similarity scores. We use Spearman's correlation to evaluate the predicted results of our method and all baseline methods on the test set. Baseline methods include Sentence-BERT (Reimers and Gurevych, 2019), ConSERT (Yan et al., 2021), Mirror-BERT (Liu et al., 2021), SimCSE (Gao et al., 2021), ESimCSE (Wu et al., 2021), and DiffCSE (Chuang et al., 2022). We use the pretrained models released by the authors as our base model, then compare the performance of our method with them, as shown in Table 2. The Sentence-BERT and ConSERT models were downloaded from `https://github.com/yym6472/ConSERT`, while the other pretrained models can be directly loaded by their names using HuggingFace API. We use the last hidden layer representation of the [CLS] token as the sentence embedding, because it performs much better than the representation after the pooling layer in almost all cases.

Model	**Name**
Sentence-BERT$_{\text{base}}$	sup-sbert-base
Sentence-BERT$_{\text{large}}$	sup-sbert-large
ConSERT$_{\text{base}}$	unsup-consert-base
ConSERT$_{\text{large}}$	unsup-consert-large
Mirror-BERT$_{\text{base}}$	cambridgeltl/mirror-bert-base-uncased-sentence
Mirror-BERT$_{\text{large}}$	cambridgeltl/mirror-bert-large-uncased-sentence
SimCSE-BERT$_{\text{base}}$	princeton-nlp/unsup-simcse-bert-base-uncased
SimCSE-BERT$_{\text{large}}$	princeton-nlp/unsup-simcse-bert-large-uncased
SimCSE-RoBERTa$_{\text{base}}$	princeton-nlp/unsup-simcse-roberta-base
SimCSE-RoBERTa$_{\text{large}}$	princeton-nlp/unsup-simcse-roberta-large
ESimCSE-BERT$_{\text{base}}$	ffgcc/esimcse-bert-base-uncased
ESimCSE-BERT$_{\text{large}}$	ffgcc/esimcse-bert-large-uncased
ESimCSE-RoBERTa$_{\text{base}}$	ffgcc/esimcse-roberta-base
ESimCSE-RoBERTa$_{\text{large}}$	ffgcc/esimcse-roberta-large
DiffCSE-RoBERTa$_{\text{base}}$	voidism/diffcse-bert-base-uncased-sts
DiffCSE-RoBERTa$_{\text{large}}$	voidism/diffcse-roberta-base-sts

Table 2: Base models and their names.

4.2 Main Result

Table 1 shows the Spearman's correlation results on the seven STS datasets. In each entry, the number before "/" is the result of the original model (using the embedding of the CLS token as the sentence embedding), while the number after "/" is the result of applying our method on top of the original model. The last column shows the average absolute gain of our method compared to the baseline method across all tasks.

Our method can improve the results for almost all models. The improvement is particularly signif-

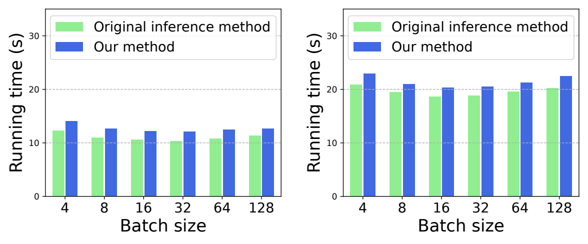
Variant	STS-B	SICK-R
TOKEN MATCHING FUNCTION		
2·max−2nd-max (our model)	**82.0**	**69.6**
max	81.6	69.3
max−2nd-max	68.1	56.0
TOKEN WEIGHTS		
Pretrained weights (our model)	**82.0**	**69.6**
IDF weights	79.9	67.5
No weights	80.2	68.6
max, IDF weights (BERTScore)	79.9	67.8

Table 3: Ablation study of different token matching score functions and token weights. The base model is SimCSE-RoBERTa$_{\text{base}}$.

icant if the original model does not perform well, e.g., Sentence-BERT (+5.2% and +4.9%) and ConSERT (+4.9% and +3.0%). From another perspective, our method can be seen as a universal booster for language models on the STS task. For example, our method can improve the Spearman's correlation of all base models to around 80 or even higher on STS-B dataset, regardless of the original performance of the base model. This indicates that even "poor" language models can still generate high-quality token embeddings that preserve token similarity information very well. However, existing language models only use a single embedding to represent a sentence, which mixes all the information of the sentence together and makes it difficult for language models to be well-trained.

