Extracting Sentence Embeddings from Pretrained Transformer Models

Lukas Stankevičius[†]   Mantas Lukoševičius[‡]

Faculty of Informatics, Kaunas University of Technology,
LT-51368 Kaunas, Lithuania

† lukas.stankevicius@ktu.lt,   ‡ mantas.lukosevicius@ktu.lt

February 20, 2025

Abstract

Pre-trained transformer models shine in many natural language processing tasks and therefore are expected to bear the representation of the input sentence or text meaning. These sentence-level embeddings are also important in retrieval-augmented generation. But do commonly used plain averaging or prompt templates sufficiently capture and represent the underlying meaning? After providing a comprehensive review of existing sentence embedding extraction and refinement methods, we thoroughly test different combinations and our original extensions of the most promising ones on pretrained models. Namely, given 110 M parameters, BERT's hidden representations from multiple layers, and many tokens, we try diverse ways to extract optimal sentence embeddings. We test various token aggregation and representation post-processing techniques. We also test multiple ways of using a general Wikitext dataset to complement BERT's sentence embeddings. All methods are tested on eight Semantic Textual Similarity (STS), six short text clustering, and twelve classification tasks. We also evaluate our representation-shaping techniques on other static models, including random token representations. Proposed representation extraction methods improve the performance on STS and clustering tasks for all models considered. Very high improvements for static token-based models, especially random embeddings for STS tasks, almost reach the performance of BERT-derived representations. Our work shows that the representation-shaping techniques significantly improve sentence embeddings extracted from BERT-based and simple baseline models.

Keywords: BERT; embeddings; large language models; natural language processing; text embeddings; sentence vector representation; semantic similarity; transformer models; prompt engineering; unsupervised learning.

1 Introduction

Early work on learnable word-level representations [1] showed that semantic meaning can be embedded in numerical vector representations. Arithmetic operations such as `king − man + woman` ≈ `queen` were valid. But can similar or higher abilities also be achieved for whole sentences, not just individual words? They would enable better clustering, classification, and other tasks depending on the whole meaning of the word sequence.

At the core of recent advancements in artificial intelligence is the transformer architecture [2]. Compared to previous RNN-based models, it is fully parallelizable, allowing faster training and greater generalization acquisition from the data. On their 10th birthday, a child is expected to have already encountered and understood the meaning of more than 100 million words [3], and now large language models surpass such scales by multiple orders of magnitude. In addition to improved throughput, transformer models are based on a self-attention mechanism [4] that allows the model to attend to relevant parts of the input sequence. That is similar to how humans understand and experience the meaning of words: using context. The success of the transformer architecture in Natural Language Processing (NLP) tasks, starting with the BERT model [5], was also repeated in other fields such as vision [6], speech [7], and reinforcement learning [8]. Transformer models enabled the solving of more sophisticated tasks such as sentiment analysis or question answering, as well as became the state-of-the-art in almost every NLP task. Therefore, it should possess the representation of the whole text sequence.

Transformer models generally have at least 12 layers and their parameters are counted in hundreds of millions or much more. Condensing the sentence representation using these weights into a fixed 768-length vector (a common length for token vectors in base transformer models) is a challenge. It turns out that simply extracting features from a transformer model's last layer activations yields even worse results than much simpler models [9]. There were multiple works that tried to optimize the architecture parameters (see [10], which evaluated multiple proposed modifications), and multiple investigative works probing the properties of different parts of the model (see a critical review [11] of such approaches). Only the following methods gave tangible results: (1) the plain embedding averaging of all tokens comprising the sequence; (2) engineering a prompt template to condensate sentence representation into a single token; and (3) using a specially dedicated fine-tuned model designed to produce such vectors.

Fine-tuning is definitely the most efficient option. Resulting models, such as InferSent [12] or Sentence-BERT [9], showed state-of-the-art performance in sentence-level tasks at the time. However, the success of fine-tuning depends on several factors. It requires high-quantity and high-quality target domain data, as well as computational resources, which may not always be available. Sometimes, even the existing target domain data cannot be used, as they are very expensive to label. Other difficulties emerge if the data contain sensitive or private information and present a risk of it surfacing during inference. Having such constraints, one has to resort to the first two options of using raw encoded features to produce a vector for a text sequence.

Using feature aggregation instead of fine-tuning also allows us to better explain the inner workings of the state-of-the-art black-box models. Different parts of a transformer model may be responsible for different levels of representation, which favor different tasks. It is also important which levels in the representation hierarchy are easier to shape or process. A better understanding of the inner workings could help address hallucinations or other problems that current large consumer-grade language models face.

One of the two pre-training tasks of the famous BERT model [5] was next sentence prediction. It was optimized through a special `[CLS]` token, which sought to capture the whole-sequence-level representation. But in later works, such as [9], it was revealed that such a representation is very poor, not better than the classic ones, and the authors opted for simple averaging of the last layer tokens instead. The authors of [13] proposed using averaging of tokens of the first and last layers, and the authors of [14] also included hidden token representations of the second layer. But is that really the best way to obtain a numerical representation of the text's meaning?

Pre-trained models like BERT capture a lot of useful representations, yet it is not that trivial to extract them. The authors of [15] showed that further improvements can be achieved by removing the most frequent, sub-word, uppercase, and punctuation tokens before averaging. Furthermore, even larger gains can be achieved by using a prompt template, "`This sentence: "[X]" means [MASK]`", where the target sentence is placed instead of `[X]` and the representation of the token `[MASK]` is used as the final representation of the whole sentence. Such a method is a good example of representation extraction without any specific fine-tuning.

Inspired by the above findings, we hypothesize that there may be more ways to distill relevant sentence-level embeddings without directly fine-tuning the pre-trained BERT model. More concretely, in our analysis, we try to find a function that would shape and extract representations of `bert base-uncased` along its layers, target tokens, and additional corpus, so that the best performance in multiple tasks would be achieved. We evaluate our approach on short text clustering, semantic textual similarity, and classification tasks.

Considering the difficulties of acquiring representations from the state-of-the-art transformer models and following the success of recent works demonstrating it, our approach offers these main contributions:

- We provide an extensive and organized review of related work on producing sentence-level embeddings from transformer models.
- We experimentally test how multiple combinations of various of the most promising token aggregation and sentence representation post-processing techniques impact the performance of three classes of different tasks and properties of representations on several models.
- We propose two competitive and simple static token models as baselines: random embeddings and averaged representations ("Avg").
- We propose an improvement for BERT: the BERT + Avg combined model. We experimentally test many weights and layers of how the representations of BERT and Avg can be most effectively mixed.

The rest of this paper is organized as follows. We provide a review of related work in the literature on composing word vectors and representation reshaping to obtain sentence- or text-level embeddings in Section 2. In Section 3, we outline the experimental setting and give a detailed background on our chosen approach, models, and datasets. In Section 4, we present the results. Finally, we summarize the findings of this work in Section 5.

2 Related Work

Taking the BERT model as an example, we are interested in how representation can be aggregated over tokens, layers, and possibly modified, and which models produce the best representations. Finally, we look at the evaluation options.

2.1 Composing Word Vectors

Thousands of works are being carried out on word-level vectors. Now, you can easily download popular publicly accessible Word2Vec [16] and GloVe [17] embeddings. They are lightweight and usually perfectly fit various word-level tasks. On the other hand, modeling sequences of words is considered much more challenging. Count-based approaches lose information on word order and are sparse. Learning higher-order n-gram vector space models, not just words, but phrases or sentences, leads to sparsity, as frequencies vanish for target n-grams and contexts. The latter, in particular, is the main driving force for distributed representations. We will cover special methods dedicated to sentence level in Section 2.3. Nevertheless, the easiest solution is to reuse individual word vectors of the sequence.

The Principle of Semantic Compositionality (usually called Frege's principle) states that "the meaning of an expression is a function of, and only of, the meanings of its parts together with the method by which those parts are combined." [18]. Many scholars use this as a guide to how a sentence/paragraph/document vector should be formed. According to the principle, for much easier acquisition of word vectors, only the method of combination needs to be found. Therefore, in this subsection, we review the most popular candidate combination methods that should give us a sequence vector from its multiple word vectors.

2.1.1 Formal Semantics

Historically, the first methods were formal and based on logic. Here, the meaning of a sentence lies in the conditions under which it is true. A semantic parser, such as Boxer [19], is used to produce semantic representations of the given raw text. One can easily imagine the parse tree as a result of this analysis. As the structure is converted to first-order logic, resulting in a formula, it can then be checked with a theorem prover or interpreted with respect to a model, which is an abstract representation of a situation or setting [20, 21, 22]. Formal representations ensure that both semantic and syntactic information is preserved.

Unfortunately, formal methods have many practical shortcomings. A logic-based system must explicitly maintain the lexical knowledge necessary for the inference. Therefore, expensive human labor is involved in the construction and maintenance of these knowledge resources. More importantly, it must be domain-specific. One cannot simply use all the knowledge bases of the entire community, as this would hinder complex inferences of theorem provers and model building. There is an area of research on reducing processing time, given the large amount of knowledge resources [23]. Due to this property, logic-based systems have been criticized for their lack of robustness and scalability; implemented systems tend to be small-scale and domain-specific [24]. Although being white-box, theoretically clear, and promising, in practice, formal semantics methods are often surpassed by unsupervised distributional approaches capable of utilizing huge amounts of data. As shown in [25], symbolic representations can at least provide additional features for neural approaches.

2.1.2 Tensor Products

In [26, 24], it was proposed to combine two representations, each in vector form, using tensor products, i.e., every element of the first vector is multiplied by every element of the second vector, retaining the products of all the pair combinations. This way, two rank-1 tensors result in a rank-2 tensor. Combining more vectors results in higher-rank tensors. It allows the tensors to represent the relations and role-filler bindings in a distributed fashion. However, it raises problems due to the dimensionality growing exponentially in size as more constituents are composed [27]. Furthermore, as noted in [28], tensor-based models can only efficiently handle sentences of a fixed structure. Unfortunately, in most practical applications, this is not the case.

Some methods try to solve the mentioned problems by sticking to the original vector dimensionality. It can be accomplished using convolution methods [29, 30]. For example, the circular convolution, as presented in [31], achieves compression by summing along the transdiagonal elements of the tensor product. The compression is lossy, but the noisy version of the original vector can be recovered using circular correlation [27] and matched to the original one by comparing with all known component vectors. Due to the same mathematical principles as light holography, these models are also referred to as holographic.

Some works related to tensor products stem from formal semantics in a way that syntax drives the compositional process. In particular, it was expected that one of the most important parts of the sentence is the pair of a noun and an adjective. Machine learning, in particular regression-based, can be used to find the composition method. The idea is that vectors for AN (adjective–noun) pairs can be learned in the same fashion as for regular words. Then, given the constituents and the final AN vector, a linear mapping is learned. In [32], the mapping is learned as a separate matrix for each adjective, while in [33], a generic "AN-slot" function is trained. These methods do not increase the dimensionality, yet it is not clear how they would work with more words than only the AN pairs. A later work [34] extended the idea of the adjective matrix of [32] to other types (in addition to adjectives and nouns). They emulate formal semantics by representing functions as tensors and arguments as vectors. This way, subjects and objects are rank-1 tensors, while verbs are rank-3. However, as the authors trained only nouns, verbs, subject–verb pairs, and subject–verb–object triplets, their method is still limited to phrases no longer than three words.

Although tensor products share some exotic mathematical properties with quantum mechanics, for example, quantum entanglement [30], the framework relies heavily on formal semantics (see [35]). Therefore, it also shares the same weaknesses and is outperformed by pure machine learning approaches.