4.3 Ablation Study

We investigate the impact of different token matching functions and token weights, which are two key components of our method. The base model here is SimCSE-RoBERTa$_{\text{base}}$, but the conclusion is similar for other base models. The results are reported in Table 3. For the token matching function, we find that the performance slightly drops when using only `max` and substantially drops when using only `max−2nd-max` . This suggests that significance is more important in measuring token matching scores, while considering uniqueness further improves the performance. For token weights, we observe that IDF weights do not perform well and are even worse than the variant with no token weights. We also evaluate the variant of `max` + IDF weights, which is the same design as BERTScore (Zhang et al., 2019). Our model outperforms BERTScore by around 2% on both datasets.



(a) SimCSE-RoBERTa$_{\text{base}}$ (b) SimCSE-RoBERTa$_{\text{large}}$

Figure 1: Running time of the original inference method and our method on two base models for STS-B dataset.

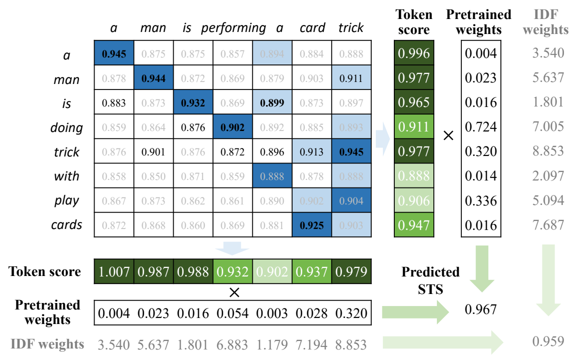

a man is performing a card trick
a 0.945 0.875 0.875 0.857 0.894 0.884 0.888
man 0.878 0.944 0.872 0.869 0.879 0.903 0.911
is 0.883 0.873 0.932 0.869 0.899 0.873 0.897
doing 0.859 0.864 0.876 0.902 0.892 0.885 0.893
trick 0.876 0.901 0.876 0.872 0.896 0.913 0.945
with 0.858 0.850 0.871 0.859 0.888 0.878 0.888
play 0.867 0.873 0.862 0.861 0.890 0.902 0.904
cards 0.872 0.868 0.860 0.869 0.881 0.925 0.903
Token score
0.996 0.977 0.965 0.911 0.977 0.888 0.906 0.947
Pretrained weights
0.004 0.023 0.016 0.724 0.320 0.014 0.336 0.016
IDF weights
3.540 5.637 1.801 7.005 8.853 2.097 5.094 7.687
×
Token score 1.007 0.987 0.988 0.932 0.902 0.937 0.979
×
Pretrained weights 0.004 0.023 0.016 0.054 0.003 0.028 0.320
IDF weights 3.540 5.637 1.801 6.883 1.179 7.194 8.853
Predicted STS
0.967
0.959

Figure 2: Case study on a sample sentence pair from STS-B dataset. See Section 4.5 for details.

4.4 Running Time Analysis

We investigate the running time of our method. We set the base model as SimCSE-RoBERTa$_{\text{base}}$ or SimCSE-RoBERTa$_{\text{large}}$, and then run the original inference method and our inference method on STS-B dataset with batch size ranging from 4 to 128 on an Nvidia Tesla P40 GPU. The results, shown in Figure 1, indicate that our method only incurs an average time overhead of 12.9% and 9.1% on the two base models, respectively.

4.5 Case Study

As a case study, we consider a sentence pair from STS-B dataset: "`a man is performing a card trick`" and "`a man is doing trick with play cards`", whose ground truth similarity is the highest level. The token similarity matrix for this pair is shown in Figure 2, with a dark/light blue background indicating the first/second highest scores in each row, and bold/black numbers indicating the first/second highest scores in each column.