2.1.3 Averaging

As neural vectors became more effective than traditional approaches [28], it turned out that it was common to derive a vector for a sequence of words simply by summing or averaging individual vectors. In 2009, the authors of [27] reported that for composing a phrase representation, "averaging is the most common form of vector combination". Despite its simplicity, the basic rule of averaging word vectors was shown to work very well [36, 37, 38]. Later work, such as [39, 40, 41], even showed that simple pooling methods, such as basic vector addition or averaging, match or outperform much more sophisticated methods for encoding the meaning of a text sequence. The authors of [42] reported that in out-of-domain scenarios, simple architectures such as word averaging outperform complex Long Short-Term Memory (LSTM) models. In addition, they are even competitive with systems tuned for particular tasks. Later work by [43] evaluated various compositional models and found that word vector averaging performed reasonably well in most supervised benchmarks. According to the authors of [9], even for the currently trending BERT transformer model, to map the sentence to a single vector, the most popular approach is to average the word embeddings of the BERT output layer.

As averaging is well-suited to acquiring sentence optimization, some methods take averaging into their structure to further optimize it. One such example is the deep average network (DAN) [44]. It takes the input as an averaged text sequence and passes it to one or more feedforward layers to finally perform linear classification. Another, C-PHRASE model of [45], is trained to predict the contexts of phrases from the additive combination of their elements. Such a design results in a useful property of C-PHRASE, that summing word vectors yields sequence representation. The authors of [46] found that one should use an average of all hidden states of LSMT, rather than using representation only from the last one. In a similar fashion, the averaging or summing is optimized in the works of the Siamese CBOW model [47], Sent2Vec model [48], and in other works [49, 50, 51].

A model derived from BERT, specially designed for sentence embeddings, SBERT, also uses averaging to pool words from two sequences to a pair of fixed-size sentence embeddings. Later, using the pair and its element-wise difference, the representation is passed to the softmax layer for classification and regression tasks. The authors of [52] tried to improve this configuration by replacing the averaging with a convolutional neural network. However, their improvement was limited to only better scores for the SALBERT model, which was still lower than SBERT based on averaging. Therefore, calculating the mean embedding from multiple word vectors is still a solid approach to the inner workings of current sentence models.

Averaging is used in various scenarios. Some authors employ it to derive static embeddings from contextualized models. In [49], word vectors are learned by predicting them from the average of all contextual embeddings of words (except the target word) returned by the BERT encoder. Other authors of [53] derived static word embeddings from contextualized transformer representations by averaging word embeddings for each word in 100k different contexts. Such derived static versions are of better quality than classic Word2Vec and share their benefits, such as having tens of millions of times lower computational cost than using standard contextual embedding models [49]. The other scenario is to compose a new more expressive meta-embedding from different models and domains. As a means of composition, the authors of [54] proposed concatenation with some trainable methods. However, a later work [55] found that averaging word vectors and padding them with zeros to compensate for dimensionality mismatches is a surprisingly effective method.

Despite its popularity, averaging has inherent deficiencies. The most prominent is the loss of word order information. For this reason, it is sometimes called a bag-of-words representation (that is, imagine that words lay in the bag in no specific order). As shown in [44], models based on averaging are weaker in tasks that require syntax information. The mean pooled static representations of the words "dog chased human" and "human chased dog" will be the same. In addition, simple summation can cause destructive interference, affecting valuable information. This is especially relevant in long documents. In addition to neglected sentence structure and word order, the authors of [43] also note that individual word identities are lost and noninformative words are more prominently represented than essential ones.

However, to some extent, drawbacks can still be overcome. Most tasks do not usually rely on word order, and the number of occurrences of order-sensitive elements, such as double negations, is generally low. Word vectors have a rich representation and some related knowledge can still be found in them. For example, the authors of [56] showed that the average of the word vectors retains information about the original sequence length. It is even more the case with contextualized transformer representations, where starting from the second layer, each hidden vector is expected to "know" every other token vector and, therefore, to be aware of the word order.

2.1.4 Weighted Average

Not all constituent words are of the same importance; therefore, the corresponding vectors should be weighted. As a surprising event in Information Theory has higher information content than an expected event [57], some specific words should also add more weight to the sentence vector than others.

In an early work [27], we can already find different weights for the members of the pairs of adjective–noun, noun–noun, or verb–object phrases. Later, work went from formal semantics to statistics-based measures, in particular the tf-idf scores. They can be used as features on their own, such as in [58], but it is better to use them to weight neural word vectors. For example, the authors of [59] proposed the Composite Document Vector as a concatenation of idf-graded weighted Word2Vec vectors and tf-idf features. The authors of [60] proposed another better weighting approach, called Smooth Inverse Frequency (SIF). Here, higher-frequency words are down-weighted smoothly. The authors of [43] evaluated various compositional models and found that weighted averaging, in particular SIF, resulted in better performance in unsupervised similarity tasks that outperformed all other models. The author of [61] presented the uSIF method, an improvement to SIF, omitting hyperparameter tuning and constructing weighting with both word frequency and word vector length information. The authors of [62] also showed that the mean pooled output of transformers using tf-idf weights is better for clustering than the only regularly averaged output.

A drastic case of weighting—a total removal of some words—is also shown to improve results. The authors of an older work [37] removed the stop words from the sequence before averaging. Meanwhile, as shown in [63, 15], the current performance of the BERT transformer model is also significantly improved in Semantic Text Similarity (STS) tasks if the most frequent tokens are removed. In other cases, the task itself focuses only on a few words, and others become redundant. The authors of [64] found that for the relation extraction task, some sentence embedding methods work better with shorter spans of words than the original sentences. They performed sentence segmentation in a way that a (sub)sentence would cover the identified entity mentions. Other authors of [64] also noted that SentenceBERT and Quickthought on spans or short segments containing two entity mentions are more clusterable than on the original sentences. These works also highlight the unequal contribution that each word has to the final sentence representation vector.

Some works use more complex word weighting schemes. For example, the authors of [46] proposed a form of attention to weight each hidden state of LSTM. Other authors of [65] proposed the idea that each word brings a novel orthogonal basis to the sentence. Therefore, the length of a projection in this direction can be converted into a word's weight for use in the averaging. They come up with the final weight, consisting of three scores: novelty, significance, and corpus-wise uniqueness. Authors of [64] found that for the methods analyzed, such weighting of GloVe embeddings made them the most clusterable. Similar work by [66] incorporated alignment and novelty scores and applied them to contextualized representations of SBERT. Here, the alignment measures how well the word aligns with the neighboring ones, i.e., a well-aligned one is less informative and should be weighted less. The novelty, similar to [65], is expressed as the magnitude of the orthogonal component of the word to the subspace of neighboring words. Furthermore, the authors of [66] perform average weighting through all SBERT layers and also weight each layer-aggregated vector via l_1-normalized variance. This means that words that evolve faster across layers will receive higher weights since they have greater variance. Although complex to implement, such methods are reported to provide some minor performance improvements.

Surprisingly, the norm of word vectors has a large dispersion, as observed by the authors of [67, 68], and during the average pooling operation, it acts as a weight of the word vector. The authors of [48] observed that their trained word vector in this way down-weights frequent tokens by itself, and this weighting follows the hypothesis of Luhn [69], a well-known information retrieval paradigm, stating that mid-rank terms are the most significant for discriminating content. Using these insights, the authors of [70] used the norm of word vectors as a proxy for the importance of words.

2.1.5 Clustering

As shown by [71], averaging word vectors leads to a loss of information. It is in particular significant for longer text sequences, such as documents. Therefore, the idea was developed to aggregate vectors in a more smooth and information-preserving manner. The authors of [72] clustered the word vectors into k groups using k-means. Then, for a document, each cluster vector is obtained as a sum of its constituent vectors. The final representation is then the concatenation of the cluster vectors and the inverse cluster frequency (icf) values, which are calculated using the idf values of the words present in the document. Other authors of [73] proposed a similar method, but viewed it as one that reduces the dimensionality from words to concepts. Concepts (as clusters) are created by clustering word vectors generated from Word2Vec, and frequencies of these concept clusters are used to represent document vectors. Finally, similarly to tf-idf applied to bag-of-words, a weighting scheme, concept frequency-inverse document frequency (cf-idf), is applied to acquired bag-of-concepts. The authors of [74] proposed the concepts of word containers and document containers to explain how such procedures bring benefits. Similarly to other works, they perform clustering of words into distinct clusters. Vectors belonging to the same cluster are averaged, while resulting representations among different clusters are concatenated. Using standard contextualized pre-trained transformer models with frozen weights, the authors of [75], instead of a simple mean pooling, proposed training a categorical variational autoencoder and reported the performance improvement on some STS tasks. This way, similar to clustering, the lower intermediate compressed representation would carry the essential representation of the sequence. Overall, using clustering of independent word vectors allows the document representation to be expressed in a smaller space of concepts, whose concepts are otherwise erased due to destructive interference during the global averaging.

The authors of [76] improved the aforementioned method of [72] by using soft clustering and allowing a single word vector to participate in multiple clusters. More specifically, each word is represented as a $K \times d$ dimensional embedding, where each k^{th} row corresponds to the original word vector weighted by its probability distribution in the k^{th} cluster. The embedding of this word is weighted with the inverse document frequency of the word and summed with the embeddings of other words to form a document vector. As many values in such a vector were observed to be close to zero, sparseness is induced with a given minimal value threshold. The authors of [76] named their method Sparse Composite Document Vector (SCDV).

The success of SCDV was supplemented by numerous other contributions. The authors of [77], instead of the Gaussian Mixture Models (GMMs), used K-SVD (an algorithm for designing overcomplete dictionaries for sparse representation) [78] for the topic modeling. Furthermore, they used a newer word vector algorithm Doc2VecC and a modern SIF weighting and removal of the top principal component of [60]. Other authors of [79], with their SCDV-MS method, additionally performed word sense disambiguation to dissect polysemous words into distinct vectors. In this way, the quality of the clusters improved. To induce sparsity, the authors also applied hard thresholding in an earlier stage in word cluster assignments and trained word vectors with Doc2vecC. Next, with the emergence of transformer models, the authors of [80] utilized contextualized representations in their SCDV + BERT (ctxd) method. First, the corpus is contextualized following the technique of [81]. That is, each corpus word is clustered among its individual recurrence contexts into distinct clusters corresponding to the different meanings of the same word. After such a procedure is repeated for all words, the vector for each word will then be the centroid vector of the closest cluster. The authors of [80] then repeated the original SCVD procedure and the result was an average improvement for STS tasks. Therefore, the idea of turning global averaging to local inside-cluster averaging did not change; only newer representation techniques and models were employed.

2.1.6 Spectral Methods

A text sequence comprising word embeddings can be interpreted as a multidimensional signal over time. Therefore, temporal summarization techniques can be employed.

One such method is the Discrete Cosine Transform (DCT), originally presented in [82]. This invertible function maps an input sequence of N real numbers to the coefficients of the N number of orthogonal cosine basis functions. The DCT components are arranged in order of significance, with the first one being proportional to the simple average of the sequence. DCT is used in data compression by preserving only the most important coefficients.

In [83, 84], DCT was shown to outperform simple averaging of word embeddings. First, it can be attributed to the fact that DCT is structure-sensitive, as it captures signal dynamics. Second, the DCT has multiple coefficients, with the first one already corresponding to averaging; thus, it should capture more information and enrich the performance of probing classifiers. The authors of [83] apply DCT along the sequence of words for each embedding dimension. Then, they retain lower-order coefficients and concatenate them to obtain overall feature patterns in the word sequence. One drawback of this technique is that short sentences must be padded with zeroed vectors. The following work [84] also showed good results for DCT in multilingual and cross-lingual settings.

Concerning the spectral decomposition of DCT, the authors of [85] proposed EigenSent, which utilizes Higher-Order Dynamic Mode Decomposition (HODMD). The method summarizes transitions in a sequence of words into one representative sequence embedding. The authors found that the best performance is achieved when such an embedding is concatenated with simple word vector averaging. This way, information on both the dynamics and the scale of the sequence is captured.

2.1.7 Using Special Tokens

Modern transformer models have several special tokens that can play an important role during representation aggregation. For example, BERT has `[CLS]` designed for the next sentence prediction task, and it is supposed to carry the meaning of the input text sequence. The authors of [86] tried to use this property and trained the model to learn to aggregate everything in this `[CLS]` token. Other special tokens are `[SEP]`, used to separate two text inputs, `[MASK]`, to concentrate the representation of the masked word, padding, and various sentinel tokens (as in T5). Despite its intended purpose, special tokens are generally dismissed, as in [87, 14, 88], and a simple average is used instead. However, such tokens are important, as the authors of [89], after fine-tuning BERT, found a clear tendency for earlier layers to pay attention to `[CLS]` and for later layers to pay attention to `[SEP]`.

Recently, a prompt-based learning paradigm has emerged (see the survey [90]). Here, instead of fine-tuning the model to the downstream task, one reformulates it to look like the one that was being solved during the model pre-training process. Therefore, the main effort goes to prompt engineering, i.e., finding the most appropriate prompt for the given task. Such a strategy is especially suitable for very large, even colossal language models that are difficult to train.

The authors of [15] successfully used prompt engineering to better capture sentence embeddings with the BERT model. During the manual prompt search, the best prompt found was “`This sentence: "[X]" means [MASK].`”, where `[X]` is replaced by the input sentence, and the output vector of token `[MASK]` is used as a final representation of the input sequence. Then, they further optimized the template by fine-tuning it on Natural Language Inference (NLI) data with the contrastive objective, while the BERT model weights were frozen. The authors used the continuous template technique of [91], where each template token is treated as a vector and optimized by gradient descent. The final template was shown to outperform multiple untrained baselines. The authors also showed that different templates can be used effectively to represent the same sentence with different points of view during supervised contrastive learning. The following work by [92] adapted the idea of using prompt instead of mean pooling and also used this representation extraction during contrastive learning.

2.1.8 Aggregating through Layers

One special aggregation that is possible with newer contextualized models is through deep neural network layers. For simplicity, we refer to transformer model blocks as layers.

One can imagine that “each layer will increasingly magnify small but significant differences” [44]. Therefore, the early layers capture more fundamental and low-level information [89], which dominates the learning of shallow lexical and meaning-related knowledge [93]. The authors of [75] reported that the word embeddings of the lower layers of BERT perform better than their upper layers on a word analogy task, and other authors of [53] showed that the first quarter of the models’ layers perform best in lexical semantic understanding. In the middle layers of the transformers, the hidden states are the most transferable [94] and contain the most relevant information [95]. The authors of [96, 97] report that the middle layers of the multilingual transformers are more multilingually aligned. According to the authors of [93], with the inclusion of context in the upper layers, the encoded concepts evolve into a linguistic hierarchy where the middle and upper layers have a better representation of the core linguistic and semantic concepts. Finally, as the author of [98] explains, upper-layer representations become more context-specific.

A general tendency, as shown by [99], is that the STS performance increases until the middle layers before decreasing toward the final ones. The authors of [99] even report that the final layer of most transformer models produces the worst-performing representations. Most scholars agree that this is due to the overspecialization of the last layers to the pre-training task [100]. Therefore, the final layers of BERT are the most task-specific [101] (but not as much as LSTM, as analyzed in [94]). Even during the fine-tuning phase, the authors of [89] determined that the last two layers encoded the highest share of task-specific features attributed to the score gain. Therefore, the last layers are specific to the training objective and cause problems if it differs a lot from the downstream task objective.

However, some work in [102, 103] found that it was beneficial to apply post-processing to the final layers. This enabled them to restore the apparent drop in representational quality in later layers and gave large performance improvements. It is thought that anisotropy, known as the narrow cone of embeddings in vector space, may be to blame, which in

[98] was found to increase from the earlier to the later layers. However, successful post-processing reveals that the last layers also possess relevant knowledge, but it is somewhat obscured in the raw outputs.

The question is how to compose a universal representation from the relevant pieces in multiple layers. One way is to try various combinations. The authors of [14] analyzed all possible two-layer combinations for BERT. They found that combining the top and bottom layers is better than using only the top layers. They tried to expand it to more layers but remained on a combination of three layers (L1 + L2 + L12). From the older works, we can mention the AdaSent model [104]. It operates on a hierarchy of representations derived from a pyramid-like multilayer network structure. The gating network is trained to adaptively select the most appropriate representation in the hierarchy for the given task. A similar gating system for dynamically deciding which intermediate transformer layers to use was proposed by the authors of [105].

The contribution of each layer can also be accounted for by weighted average. The authors of ELMO [106], a contextualized model based on LSTM, present this configuration with additional weight parameters for each layer. Other authors investigating representations [107, 94, 95] call this weighting a scalar mix technique. The authors of [66], instead of learning layer weights for each word, compute them using complex alignment and novelty measures. For a few-layer model like ELMO, other authors of [108] found it useful to concatenate representations across layers while averaging across tokens, rather than using only the top layer. Other authors of [86] extract relevant information from intermediate hidden representations of BERT by treating them as positive samples during contrastive learning and determining which final sentence embedded in the tuned model should be close.

In conclusion, there is no generally accepted method for taking advantage of representations across all layers in an unsupervised setting. In the other, supervised case, the whole network is fine-tuned, and the whole hierarchy of representations is taught to surface representations according to the task at hand.

2.1.9 Other Means of Composition

Many other composition functions can be used to derive sentence embeddings. In an early work [27] on phrase composition, the authors also found that multiplicative and dilation models perform well. However, there are caveats; element-wise multiplication of sparse representations results in loss of information (this effect is especially strong for more than two words), while dilation, just like weighted average, requires additional parameters. It should also be mentioned that max pooling is quite a popular operation to aggregate the output of the LSMT encoder, as shown in [109, 12], and also by the authors of [86], who found it to work best for all BERT layers for the STS benchmark task. According to the authors of [110], max pooling is also competitive with boundary-based methods (such as those that use embeddings of words in the boundaries) for the representation of the text span.

The input text sequence can be combined hierarchically. The authors of [111] used a recursive autoencoder to contract two child words to a single parent one repeatedly, to arrive at a final single common parent representation of the whole sequence. Later, the authors of [112] extended this combination to specifically follow the structure of the parse tree. Such a recursive model allowed them to compose word vectors in a bottom-up approach up to a sentence level and outperform the simple averaging method. However, a drawback is that such a composition is limited to only sentences. Other authors of [41] used similar ideas for combinations, but did not learn any composition function. Instead, given a specific window length, the vectors of words in these windows are averaged, and the resulting embeddings are then max-pooled. Such a setup better preserves the spatial information.

The common average operation can be generalized as a power mean, as proposed for information retrieval in [113]. The authors of [114] introduced such an idea for combining word vectors. They showed that using the document vector as a concatenation of the power mean with different power values, particularly $p = \pm\infty$ (maximum and minimum), $p = 1$ (regular arithmetic mean), and other $p > 0$, substantially improves the representation of the document. Other authors of [115] also proposed a long vector representation. They employ self-attention to replace the max (or average) pooling from either RNN hidden states or convolved n-grams, resulting in the sentence embedding of the matrix form. However, as the authors of [116] criticize, concatenating representations is problematic. According to them, bigger embeddings will always increase the performance of the linear model on top of them. Therefore, embeddings of the same size should always be compared. In light of this, concatenation is an undesirable operation.

2.2 Reshaping Representation Spaces

Word and sentence vectors obtained by older Word2Vec [1], GloVe [17], and current state-of-the-art transformer techniques are not ideal. Therefore, additional processing techniques are applied to eliminate side effects and improve the performance of downstream tasks. In this subsection, we will describe anisotropy, a phenomenon believed to be responsible for this, the reasons for its appearance, and the techniques that attempt to improve the representation space.

2.2.1 Isotropy

Isotropy is defined as uniformity in all orientations. In this work, we talk about embedding spaces of words and sentences. According to the authors of [117], “distribution is isotropic when the variance is uniformly distributed across all dimensions”.

The anisotropic properties of independent word embeddings were first observed by the authors of [118]. They analyzed common embedding models and found that, in all cases, word vectors share a nonzero common vector; therefore, they are not zero-centered. The authors also found that the variance ratios of the first few components derived by PCA decay nearly exponentially. Furthermore, the variances explained by the leading components “encode the frequency of the word to a significant degree”. Therefore, they proposed eliminating the common mean vector and then removing the top principal components, computed on representations from the entire vocabulary. A similar procedure was also shown in an earlier work [60] to be a very good baseline. Here, words were combined by a frequency-weighted average, and then just the first principal component (yet dataset-specific, as computed on the entire dataset) was removed. As shown by the All-But-The-Top (ABTT) method of [118], such a simple procedure, eliminating the common mean vector and a few top dominating directions from the word vectors, greatly improves both the isotropy and the downstream task performance (for sentence tasks, a simple average of preprocessed word vectors is used). As found by [119], it is already known that the isotropy of the target embedding is critical for the alignment of embeddings, which is important for areas such as domain adaptation, word embedding evaluation, and machine translation. However, the work [118] became the main stimulus for the research direction that attempts to improve the isotropy of word embeddings.

Word representation anisotropy is also observed in the latest contextualized models. The authors of [120] first observed this effect on the transformer machine translation model [2], while the author of [98] found that this is the case for all ELMo [106], BERT [5], and GPT-2 [121] layers. All representations for randomly sampled words from these contextualized anisotropic spaces were observed to be highly similar, far from zero average cosine similarity. Furthermore, the embeddings of any two words are positively correlated. Finally, it was observed that all word vectors in the representation space tend to occupy a narrow cone.

The research community offered some explanations for why anisotropy may occur. We will cover them in more detail below.

Word Frequencies The authors of [122] were among the first to observe that word embeddings encode a surprising degree of information about word frequencies. In [60], it was shown that simply representing sentences as a weighted average of words by their frequencies is a very competitive method. In particular, the authors computed the weighted average of the word vectors in the sentence (the weight of a word w is $a/(a+p(w))$, with a being a parameter and $p(w)$ the frequency) and then removed the projections of the average vectors on their first singular vector ("common component removal"). Later, the authors of [118] suggested removing more than one component. They found that the top PCA directions encode word frequency information; therefore, the method was proposed to null these directions. As this bias is very strong in representation space, it is inevitably related to the anisotropy phenomena.

Similar encoding of frequency information is also evident in contextualized embeddings of modern transformer models. The authors of [123] investigated BERT embeddings and found that high-frequency words are all close to the origin and densely concentrated, while low-frequency words are far away and scattered sparsely. The work in [124] highlighted that anisotropy is the most pronounced in rare words. The authors of [125] also showed that the same applies to multilingual BERT embeddings; they have a biased structure towards word frequency.

This bias to word frequency reveals a weakness. The authors of [126] constructed a dataset using semantic relations extracted from WordNet [127] to test the semantic properties of words. Using that, they showed that the ability of BERT to understand words depends highly on their frequency, and therefore rare words are neglected. It was depicted in [63] that removing the 34 most frequent tokens before averaging can greatly boost the performance on semantic textual similarity tasks. The authors of [124] conducted a simple experiment on the CNN News corpus with popular WordPiece tokenization and found that 30% of the corpus can be accounted for using the 13 most frequent tokens, while to cover at least 98% of the corpus, nearly 15,000 tokens are needed. Therefore, rare words are a constituent part of modern NLP pipelines and, unfortunately, induce deficiencies.

The authors of [120] proposed an explanation for generation models that related observations of the narrow cone form in the representation space and the frequencies of words. They argue that the main culprit is the way models are trained on the language modeling task. During the training process, the ground-truth word embedding will be pushed toward the direction of the hidden state to obtain a larger likelihood. Meanwhile, the embeddings of most words in the vocabulary (with non-appearing and rarely appearing frequencies) will be pushed towards similar directions negatively correlated with most hidden states and thus end up grouped together in the local region of the embedding space. The following work [124] complemented the given explanation with the "common enemies effect"—the effect of the target words producing gradients of the same direction for all of the non-target words at each step of training with cross-entropy loss, and rare tokens are the most affected by it. The authors also found that the embeddings learned by GPT-2, BERT, and RoBERTa do not degenerate into a narrow cone (they only appear as a cone when projected to a lower-dimensional space), but instead drift in one shared direction. Therefore, the "common enemies effect" is argued to be the main culprit behind the anisotropy of the representation space.