Exactly matched tokens ("`a`", "`man`", "`is`", and "`trick`") receive the highest scores, which are indicated by color dark green. Tokens that cannot be matched ("`a`", "`with`", and "`play`") receive the lowest scores, which are indicated by color light green. Tokens that are not exactly the same but are semantically equivalent (“performing”-“doing”, “card”-“cards”) receive scores that fall in the middle level. While using only max (i.e. significance) as the token matching score may produce similar result, adding the term max−2nd-max (i.e. uniqueness) improves the reliability and distinguishability of those scores. This is why our model performs slightly better than max, as shown in Table 3.

Additionally, Figure 2 demonstrates that pretrained token weights are more accurate than IDF token weights. Some semantically unimportant tokens, such as “a”, “is”, and “with”, are given too much weight when using IDF method, which affects the overall accuracy of the prediction. As a result, the predicted STS using pretrained token weights (0.967) is also more accurate than using IDF token weights (0.959).

5 Conclusion

This paper presents a token-level matching algorithm for calculating STS between pairs of sentences. Unlike previous approaches that use pretrained language models to encode sentences into embeddings, our method calculates the pairwise token similarity, and then applies a token matching functions to these scores. The resulting scores are averaged with pretrained token weights to produce the final sentence similarity. Our model consistently improves the performance of existing language models and is also highly explainable, with minimal extra time overhead during inference.

Limitations

Our model does not follow existing sentence embedding models that encode sentences into embeddings. Therefore, one limitation of our method is that it is specifically designed for STS task (or more precisely, sentence comparison task) and cannot be easily transferred to other tasks, such as sentence classification.

Additionally, our approach incurs a slight extra time overhead of approximately 10%, which may be unacceptable for applications that require high time efficiency.

Our method only takes into account the semantic comparison of individual tokens, rather than considering the meaning of combinations of tokens or phrases. A possible direction for future work is to incorporate the consideration of compositional semantics, for example by grouping tokens into phrases and applying a similar phrase-level matching algorithm.

References

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Inigo Lopez-Gazpio, Montse Maritxalar, Rada Mihalcea, et al. 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.

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel M Cer, Mona T Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2014. Semeval-2014 task 10: Multilingual semantic textual similarity. In *SemEval@ COLING*, pages 81–91.

Eneko Agirre, Carmen Banea, Daniel Cer, Mona Diab, Aitor Gonzalez Agirre, Rada Mihalcea, German Rigau Claramunt, and Janyce Wiebe. 2016. Semeval-2016 task 1: Semantic textual similarity, monolingual and cross-lingual evaluation. In *SemEval-2016. 10th International Workshop on Semantic Evaluation; 2016 Jun 16-17; San Diego, CA. Stroudsburg (PA): ACL; 2016. p. 497-511.* ACL (Association for Computational Linguistics).

Eneko Agirre, Daniel Cer, Mona Diab, and Aitor Gonzalez-Agirre. 2012. Semeval-2012 task 6: A pilot on semantic textual similarity. In ** SEM 2012: The First Joint Conference on Lexical and Computational Semantics–Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 385–393.

Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. 2013. * sem 2013 shared task: Semantic textual similarity. In *Second joint conference on lexical and computational semantics (* SEM), volume 1: proceedings of the Main conference and the shared task: semantic textual similarity*, pages 32–43.

Vuk Batanović and Dragan Bojić. 2015. Using part-of-speech tags as deep-syntax indicators in determining short-text semantic similarity. *Computer Science and Information Systems*, 12(1):1–31.

Julio Castillo and Paula Estrella. 2012. Semantic textual similarity for mt evaluation. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 52–58.

Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation. *arXiv preprint arXiv:1708.00055.*

Yung-Sung Chuang, Rumen Dangovski, Hongyin Luo, Yang Zhang, Shiyu Chang, Marin Soljačić, Shang-Wen Li, Wen-tau Yih, Yoon Kim, and James Glass. 2022. Diffcse: Difference-based contrastive learning for sentence embeddings. *arXiv preprint arXiv:2204.10298*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Tomáš Foltýnek, Norman Meuschke, and Bela Gipp. 2019. Academic plagiarism detection: a systematic literature review. *ACM Computing Surveys (CSUR)*, 52(6):1–42.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. Simcse: Simple contrastive learning of sentence embeddings. *arXiv preprint arXiv:2104.08821*.