Outliers There is evidence from multiple authors that contextualized representations of transformer models (in particular, the BERT family) contain undesirable outliers (interestingly, abnormal dimensions were also previously observed for GloVe vectors in [128]). These are certain positions in word vectors with unusually high values. The authors of [129] called it outlier dimensions; in [130], it is referred to as outlier neurons, while the authors of [102] call it rogue dimensions. In all of these works, it is agreed that it is a significant contributor to the anisotropy of the representation space.

There is debate about what causes outliers. The authors of [129] found that outliers are essential for good downstream performance. They regard it as a distinct model property that emerges during training and even encourage one to consider it during weight initialization. In contrast, in [102], it was found that the behavior of the model is not driven by outliers; rather, only a small subset of linguistic abilities is handled there (such as positional information; its relationship with outliers was revealed in [130]). Even more strangely, the authors of [125] found that the embedding space of the multilingual BERT model does not have outliers as the English BERT does, but is still anisotropic.

There are several options to account for outliers. The most obvious solution is to remove them. The authors of [130] showed that this brings improvements for tasks that directly use the geometry of the embeddings, such as semantic textual similarity; however, ref. [129] showed that disabled outliers significantly degrade both Masked Language Modeling (MLM) loss and downstream task performance. The authors of [102] suggest that rogue dimensions be accounted for through the standardization transformation. Both options aim to reduce the dominating effect of the highest-value dimensions during the similarity measure, such as cosine similarity or Pearson correlation. On the other hand, one can argue that the similarity measure we use, not the representation space, needs improvements.

Criticism Many works try to improve the isotropy of representation spaces, but is that truly the right way to seek performance improvements? Some recent publications, such as [124], expressed doubts about the role of isotropy in model performance, and the authors of [131] even observed a negative relation between isotropy calibration and downstream performance. The works in [124, 117] ask whether there is truly a “narrow cone” in the embedding space. Other authors of [15] argue that the main problems are various biases unrelated to isotropy. Furthermore, the authors of [117] criticized multiple popular measures of isotropy and warned that they are misleading and inaccurate.

The authors of [132] analyzed the contextual embedding space and found isolated clusters and low-dimensional manifolds. They concluded that although the embeddings are globally anisotropic, local isotropy exists. The authors of [133] continued this observation and applied the known preprocessing techniques of [118], not globally but locally, in clustered regions of representations. They observed that sense-level information is shadowed by structural, syntactic, and tense biases (for verbs), which their method of dominant direction removal helps to reduce. Another work [134] showed that during the fine-tuning phase, the anisotropy becomes even worse, while, in contrast, the performance of STS increases. They found that removing the top dominant directions for fine-tuned representations becomes detrimental, as the most essential information resides there. These findings indicate that the representation space of transformer models is quite complicated and that a simple fixation on isotropy can destroy its native features. These also include biases, the usefulness of which may not be obvious to us, but models have found them helpful during the pre-training. In the end, brute-forcing the plain isotropy may harm some intricate semantic details, which we seek to capture.

2.2.2 Post-Processing Methods

Many post-processing methods have been proposed to improve the embeddings. Whether they improve the isotropy of the representation space or help surface-relevant information [135], multiple authors show the benefits of using them. However, these methods generally favor unsupervised tasks, such as semantic textual similarity, with limited improvements for downstream tasks. If enough data on the target task are available, supervised training will inevitably be a better choice, as the transformation relevant to the task is learned. However, post-processing may be the only option if the training samples are obscure.

The simplest is the centering operation. It is especially important for the STS task that involves cosine similarities. The authors of [124] found that it restores a nearly perfectly isotropic distribution. Subtracting the mean is also the first step in several other post-processing methods. In some sense, such operations as centering can be viewed as fine-tuning to the target domain, as all target embeddings participate in providing the mean coordinates.

We already mentioned some operations related to isotropy enhancement in the previous section, such as zeroing target outliers or ABTT. In the following, we will mention the remaining important ones.

Z-Score Normalization The Z-score describes the position of a point in terms of its distance from the mean when measured in units of standard deviation (in the distribution of the target samples). Z-score normalization, also called standardization, transforms the distribution of embeddings to have a zero mean and a unit standard deviation in all dimensions. The works in [102, 103] recommend that you consider it as a post-processing step. The authors of [116] advise using normalization, such as the z-score, in supervised settings as a binary hyperparameter, because it can lead to rank changes.

All-But-The-Top (ABTT) Many works are based on the influential ABTT algorithm [118]. It has a hyperparameter, which tells the number of dominant directions to remove. Principle components are either completely removed or left intact. As a result, the main weakness is finding the right number of top components to remove, which can either cause information loss or eliminate an insufficient amount of noise. The authors of [136] proposed post-processing through variance normalization (PVN) to normalize the variance of leading principal components to the same level instead of total component disposal. The authors of [137] performed the removal in a softer way by employing matrix conceptors [138] in an unsupervised way. A similar technique that employs conceptors can also target specific components, such as those that incorporate social biases, and diminish them, as in [139]. Other authors of [140] proposed learning weights for each dominant direction removal. All these approaches try to preserve useful information residing in the top principal components while narrowing the target noise.

Authors of [141] combined ABTT and dimensionality reduction. They found that the best pipeline is to perform ABTT twice while performing dimensionality reduction in between. In another line of work, the authors of [133, 125], instead of global post-processing, remove local dominant directions in separate clusters in the representation space. In this way, the structure of the representation space is accounted for. Similarly, the authors of [65], in light of the predecessor method of ABTT [60], proposed to eliminate the sentence-dependent principal component, where they rerank the top principal vectors based on correlation with each sentence. This individual removal of dominant directions was shown to improve performance on the STSB task.

Whitening The authors of [13] proposed the whitening approach to alleviate the anisotropy problem of sentence embedding. The whitening operation involves centering embeddings at the origin and making different dimensions have a unit variance and be uncorrelated, turning their covariance matrix into the identity matrix. It was also found to be useful in earlier work for the alignment of bilingual embeddings [142]. As a side effect, the authors of [13] showed that whitening can also be used with a dimension reduction operation. A concurrent work [14] also used a whitening algorithm to improve performance on STS tasks. In that work, the authors combined the first and last layers of BERT embeddings and then normalized them with whitening. Despite the initial success, subsequent work criticized the whitening operation. The authors of [143] showed that whitened representations greatly improve uniformity, but also suffer degeneration in the alignment property (for alignment and uniformity properties, see [144]). To make matters worse, the authors of [145] called whitening a "trick" that only helps with similarity tasks (by partially overfitting) and harms downstream task performance.

2.2.3 Retrofitting

This word vector space specialization method can incorporate semantic knowledge from external resources into word embeddings. The authors of [146] were the first to use the term "retrofitting" for their post-processing step of vector space word representations. Their idea was to incorporate rich relational information from semantic lexicons to word vectors, which are trained in a data-driven fashion using plain texts. The authors showed that the new vectors are similar in both their purely distributional representations and related word types. In addition, they showed improvements in several benchmark tasks.

Retrofitting can be useful for fitting a desired domain that is different from the original one used to pre-train the word vectors. An example is [147] where the authors retrofitted to linkage information in biomedical taxonomy. Another related field is the alignment of cross-lingual word embeddings. The authors of [148] used retrofitting to align the vectors of the source and target languages in the dictionaries. This way, translation pairs are pulled closer while minimizing deviation from the original embeddings and preserving the existing representation. In all these cases, the retrofitting is based on external relational data.

According to the authors of [149], the weakness of retrofitting is that only the vectors of words present in the external constraints (resources) are modified. Therefore, they proposed an explicit retrofitting model in which external knowledge relations are turned into supervised training examples. Here, the distance between synonyms is supposed to be as low as possible, and between antonyms as high as possible, while between remaining words not present in the external knowledge base, it should remain the same as in the original representation space. This way, on top of word vectors, a deep feedforward neural network is trained to retrofit all the word embeddings.

Some work also found success in applying retrofitting to contextualized representations. The authors of [108] observed that for ELMo [106] representations, the distance between the shared word in the paraphrases is even greater than the distance between "large" and "small" in random contexts. Therefore, to improve the representation capabilities, they minimized the difference in contextualized representations of the shared word in paraphrased contexts while differentiating between those in other contexts. This retrofit resulted in improved performance of downstream applications. Other authors of [150] proposed a two-step process: first to train static vectors from contextualized ones, and then to perform retrofitting. They showed that compared to baselines, such an approach gives the best results in a range of intrinsic and extrinsic tasks.

2.2.4 Other Methods

Some methods are proposed to reduce anisotropy during the training process. In particular, the authors of [120] added the specific regularization. According to them, the aperture of the narrow cone of representations can be improved by minimizing the cosine similarities between any two word embeddings. Such regularization encourages the vectors to be more evenly spread and expressive. This method is highly related to the now-popular contrastive learning approach (see Section 2.3.3). Other authors of [151] improve the isotropy of the output embedding matrix using the spectrum control method. They guide the singular value distribution of the embedding matrix throughout the training process and control the decay rate of these singular values.

The authors of [123] suggest learning transformation of the embedding space of the transformer, sometimes called a flow technique. Their method transforms the BERT sentence embedding distribution into a smooth and isotropic Gaussian distribution. During unsupervised training, only the flow network is optimized, while the BERT parameters remain unchanged. Although BERT-flow showed performance improvements on multiple tasks, the technique is criticized in the literature. First, it needs a specialized implementation and has multiple additional parameters. The authors of [131] report that BERT-flow requires on average 4.2 times more time per training epoch. Furthermore, for the STS benchmark task, the authors of [145] report that BERT-flow with the l_2 similarity metric performs even worse than the baseline of the BERT average.

Different post-processing can help surface the different information residing in the embeddings. The authors of [135] proposed a simple unsupervised singular value decomposition to reassign the feature weights. By changing the similarity order of their transformation, they tailored word embeddings in the semantics/syntax (tasks focusing on sing–chant or sing–singing) and similarity/relatedness (tasks solving car–automobile or car–road relationships) axes. Other authors of [152] used spectral filters to dissect BERT representations at different temporal scales. They showed that a low-pass filter yields the highest probing accuracy in topic classification, a high-pass filter in speech tagging, and a middle-pass filter is best in dialog act speech tasks.

2.2.5 Similarity Measures

Some most basic tasks for sentences require measuring the distances between the corresponding embeddings. Usually, cosine similarity is employed. However, if the continuous representation space is curved or a text sequence is treated as a set of words, different metrics or representation space manipulations can be beneficial. The authors of [153] proposed Mutual Information (MI), well known in information theory and statistics, as a candidate for a similarity measure. They managed to successfully estimate MI for continuous random variables by the use of Kraskov–Stögbauer–Grassberger (KSG) estimator [154], which is based on elementary nearest-neighbor statistics. The other authors of [155] advised comparing sentence embeddings consisting of word vectors averaged by rank correlation, such as Kendall's τ. They argued that such measures enable mean pooled representations to rival modern deep ones, used with cosine similarity. The authors of [156] proposed treating sentences as fuzzy sets of words and showed good performance with the specially adapted DynaMax similarity measure. They showed that word vectors alone are sufficient to achieve excellent performance on semantic textual similarity tasks when sentence embeddings and similarity measures are derived using ideas from fuzzy set theory. Other authors of [157] suggest processing each sentence with respect to its found k-nearest neighbors. Then, they find an optimal Euclidean subspace of the sentence manifold where cosine similarity would work best. The authors of [158] also project the sentences onto a fixed-dimensional manifold with the objective of preserving local neighborhoods in the original space. All these works mentioned employ various strategies to enhance the comparison of document representations, tailoring their approaches to the given task.

Many similarity measures between sets employing earth mover's distance were proposed. The authors of [159] were the first to frame the similarity of documents as a transportation problem. The idea is that the distances between similar but different words in two documents should be small. Their Word Mover's Distance (WMD) is the cost of transporting a set of word vectors in an embedding space. This approach was effective, as it managed to exploit similarities between different Word2Vec word vectors, such as the relation of analogies. However, the main deficiency of WMD was that the distances were expensive to compute. Moreover, an output, the single number of a distance between two given documents, can only be combined with k-nearest neighbors or k-means, while applications usually require a whole feature vector. The authors of a subsequent work [160] managed to derive vectors for documents using WMD. They constructed a positive-definite word mover's kernel using a feature map given by the WMD to random documents and then derived document embedding via a Random Features approximation of the kernel. Other authors of [70] proposed Word Rotator's Distance (WRD). It is designed so that the norm and angle of word vectors correspond to the probability mass and transportation cost in the earth mover's distance, respectively. Therefore, the norm of vectors, which is associated with the vector's significance, does not interfere with the calculation of transportation cost, as in WMD. Finally, the authors of [161] attempted to include structural information absent in the WMD's distance estimation between two sets of words. They represented the sentence vector as a weighted average of substructure vectors at a lower level in a recursive way, while a transport plan at a different level explains how the different substructures align.

2.3 Learning Sentence Embeddings Directly

We already mentioned some models that learn the aggregation of tokens or their combination rules. However, in this section, we will look at more direct approaches to learning a representation vector for a given text.

2.3.1 Paragraph Vectors

The method presented in [162], and sometimes called Doc2Vec, is one of the first successful Word2Vec [16] adaptations to sequences of tokens. The authors presented two methods of how document representation could be trained.

In the Distributed Memory model of Paragraph Vectors (PV-DM), every paragraph is mapped to a unique vector, which, together with context words, participates in predicting the next word. This way, such a unique vector acts as a memory that stores the topic of the paragraph and bears its representation. The second is the Distributed Bag Of Words version of the Paragraph Vector (PV-DBOW) model. It leaves only a unique paragraph vector in the input and predicts words randomly sampled from that paragraph in the output. However, unlike the first method, it does not account for word ordering.

In the original article [162], authors represented sentences as a concatenation of DM and DBOW vectors, as they saw such a setup as more consistent. Later works [163, 164] also reported that individual model performance had only marginal differences, yet deviations were usually task-dependent.

The authors of [165] proposed a modification to Paragraph Vectors and called it Doc2VecC. Differently from Doc2Vec, a document vector is derived not as a unique vector, but as an average of sampled constituent word embeddings. The other notable contribution is the added data-dependent regularization that favors informative or rare words while forcing the embeddings of common and non-discriminative ones to be close to zero. At the time, the authors showed Doc2VecC to match state-of-the-art in multiple tasks.

2.3.2 To RNN- and Transformer-Based Models

Shallow Doc2Vec-like models soon met competition from more complicated Recurrent Neural Networks (RNNs). In a sequence-to-sequence [166] (like machine translation) setting, such a model usually has two parts: encoder and decoder. The encoder, going through each token one by one, encodes the input sequence into a fixed hidden representation, which is later used by the decoder to generate the target sequence (translation in NMT) token by token. Meanwhile, only the decoder part is needed to generate sentence representation. Words of the sentence are sequentially fed as input to the RNN, and the final hidden state is interpreted as its representation.

The benefit of RNN models is that the sequential consumption of tokens allows the final representation to account for the word order and process the arbitrary number of tokens. However, the effectiveness of this mechanism is questionable. It turns out that information has a hard time propagating all the way to the final hidden state. As compensation for that, a bidirectional [167] setting was used, which connects two hidden layers of opposite directions to the same output, simultaneously getting forward and backward information. The second solution was to use an attention mechanism [4], which, during decoder token generation, takes a weighted average of the encoder hidden states from all the input tokens. But it was only helpful for sequence-to-sequence tasks, like translation, as such a mechanism removed the last hidden state bottleneck, which was required to compress a long sequence of tokens into a single vector for a whole text representation.

One of the first famous recurrent models for sentence representation is the SkipThought [168]. This model is trained to predict neighboring sentences from the source one. The center sentence is encoded by a bidirectional GRU (Gated Recurrent Unit [169]) that concatenates the last hidden state of a forward GRU and the last hidden state of a backward GRU, and then decodes it into the two target sentences. This way, the encoder is trained to map a sentence to a single representative vector. The authors showed that the model is robust and performed well on all tasks considered. The main drawbacks of this model are the month-long training, huge vector size of 4800 (resulting from the concatenation of 2400 vectors from two separate models), and support of only sentence-level embeddings, as well as the need for training text with a coherent inter-sentence narrative.

As recurrent GRU or LSTM [170] architectures became the standard, the most recent advances in text representation were due to how data and training objectives were chosen. Here, we have to mention a famous InferSent [12] model. The authors showed that the high-quality supervised data, although in low quantity, can hugely increase the performance. They trained an encoder based on a bi-directional LSTM architecture with max pooling, on the Stanford Natural Language Inference (SNLI) [171] dataset. The resulting model outperformed SkipThought after less than a day of training on a single GPU. Other authors of [172] modified the SkipThought model to QuickThoughts by framing the training task as a classification. Instead of generating the target sentence, the model encodes the likely candidates and chooses one of them. Authors of [173] leveraged several data sources with multiple training objectives. Their GenSen model training tasks included Neural Machine Translation, Constituency Parsing, Natural Language Inference, and SkipThought-like training. Finally, authors of [174] proposed even greater utilization of training data by constructing a discourse marker prediction task, predicting tokens such us `because`, `and`, `if`, etc. Such framing of the task allowed the authors to mine vast amounts of text pairs together with the connecting markers. Their DisSent model performed similarly well to InferSent on various transfer tasks. Overall, one can see that how the model is trained and data quality and quantity play a huge role in the resulting text representation performance.

Many drawbacks of the recurrent models were solved by the transformer architecture [2]. The authors decided to get rid of recurrence and leave only the attention mechanism [4] itself. The resulting model (1) is highly parallelizable; (2) only after the first layer, each token has already attended to every other; and (3) it is faster than RNN models when the sequence length is smaller than the representation dimensionality. These properties allowed us to train transformer models with staggering amounts of text much faster and they became state-of-the-art models on multiple tasks.

Previous advances in sentence representation using RNNs were quickly applied to the new architecture. The authors of [175] presented a Universal sentence encoder model. It used only the encoder part of the original transformer model trained with multiple tasks: Skip-Thought-like training, SNLI, and conversational response prediction [176]. After the appearance of the pre-trained BERT model, the authors of [9] proposed SBERT: a pre-trained BERT model further fine-tuned with NLI data. It was a huge success, similar to the earlier InferSent model, yet this time both the new architecture and the general pre-training were the new factors for additional performance advancement. Other authors of [177] tried to further utilize the unsupervised data. Their PAUSE model learned sentence embeddings from a partially labeled dataset and showed that this way, the same performance can be achieved with a smaller fraction of labeled NLI data. Here, we see the same tendency as with the RNN models: new efforts to construct better fine-tuning tasks and better utilize existing data.

2.3.3 Contrastive Learning Approaches

The search for better training tasks and data utilization culminated in a new area of contrastive learning.

Distance-based contrastive loss [178] is more attractive for learning sentence-level embeddings than the more conventional error prediction losses. It allows for a simple construction of self-supervised learning using pairs of positive and negative examples. Embeddings of the first group of samples are encouraged to be the same (by utilizing a loss on a distance function between the text vectors), while the negative ones are encouraged to be different.

In this way, the burden of expensive labeling can be relieved. Moreover, it can be used as an intermediate step between pre-training and fine-tuning to inexpensively align the model to the target task domain. These properties are especially useful for the "data-hungry" transformer models.

The models in this class differ mainly in how they construct the positive example pairs. This can be achieved in many different ways. We will describe them below.

Feature/Vector/Embedding-Level Augmentations These are the modifications to the sentence in its vector representation space.

Dropout [179] is one of the most popular feature-level augmentation techniques. It was originally used as a regularization of neural networks to increase the robustness to noise from the vector space. However, the same can be applied to derive an augmented text sequence representation.

Popular BERT and RoBERTa pre-trained transformer models already have dropout layers and require almost no modifications to employ the dropout augmentation. Two positives are acquired by passing the same sample through the dropout-enabled network twice [180, 143, 181]. This does not require additional preprocessing and can be applied on the fly during the training.

Despite the effectiveness reported, dropout augmentation has the disadvantage of being biased toward sentence length. This was observed in [181]. Two augmented versions of the same sentence are of the same length and can be easily discriminated against randomly drawn negatives of varying lengths. Therefore, instead of learning the general sentence representation to match the same sentence samples, the model now just takes the shortcut by only comparing their lengths. To counteract this, the authors of [181] added token-level augmentations in addition to dropout so that the length of positives would be different. Other authors of [182] also managed to avoid this problem by using negatives produced by a much higher dropout rate than those used for positives.

Another feature-level (also regarded as token-level) augmentation technique is the shuffle of tokens. Usually in pre-trained language models, positional information of each sequence element is brought about by the addition of special positional embeddings to the existing token ones. As a result, the shuffling of positions can be implemented simply as the shuffling of position IDs and the corresponding positional vectors.

Similarly to dropout, there is a cutoff augmentation [183]. Dropout can also be considered as random erasing of weights in $L \times d$ text sequence embedding matrix with L tokens of length d vectors. The token cutoff erases randomly selected rows, while the feature cutoff erases columns of the $L \times d$ matrix. In the work of ConSERT [63] it was found that the shuffle and token cutoff are the two most effective augmentation strategies, significantly outperforming the feature cutoff and dropout.

There are more sophisticated approaches to alter embeddings than a dropout randomly replacing weights with zero values. Specifically, adversarial attacks were shown to be successful. They aim to add worst-case perturbations to the input samples. The authors of [63] used the Fast Gradient Value (FGV) [184] method, which unfortunately relies on supervised loss to compute adversarial perturbations. Other authors of [185] employed the Fast Gradient Sign Method (FGSM) [186]. Perturbation is obtained by applying the sign function to the derivative of the model with respect to the input by contrastive loss. The authors of [185] composed their best augmentation mix as a dropout with FGSM.

Token-Level Augmentations Language gives us many possibilities to convey almost the same meaning through similar or very different sequences of words. This can be easily utilized for augmentation purposes.

Using synonym replacement, random swapping of two consequent words, random insertion and deletion, the so-called Easy Data Augmentation (EDA) the authors of [187] managed to reduce the required dataset size by half for the same performance in multiple classification tasks. In contrastive learning experiments for the biomedical relation extraction task, the authors of [188] found that synonym replacement outperformed random swap and random deletion. Meanwhile, one of the first contrastive works for text [189] showed that back-translation [190] augmentation outperformed the four mentioned techniques. The authors took a pre-trained BERT [5] and pre-trained it further on the input text of GLUE [191] tasks in a contrastive fashion, to be later fine-tuned and evaluated in GLUE. The resulting CERT model achieved slight improvements in addition to the regular BERT. CLEAR [192] was also an attempt to improve the regular BERT and RoBERTa pre-training by jointly using MLM loss and constrastive loss. The authors found that the different combinations of different augmentation strategies (in particular, substitution, span-deletion, and reordering) favor different improvements for GLUE and various SentEval tasks. A recent work [185] tried token-level augmentations such as typos, synonym replacement, paraphrasing, and back-translation but found that they underperformed feature-level ones.

As the popular BERT model is pre-trained with a masked language modeling task, i.e., predicting the true masked token behind the special input `[MASK]` token, the same `[MASK]` can also be used to induce augmentations by randomly masking tokens. The mirror-BERT model of [180] used such random span masking for input augmentation, which, together with feature augmentation, showed gains over off-the-shelf models in both lexical and sentence-level tasks, across different domains and different languages.

Token-level augmentations can greatly contribute to existing vector-level ones. Feature-level augmentations do not affect the length of the text sequence and are therefore susceptible to length bias. ESimCSE [181] showed that simple word repetition augmentation can effectively counteract it. The authors also tested the insertion of the `[MASK]` token with small improvements, while the insertion of stop words slightly decreased the effect.

A better distinction of negatives can also be obtained by token augmentations. The authors of [92] proposed Bidirectional Margin Loss (BML) to incorporate soft-negative samples that are generated using a simple rule-based method. According to its dependency syntax tree, the positive sentence is converted to its negation with correct syntax and clear semantics. The proposed setup saw improvements in the semantic textual similarity tasks from SentEval.