Ming Che Lee, Jia Wei Chang, and Tung Cheng Hsieh. 2014. A grammar-based semantic similarity algorithm for natural language sentences. *The Scientific World Journal*, 2014.

Bohan Li, Hao Zhou, Junxian He, Mingxuan Wang, Yiming Yang, and Lei Li. 2020. On the sentence embeddings from pre-trained language models. *arXiv preprint arXiv:2011.05864*.

Yuhua Li, Zuhair Bandar, David McLean, James O'shea, et al. 2004. A method for measuring sentence similarity and iits application to conversational agents. In *FLAIRS Conference*, pages 820–825.

Fangyu Liu, Ivan Vulić, Anna Korhonen, and Nigel Collier. 2021. Fast, effective, and self-supervised: Transforming masked language models into universal lexical and sentence encoders. *arXiv preprint arXiv:2104.08027*.

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 preprint arXiv:1907.11692*.

Goutam Majumder, Partha Pakray, Alexander Gelbukh, and David Pinto. 2016. Semantic textual similarity methods, tools, and applications: A survey. *Computación y Sistemas*, 20(4):647–665.

Christoph Mangold. 2007. A survey and classification of semantic search approaches. *International Journal of Metadata, Semantics and Ontologies*, 2(1):23–34.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. 2014. A sick cure for the evaluation of compositional distributional semantic models. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 216–223.

Suphakit Niwattanakul, Jatsada Singthongchai, Ekkachai Naenudorn, and Supachanun Wanapu. 2013. Using of jaccard coefficient for keywords similarity. In *Proceedings of the international multiconference of engineers and computer scientists*, volume 1, pages 380–384.

Juri Opitz, Angel Daza, and Anette Frank. 2021. Weisfeiler-leman in the bamboo: Novel amr graph metrics and a benchmark for amr graph similarity. *Transactions of the Association for Computational Linguistics*, 9:1425–1441.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Ray Richardson and Alan F Smeaton. 1995. Using wordnet in a knowledge-based approach to information retrieval.

José Santos, Ana Alves, and Hugo Gonçalo Oliveira. 2020. Leveraging on semantic textual similarity for developing a portuguese dialogue system. In *International Conference on Computational Processing of the Portuguese Language*, pages 131–142. Springer.

Yanshan Wang, Sunyang Fu, Feichen Shen, Sam Henry, Ozlem Uzuner, Hongfang Liu, et al. 2020. The 2019 n2c2/ohnlp track on clinical semantic textual similarity: overview. *JMIR medical informatics*, 8(11):e23375.

Zhiguo Wang, Haitao Mi, and Abraham Ittycheriah. 2016. Sentence similarity learning by lexical decomposition and composition. *arXiv preprint arXiv:1602.07019*.

Xing Wu, Chaochen Gao, Liangjun Zang, Jizhong Han, Zhongyuan Wang, and Songlin Hu. 2021. Esimcse: Enhanced sample building method for contrastive learning of unsupervised sentence embedding. *arXiv preprint arXiv:2109.04380*.

Yuanmeng Yan, Rumei Li, Sirui Wang, Fuzheng Zhang, Wei Wu, and Weiran Xu. 2021. Consert: A contrastive framework for self-supervised sentence representation transfer. *arXiv preprint arXiv:2105.11741*.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*.

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
6

☑ A2. Did you discuss any potential risks of your work?
6

☑ A3. Do the abstract and introduction summarize the paper's main claims?
1 and 2

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☒ Did you use or create scientific artifacts?

Left blank.

☐ B1. Did you cite the creators of artifacts you used?
No response.

☐ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
No response.

☐ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
No response.

☐ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
No response.

☐ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
No response.

☐ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
No response.

C ☑ Did you run computational experiments?

4

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
4

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
4

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
4

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
4

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?

Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
No response.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
No response.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
No response.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
No response.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
No response.