One drawback of token-level augmentations is the chance of producing false positives. It is especially risky in stochastic modifications, such as back-translation, deletion, or insertion, where exact output cannot be controlled. Furthermore, token-level augmentations are more complicated than feature-level ones, and thus they cannot be performed on-the-fly during the training and must be prepared in advance.

Positives by Relative Placement Similar to the older idea of skip-thoughts [168], the placement of sentences in a text can be exploited. In such a setting, text segments near or overlapping each other in the long text should also be near each other in the sentence embedding space. Therefore, text parts near the same anchor in the text can be regarded as positive, while further away as negative pairs for contrastive learning. DeCLUTR [193] for each document in a mini-batch samples multiple anchor spans. The corresponding positives are further sampled for each anchor and can be partially overlapping, adjacent, or subsumed by the anchor. This is accomplished by sampling the anchor span length to be mostly longer than the positive spans. At the time, DeCLUTR showed improvements in SentEval [194] transfer tasks in an unsupervised setting. Location exploitation is a simple approach, but it has some drawbacks. As the authors used the dataset consisting of documents of at least 2048 tokens in length, it clearly reveals that sentence placement methods have limited applications for short text domains. Moreover, random spans of DeCLUTR are subject to fragmentation in semantics.

Positives by Two Networks The augmented version of the sample can also be constructed using two encoders. Different weight initialization and training data order are the most common causes of fluctuations in neural network performance. Although the performance is generally comparable, each model learned differently represents a different local minimum. Such a subtle difference can act as an augmentation. One of the first such setups was the CT model [99]. The simple objective required two models to retain similar representations for identical sentences while distinguishing their representations from different ones.

The authors of [86] presented an approach of two BERT models, one fixed and the other tunable. The authors imposed a contrastive tension between the `[CLS]` token of the tunable model and a holistic representation of the fixed one. The latter is aggregated from all BERT layers, thus encouraging the tunable model to learn to concentrate all the relevant information to the `[CLS]` token.

The authors of [195] proposed using multiple diverse positives instead of the usual two. The authors claim that this increases the probability of "at-least-one" effective positive during training. To implement diverse positives, the agreement of two similarity distributions of samples between two encoder models had to be maximized.

Another setup incorporating two networks can be used for Knowledge Distillation (KD). In [196], after the regular KD step with the original pre-training task, an additional contrastive pre-training was added. The pair of positives for a single text segment consisted of a representation from the teacher model and another from the student. After the third stage of fine-tuning on semantic textual similarity tasks, [196] showed that the 110 M model outperformed the one with 11B parameters.

MoCoSE [185] combines both feature-level augmentations and two branches based on asymmetric BERT encoders. While the online branch is updated through the loss gradient, the second, the target branch, is updated by the Exponential Moving Average (EMA). These discrepancies between the two branches prevented the model from collapsing and allowed the achievement of competitive results in SentEval.

Although two-network approaches are conceptually similar, they require extra memory or time, which is a big drawback. Transformer models already require the latest state-of-the-art GPU with the largest available memory. Techniques such as gradient accumulation are often used to process large mini-batches sequentially, but in the contrastive learning approach, the requirement for large in-batch negatives gives additional complexity.

Positives and Negatives from Supervised Data Up to now, we have described various augmentation techniques that modify existing unlabeled samples for use in contrastive learning. However, there are several labeled datasets that can also be turned into positives or negatives by reusing the label information.

One of the first works to implement this idea was SimSCE [143]. The authors analyzed six candidate datasets. They found that the Natural Language Inference (NLI) datasets, SNLI [171] and MNLI [197], together performed the best. The combined dataset contains sentence pairs in the form of `(premise, hypothesis, label)`, labeled as entailment, neutral, or contradiction, 314k for each class. The SimSCE authors constructed positive pairs from entailment samples and negatives using contradiction ones. The resulting supervised approach achieved state-of-the-art results at the time for SentEval tasks. The authors also tried to incorporate neutral pairs as less weighted negatives but did not observe improvements.

The authors of [198] note that the use of NLI labels to construct positive and negative pairs can contradict apparent semantic information. Elements of the negative pair may not be negative in semantic space. To address this issue, the authors train the PairSupCon model by incorporating an instance discrimination objective, which is claimed to have an implicit grouping effect. The objective discriminates both hypothesis and premise sentences from the positive pair separately from all other sentences in the batch. The authors also incorporated importance weighting on negatives to facilitate the better effect of hard ones. In total, the overall loss consisted of two instances of discrimination (one for hypothesis and the other for premise) with negative weighting and a third cross-entropy predicting NLI labels. Significant improvements were shown for clustering tasks, while for semantic textual similarity tasks in SentEval, only moderate gains were shown.

A novel use of NLI dataset labels was implemented in the PairSCL [199] model. First, the pair of hypothesis and premise sentences from the NLI dataset is passed through an encoder, and the unified representation of such pair is aggregated by a cross-attention module. Then a supervised contrastive loss is applied between positive and negative samples. What is interesting is that positives are regarded as hypothesis and premise pairs from the same class, i.e., whether contradiction, neutral, or entailment. Pairs labeled in different classes are considered negatives. In addition to such supervised contrastive loss, an additional cross-entropy loss is applied that predicts the actual class label. This setup showed improvements in both the NLI and SentEval transfer tasks.

ST5 [200] adopted the two-stage training strategy. In the first, 2B mined question-answering data from Community QA sites were used, framing the question and answer into the positive pair. During the second stage, NLI data were used, similar to SimSCE [143]. Such a more data-rich training allowed the authors to outperform the previous approaches for the SentEval task. Furthermore, using larger models with up to 11B parameters brought even further gains.

Supervised datasets are difficult to produce. If the domain of the task in question differs from publicly available NLI datasets, the easiest solution is to use unsupervised methods. Otherwise, labeled sources are indispensable. Exploiting multiple datasets and label information can make the benefits even more obvious.

Direction of Contrastive Learning for NLP The classic setup of contrastive learning is becoming more nuanced. The usual setup of two positives and multiple in-batch negatives (just any other sentence) is not perfect. As a result, the exact distance to the negatives or between the positives varies, and learning produces only a coarse approximation. The distinction between soft, hard, and weighted negatives begins to arise [92, 143, 198], as well as the use of multiple positives [195]. The classic NT-Xent (also called InfoNCE) loss is thus often modified or additional losses added to incorporate finer training signals. The desire to scale models and datasets is also observed, as in [200], yet we think that more memory-efficient approaches, as in [196], should be prioritized. Nonetheless, contrastive learning currently produces state-of-the-art sentence embeddings.

3 Methods

Our extensive literature review allowed us to see the big picture of the sentence embedding research.

Currently, the evolution of models for sentence embeddings and related NLP tasks is settled at transformers. In this work, we use existing models to source the raw, token-level embeddings. In Section 3.2, we describe the main model that we use in detail, some baselines we used for comparisons, as well as some of our original extensions. In particular, in T1–T4 models, we extend original prompting templates by incorporating more than one `[MASK]` token. Next, we present a new Avg. model where we derived sentence embeddings first by averaging tokens in different contexts and then by averaging the tokens themselves. Finally, we present our BERT + Avg. model, which combines both contextual and multiple-context averaged representations, all derived from the same BERT transformer model.

In addition to these extensions, we found two main directions that can be used to improve sentence embeddings from transformer models: token aggregation and post-processing techniques. Note, however, that our main contribution here is not the methods, as we reuse most of them from the existing works, but the combinations of them on the transformer model and extensive evaluation on multiple tasks. We thoroughly describe the techniques used (and minor extensions) for token aggregation in Section 3.3 and post-processing of embeddings in Section 3.4.

We have noticed that most works confine themselves to a small subset of evaluation tasks, which limits their results' comparability to others. Papers from top conferences always include classification tasks on top of semantic textual similarity, which is usually the only evaluation. In this work, we evaluate sentence embeddings on three different types of tasks: semantic textual similarity, downstream classification, and clustering. We present these tasks in Section 3.5.

We now proceed with the formal definition of the problem we address in this work.

3.1 Problem Formulation

We are given a text sequence s_0, which was tokenized into N individual pieces (i.e., tokens)

$$t_1 \quad t_2 \quad \dots \quad t_N \ , \tag{1}$$

and a model representing each token as a d-length vector $v \in \mathbb{R}^d$ in each of its L layers. We also use static token embeddings denoted as the -1st layer and input embeddings denoted as the 0th layer. They are sums of token, segment, and position vectors, with layer normalization applied on top. Therefore, we obtain the following 3-dimensional representation space $\mathbb{R}^{(L+2)\times N\times d}$:

$$\begin{matrix} v_1^{-1} & v_2^{-1} & \dots & v_N^{-1} \\ v_1^0 & v_2^0 & \dots & v_N^0 \\ v_1^1 & v_2^1 & \dots & v_N^1 \\ \vdots & \vdots & \ddots & \vdots \\ v_1^L & v_2^L & \dots & v_N^L \end{matrix} \ . \tag{2}$$

We want to find such an aggregation function f that the $((L+2)\times N\times d)$-dimensional output from a chosen transformer model would be reduced to as meaningful a d-dimensional vector as possible in $\mathbb{R}^d$.

As a baseline for the `bert-base-uncased` model with $L = 12$ layers, we consider the average over the N tokens in text and over the first and last layers.

$$f_{\text{first+last}}(s_0) = \sum_{n=1}^{N} \frac{v_n^1 + v_n^{12}}{2N}. \tag{3}$$

Additional Context Data

Generally, a corpus has S number of text samples. Therefore, other text sequences can also be used to derive the representation of the target sentence. In this way, our aggregation function f operates on an $(S\times(L+2)\times N\times d)$-dimensional output from the model. S can also be enlarged by using additional datasets. In particular, we used Wikitext-2 from [201]. This can be used for various post-processing techniques such as centering, standardization, PCA, and others, where transformations are learned on corpora other than the target corpus.

3.2 Models

We use multiple text representation methods, focusing mainly on BERT-based ones. Prompting method T4, Averaged BERT (Avg.), BERT + Avg., B2S-100, and Random embeddings (RE) are our proposed models or modifications, while BERT, T0, and B2S are plain adaptations of existing ones. We tested token aggregation and sentence representation post-processing techniques on all eight models. We will describe them in more detail below.

3.2.1 BERT

BERT is a very popular transformer model. When first introduced in [5], it spectacularly outperformed multiple other models on a wide range of tasks.

BERT is based on the original transformer architecture of [2]. The main difference is that instead of sequence-to-sequence workings and encoder–decoder structure, it is composed only of an encoder side. This allows it to solve multiple classification and regression tasks, maintaining the same pre-trained base while only changing the classifier heads on top.

In this work, we use the BERT version called `bert-base-uncased`, a 110 M parameter model containing 12 layers (blocks) and working with lowercase text. We employ the Hugging Face implementation [202] of BERT. It was pre-trained on BooksCorpus (800 M words) [203] and English Wikipedia (2500 M words) datasets, which take 13 GB of plain text combined. The model was trained for 40 epochs (or passes through the corpus). Back then, the training time, model size, and data used were considered to be very large, yet now they are only a small fraction of what the current state-of-the-art models use.

Input Input to the BERT encoder consists of token, positional, and token type embeddings. BERT uses WordPiece tokenization [204]. Splitting less frequent words into subwords (e.g., "transformer" → "transform" and "##er") rather than splitting everything on word boundaries allows one to reach a manageable vocabulary size of 30,000 tokens. BERT, like other transformer models, perceives input as a set; therefore, order information must be supplied in addition. To achieve this, additional positional embeddings are used, with a unique value for each position in a token sequence and feature dimension. Token type embeddings allow the model to distinguish between the two (if there are two) separate sentences (e.g., ⟨ Question, Answer ⟩) in one token sequence. Overall, token, positional, and token type embeddings are added, and then layer normalization [205] and dropout [206] are applied. This results in the input to the BERT model blocks (for simplicity, we will call them layers).

There are 3 special tokens. `[CLS]` starts every token sequence, `[SEP]` ends token sequences and acts as a separator in a pair of sentences, and `[MASK]` is used during pre-training to mask some percentage of the input tokens at random for the model to predict (known as a "Cloze" task [207]). The `[CLS]` token is also used for the next sentence prediction task—predicting whether the second sentence in a pair actually follows the first one in the training dataset or is a random one—the second unsupervised pre-training task of BERT.

Transformer Block Each transformer block (or layer) has two sublayers. The first is a multi-head self-attention mechanism and the second is a position-wise fully connected feed-forward network. The output of each sublayer `Sublayer(x)` is added to the input `x` that bypasses the sublayer through residual connection [208] and the resulting signal ends in layer normalization [205], `LayerNorm(x + Sublayer(x))` becoming the input to the second sublayer or the next layer.

Multi-Head Self-Attention An input tensor $X \in \mathbb{R}^{b \times t \times f}$ with b samples in a batch, t tokens, and f features is first linearly projected into queries $Q \in \mathbb{R}^{b \times t_Q \times h \times f_Q}$, keys $K \in \mathbb{R}^{b \times t_K \times h \times f_K}$, and values $V \in \mathbb{R}^{b \times t_V \times h \times f_V}$ tensors with h heads using learned weights. Note that in the standard case, $t = t_Q = t_K = t_V$ and $f = hf_Q = hf_K = hf_V$. Then, for each sample text in a batch and each head dot, products between feature vectors of every combination of query and key tokens are calculated, resulting in an attention tensor $(QK) \in \mathbb{R}^{b \times h \times t_Q \times t_K}$. It is then scaled by dividing it by $\sqrt{f_K}$, and softmax is applied along t_K so that all the dot products for any given query token and all the key tokens would sum to 1. Now, a dot product is calculated again for the probability-like scores and values tensor, resulting in a weighted selection of V, with the form of $\mathbb{R}^{b \times h \times t_Q \times f_V}$. Finally, the heads are concatenated back and a linear transformation layer $W^O \in \mathbb{R}^{hf_V \times f}$ is applied. The resulting tensor is the same shape as the input one: $\text{MultiHead}(X) \in \mathbb{R}^{b \times t \times f}$.

For a more detailed explanation of the BERT and transformer architecture, please read the original papers [5, 2].

3.2.2 Prompting Method (T0, T4)

The classical use of the BERT model involves two steps: (1) a general pre-training on a very large corpora with unsupervised tasks, and (2) a supervised fine-tuning step on a small target task dataset. The first step is expensive, it requires a lot of data, time, and computing resources, but it is carried out only once. Weights that are produced in an unsupervised way contain a lot of useful representations for the fine-tuning to only perfect them.

Prompt-based methods take advantage of the first step and can completely avoid the fine-tuning stage. The idea is to frame the target task in the original pre-training format, for which the model is essentially optimized. To extract a general sentence representation, the authors of [15] proposed using "`This sentence: "[X]" means [MASK]`" template (which we will call T0), where the target sentence is placed instead of `[X]` and the final layer representation of `[MASK]` is used. This way, the model itself tries to predict the meaning of a given sentence, and we can extract prediction weights before they are turned into probabilities over tokens.

During initial experiments, we manually searched for other more complicated templates than T0, presented in Table 1. We will also use the T4 template, which is much longer and has 3 `[MASK]` tokens that have to be averaged.

Table 1: Manual template search by adding additional text and [MASK] ([M]) tokens. Target sentences are inserted into the place of [X]. Only the average of 12th layer [M] representations are used. Bolded results are the best.

No.	Template	STS	Clust.	Class.
T0	This sentence: "[X]" means [M].	63.4	45.5	**78.7**
T1	This sentence: "[X]" means [M][M].	63.4	46.6	78.6
T2	This sentence: "[X]" means "[M][M]" and is about [M].	**70.4**	52.8	78.3
T3	This sentence from the paraphrase dictionary: "[X]" means "[M]", which is about [M].	69.6	**54.2**	77.4
T4	This sentence from the dictionary: "[X]" means "[M]" and is about [M], which is a synonym for [M].	69.3	54.2	76.8

3.2.3 Averaged BERT (Avg.)

We follow the idea of [53] to average the representations of a token in its different contexts to acquire its static embedding. Yet, we take it even further and construct vectors for sentences by again averaging such static token embeddings over the sentences.

We construct static tokens using the Wikitext-103 [201] dataset and use the same tokenization as of BERT version `bert-base-uncased`. For each of all 28,807 tokens occurring in the Wikitext-103, we sum all vectors produced by the BERT model (which differ due to different contexts) and divide them by their count. Note that some tokens occur very often, while others are rare (for example, the most frequent, “the”, is repeated 6,470,356 times). The process is repeated for every BERT layer, as well as combinations of layers, so that we can observe the performance dependence on this factor as well.

Combining Averaged BERT and Regular BERT (BERT + Avg.) We also wanted to see how sentence embeddings derived from static averaged BERT tokens can contribute to the original BERT representations. Therefore, we averaged sentence embeddings from the two methods mentioned above.

3.2.4 BERT2Static (B2S, B2S-100)

There are more advanced distillation methods than simple BERT token averaging. For example, the authors of [49] trained static word vectors using BERT's contextualized representations. They adapted the Sent2Vec [48] model for words and trained it to predict the word given the context element of it by the contextual representation of the BERT model. This way, the authors obtained vectors for the 750,000 most frequent words. They also showed that this approach results in vectors that perform better than existing static embeddings trained from scratch, while still enjoying a small memory and computational footprint.

In our experiments, we used the `bert_12layer_para` model (downloaded from `https://github.com/epfml/X2Static`, provided in [49], accessed on 1 May 2023), which was trained from the contexts of a paragraph rather than only a sentence. We evaluated two versions of BERT2Static: a regular one (B2S) and one with the 100 most frequent words filtered out (B2S-100).

3.2.5 Random Embeddings (RE)

Instead of using a complex model to derive embeddings for words, we also tested random vectors as token embeddings. More specifically, we assign each token a random vector drawn from the normal (Gaussian) distribution, centered at 0 with 0.1 standard deviation. To facilitate better comparisons with the BERT model, we use the same tokenizer from the BERT version `bert-base-uncased` and make the vectors the same 768-size length. The whole sentence representation is then computed as the average of its constituent token vectors.

3.3 Aggregating Tokens

Every text contains multiple tokens, each with a corresponding embedding vector. To obtain one vector for the whole text out of the many, usually, a simple average is calculated, as discussed in Section 2.1. We use it as a baseline here for multiple methods.

We investigate different methods of token weighting and filtering based on their frequencies. In classical bag-of-words approaches, it is usually accounted for with the tf-idf weighting. We, however, use only idf weighting, because the same tokens in BERT cannot be simply counted due to different contextualization. Given a dataset with N documents and document frequency df_t, defined to be the number of documents in the given dataset that contain a token t, we calculate idf_t for a token t as

$$\mathrm{idf}_t = \log \frac{N}{\mathrm{df}_t}. \tag{4}$$

To account for long/short text differences, we scale idf_t token weights for each text so that they would sum up to 1. This way, to calculate the average embedding for a given text, one need only sum up the weighted vectors. We calculate two inverse document frequencies:

idf_t^W for the Wikitext-2 dataset from [201] and idf_t^T for all samples from the given target task.

We also adopt the method in [63, 15] to drop the most frequent tokens. We choose 33 tokens, which were depicted in the appendix of [15]. Following that article, we also investigate dropping punctuation and subword tokens. We found that dropping all three parts—the most frequent, punctuation, and subword tokens—has the best effect, and following the original work, we name such token aggregation as with removed biases ("-biases").

3.4 Post-Processing Embeddings

It is common in machine learning to standardize datasets, as most methods are designed to work best with normally distributed data: Gaussian with zero mean and unit variance. Yet it is not that trivial, as data may not follow a smooth distribution and may contain disturbing outliers. In the context of the best practices reviewed in Section 2.2, we investigate the following processing methods of embeddings.

3.4.1 Z-Score Normalization

The most basic is the standard score, also called the z-score. It is the number of standard deviations σ by which the value x of a raw score is higher than or below the mean value μ of the raw scores of all samples. We normalize our vectors to have z-score = 1:

$$z = \frac{x - \mu}{\sigma}. \tag{5}$$

3.4.2 Quantile Normalization to Uniform Distribution (quantile-u)

This technique works by making two distributions identical in statistical properties, thus reshaping given data values according to some known distribution function. In our case, we found that the uniform distribution worked very well as a reference. For a more detailed description of the technique, see [209].

We also tried other methods that are more robust to outliers; however, we found their performance marginally below quantile-u. This includes quantile normalization using a normal distribution and RobustScaler (as it is called in the scikit-learn preprocessing library [210], which we used). The latter method removes the median value instead of the mean and scales the data according to the selected quantile range.

3.4.3 Whitening

It is a transformation that produces uncorrelated components, each with a variance of 1.

3.4.4 All-But-The-Top (ABTT)

It is a method introduced in [118]. We start with the given embedding matrix $A \in \mathbb{R}^{b \times f}$ of b sentences, each with f features. First, it is centered by its mean $\mu \in \mathbb{R}^f$ into $\tilde{A} \in \mathbb{R}^{b \times f}$. Using the centered $\tilde{A}$, and given the number d of the top principal components to remove, we then calculate PCA components $U \in \mathbb{R}^{d \times f}$. Now, we project our data into these components to acquire $A^{PCA} \in \mathbb{R}^{b \times f}$:

$$A_{bf}^{PCA} = \sum_d \tilde{A}_{bf} U_{df}. \tag{6}$$

The final processed embedding matrix will be:

$$A' = \tilde{A} - A^{PCA}. \tag{7}$$

The only difference of our approach from the original authors of [118] is that instead of post processing words, we use all-but-the-top to post-process documents.

3.4.5 Normalization

We also experiment with normalization: scaling individual sentence vectors to have a unit norm.

3.4.6 Learning Post-Processing

Some target task datasets can also be too scarce to calculate accurate statistics, such as mean, standard deviation, or others, used in post-processing. Therefore, instead, we also experiment with learning these weights on the Wikitext-2 dataset. We indicate such techniques with a superscript $\cdot^W$.

3.5 Evaluation

We evaluate the investigated methods on multiple clustering, semantic textual similarity, and classification tasks.

3.5.1 Clustering Tasks

We assess the performance on 6 benchmark datasets for short text clustering. Compared to the usual ones, short datasets impose a challenge due to the weak signal caused by sparsity, which is a big problem for classic count-based approaches such as bag-of-words or tf-idf. Table 2 provides an overview of the main statistics, and the details of each dataset are as follows.

Table 2: Dataset statistics for the short text clustering datasets. N is the number of text samples, C is the number of clusters, L/S is the imbalance number defined as the size of the largest class divided by that of the smallest class, $\|V\|$ is vocabulary size, Len is the average number of tokens in each text sample, Alpha is the percent of tokens that are alphabetic (all token characters must be defined in the Unicode character database as "Letter", while tokens with numbers, punctuation, or BERT continuation tokens such as "##ing" are excluded). Statistics with plain word tokenization are marked with "W" and with `bert-base-uncased` model tokenizer as "B".

Dataset	**N**	**C**	**L/S**	$\|V\|$		**Len**		**Alpha, %**	
				W	**B**	**W**	**B**	**W**	**B**
agnews	8000	4	1	21,062	16,140	23	26	100	86
biomedical	20,000	20	1	18,888	9326	13	20	98	64
googleTS	11,109	152	143	19,508	14,763	28	33	100	85
searchsnippets	12,340	8	7	30,643	16,334	19	24	93	77
stackoverflow	20,000	20	1	22,909	7332	8	12	87	71
tweet	2472	89	249	5098	5091	9	11	100	81

Agnews It is a subset of news titles [211], which contains 4 topics selected by [212].

Biomedical It is a subset of PubMed data distributed by BioASQ (`http://participants-area.bioasq.org/`, accessed on 1 May 2023), where 20,000 paper titles from 20 groups are randomly selected by [213].

GoogleTS It contains titles and snippets of 11,109 news articles related to 152 events [214]. We use the full version of the dataset, which includes both titles and snippets, named GoogleNews-TS in [212].

Searchsnippets It is extracted from web search snippets and contains 12,340 snippets associated with 8 groups [215].

Stackoverflow It is a subset of the challenge data published by Kaggle (https://www.kaggle.com/competitions/predict-closed-questions-on-stack-overflow/data, accessed on 1 May 2023), where [213] selected 20,000 question titles associated with 20 different categories.

Tweet It consists of 89 categories with 2472 tweets in total [214].

We perform clustering by running k-means with the scikit-learn [216] package and reported the clustering accuracy, computed using the Hungarian algorithm [217] and averaged over 10 independent runs (we used the codebase from https://github.com/amazon-science/sentence-representations and downloaded clustering datasets from https://github.com/rashadulrakib/short-text-clustering-enhancement/tree/master/data, both were accessed on 1 May 2023).

3.5.2 Semantic Textual Similarity (STS) Tasks

STS assesses the degree to which two sentences are semantically equivalent to each other. A single sample consists of two sentences and a score ranging from 0 for no meaning overlap to 5 for meaning equivalence. The semantic textual similarity shared task has been held annually since 2012 up to 2017 [218, 219, 220, 221, 222, 223] and formed STS12-STS17 datasets. A total of 8628 carefully collected samples from these contests formed the STS benchmark [223]. Table 3 shows details of the STS datasets, including SICK-Relatedness [224] and STS-B test sets, which we also use. Similarly to [145], we also add (STR) [225], a recent semantic relatedness dataset created by comparative annotations.

STS is a very popular choice for evaluating textual embeddings in an unsupervised way. Without any fine-tuning, one can calculate the distance (usually cosine) between two vectors of two sentences, which should correlate to the target equivalence score. It is so widely adopted that almost all work on semantic representations assesses the model performance on this task.

We follow the STS evaluation settings from [143]. First, the evaluation is kept unsupervised by not applying any additional regressors; cosine similarity between embeddings in a pair is taken directly as a model score for similarity. To find the degree of correlation between the annotated and model-given labels, Spearman's rank correlation is used because it measures the rankings instead of the actual scores. Finally, for annual STS challenges, we concatenate all subsets and report the general Spearman correlation for that year (referred to in [143] as "all"). As we used the SentEval toolkit [194], we had to implement concatenation ourselves since it only had "mean" and "wmean" settings available.

Table 3: Dataset statistics for STS and classification tasks. N is the number of text samples, C is the number of clusters, L/S is the imbalance number, defined as the size of the largest class divided by that of the smallest class (for STS tasks, the two classes are binned to 1 point label value length ones from the highest and lowest sides), $\|V\|$ is vocabulary size, Len is the average number of tokens in each text sample, Alpha is the percent of tokens that are alphabetic. Statistics with plain word tokenization are marked with "W" and with `bert-base-uncased` model tokenizer as "B". For MRPC, SICK-R/E, STS-B, and STS tasks, statistics are calculated for tokenized and then concatenated sentences in each pair.

Dataset	**Split**	**N**	**C**	**L/S**	$\|V\|$		**Len**		**Alpha, %**	
					W	**B**	**W**	**B**	**W**	**B**
STS tasks										
STS12		3108		5.2	8127	7802	25	28	83	77
STS13		1500		0.7	5152	5141	20	21	88	82
STS14		3750		1.6	9117	8613	21	23	86	80
STS15		3000		0.9	7364	7185	23	24	89	85
STS16		1186		1.2	3971	4175	26	28	87	83
STR		5500		1.0	22,392	12,883	25	32	83	76
Binary classification										
MR		10,662	2	1.0	20,325	13,802	22	26	84	76
CR		3775	2	1.8	5675	5221	20	22	84	80
SUBJ		10,000	2	1.0	22,636	15,912	25	28	85	78
MPQA		10,606	2	2.2	6239	6248	3	3	97	88
SST2	train	67,349	2	1.3	14,816	11,570	9	11	88	78
	dev	872	2	1.0	4339	4542	20	23	85	76
	test	1821	2	1.0	7053	6824	19	23	85	76
MRPC	train	4076	2	2.1	16,112	12,061	44	50	81	75
	test	1725	2	2.0	10,092	8471	44	50	82	75
Fine-grained classification										
SST5	train	8544	5	2.1	16,579	12,395	19	23	84	75
	dev	1101	5	2.1	5038	5168	19	23	85	76
	test	2210	5	2.3	7929	7478	19	23	85	76
TREC	train	5452	6	14.5	9437	8492	10	11	87	81
	test	500	6	15.3	1100	1247	7	8	85	79
SCICITE	train	7320	3	4.4	27,775	13,603	31	40	100	77
	dev	916	3	4.4	7625	6918	31	40	100	78
	test	1861	3	3.8	12,609	9217	31	41	100	77
SICK-E/R	train	4500	3/...	3.8/3.6	2258	2277	19	20	99	96
	dev	500	3/...	3.8/5.0	1122	1172	20	20	99	96
	test	4927	3/...	3.9/3.9	2271	2291	19	20	99	96
STS-B	train	5749	...	1.3	12,430	10,792	22	25	86	80
	dev	1500	...	0.7	6542	6511	26	28	85	80
	test	1379	...	1.1	4888	4921	22	24	85	81

3.5.3 Downstream Classification Tasks

Differently from STS, these tasks are evaluated in a supervised way. Following the SentEval [194] benchmark suite, the commonly used evaluation protocol is to train a logistic regression or an MLP classifier with a cross-validation setup on top of the frozen representations, and the testing accuracy is used as a measure of the representation quality. We went after the logistic regression classifier and the 10-fold cross-validation scheme, the setting which is the most commonly reported in the literature. We evaluated various binary, ternary, and fine-grained classification as well as regression tasks. A more detailed description of each is presented below, while statistics are presented in Table 3.

Binary Classification It includes sentiment prediction from Stanford Sentiment Treebank dataset SST2 [112], movie reviews MR [226], and customer product reviews CR [227]. In SUBJ [228], binary subjectivity status is labeled for sentences from movie reviews and plot summaries, and in MPQA [229, 230], phrase-level opinion polarity from news articles is predicted. The last is MRPC [231], the Microsoft Research Paraphrase Corpus, from parallel news sources for the paraphrase detection task.

Ternary Classification SCICITE [232] is a domain-specific classification task that assigns one of three intent labels ("background information", "method", "result comparison") to sentences collected from scientific articles citing other articles. Meanwhile, the SICK-E [224] dataset has labels for sentence pairs as "entailed", "contradiction", or "neutral".

Fine-Grained Classification It includes SST5 [112], a 5-level sentiment analysis dataset, and TREC [233], comprising classification of 6 types of questions. We also evaluate regular semantic textual relatedness and similarity tasks SICK-R [224] and STS-B [223] with classification, by splitting the real-valued similarity targets into 5 discrete class labels. For example, a [0,5] score of 3.6 goes to class 3 with weight 0.4 and also to class 4 with weight 0.6.

3.5.4 Isotropy

We use the IsoScore [117] metric to measure the uniformity of the utilization of the embedded space. As shown by the authors of [117], IsoScore has stronger properties than other isotropy measures.

For each sentence embedding method, we calculated IsoScore for the Wikitext-2 dataset from [201]. We split the dataset into individual sentences (now samples) and omitted texts with less than 10 characters.

3.5.5 Alignment and Uniformity

Explaining the success of contrastive methods, the authors of [144] proposed using alignment and uniformity properties to better quantify the quality of representations. Alignment is calculated between semantically related positive pairs and therefore is an expected distance between embeddings of the paired instances:

$$\mathcal{L}_{\text{align}}(f) = \mathop{\mathbb{E}}_{(x,y)\sim p_{\text{pos}}} \left[\|f(x) - f(y)\|^2\right]. \tag{8}$$

Uniformity is computed using representations of the whole space:

$$\mathcal{L}_{\text{uniform}}(f) = \log \mathop{\mathbb{E}}_{(x,y)\overset{\text{i.i.d.}}{\sim} p_{\text{data}}} \left[e^{-2\|f(x)-f(y)\|^2}\right]. \tag{9}$$

Smaller values of both alignment and uniformity indicate better quality of the representations.

As a distribution of positive pairs p_{pos}, we used STS-B task training split pairs with a similarity of 5.0, and for p_{data}, we used sentences from all pairs.

4 Results

Here, we present the results of our various experiments outlined in Section 3.

4.1 Token Aggregation and Post-Processing Techniques

The results of best-performing token aggregation and embedding post-processing techniques are presented in Table 4. The results here are averaged over all the different tasks in the class. See Appendix A for the individual results.

We see that both aggregation and post-processing methods have a great impact on STS and clustering performance. For semantic textual similarity tasks, the best model was improved from 62.3 to 71.6 average Spearman correlation, while for clustering, the best model was improved from 59.2 to 64.8 average accuracy.

Our techniques can even improve the dedicated SimCSE model [143], which was fine-tuned on NLI data. Its main strength lies in semantic textual similarity tasks, where it leads with over 10% difference. However, for clustering tasks, its average accuracy is similar to the other evaluated models at 59.8% and improves up to 64.0%, if we apply the best-performing techniques. This showcases a general tendency that the top-performing models are very good only in a narrow subset of tasks and highlights the importance of our more general methods.

For maximum performance, usually, both aggregation and post-processing techniques must be used. As an example, sentences placed within the T4 template with no techniques applied have a 62.3 average Spearman correlation. With a better aggregation method of idf_t^T, it is pushed up to 69.2, while only using quantile-u post-processing gives 67.3 (see Table 5 with only post-processing techniques applied). However, a combined effort of both idf_t^T and quantile-u gives the average Spearman correlation score of 71.6.

Still, the improvements from both aggregation and post-processing do not exactly add up linearly. This indicates that the improvements to representations of this type may saturate below the perfect scores.

We considered many post-processing techniques for sentence vectors and show the performance of the most popular and best-performing ones in Table 5 separately. Indeed, almost every method somehow improves the average Spearman correlation for STS tasks. For all-but-the-top (abtt), we varied the number of components to remove up to a hundred but settled on removing only the first two, as it was slightly better than the rest. Although highly credited in the literature, abtt-2 still obtained smaller scores than the others. Plainly averaging the token representations, the highest scores for both STS and clustering tasks were achieved using quantile-uniform normalization. We would also like to mention the simple normalization of vectors to the unit length, which was often beneficial for clustering tasks, and whitening normalization, which was learned on Wikitext-2 dataset, and was favorable with the random embeddings model. We trained other post-processing methods on Wikitext-2 too, yet they resulted in similar or slightly smaller scores.

Unlike unsupervised ones, classification tasks do not benefit from the two representation-shaping techniques. There is only negligible improvement for classification tasks, when the biases (punctuation, most frequent, and subword tokens) are eliminated. Otherwise, performance is only decreased.

This result is logical because similarity or clustering tasks are carried out with the resulting text representations directly, while the classification is a supervised task learned on top. In the first case, the most informative components of the representations must be present in the largest principal components (i.e., constitute most of the variance in the data) for high scores, while the supervised logistic regression classifier can learn to extract them from the small principal components or ill-shaped representations on its own.

4.1.1 Avg. versus B2S and B2S-100

An interesting result, as seen in Table 4, is the comparison of simply averaged BERT tokens from many contexts (the Avg. model) and word vectors from a specially trained Word2Vec-style model BERT2Static (the B2S model) using the same BERT contexts, as described in Section 3.2.4. If no token aggregation and post-processing techniques are used, B2S is of similar performance in STS tasks (with average Spearman correlation of 56.7 versus 56.4), worse at clustering (with average clustering accuracy of 53.8 versus 55.2), and comparable in classification tasks (with average B2S accuracy of 76.9 versus 77.3) to the Avg. model. As we can see, the task performance differences are very small.

Apart from training, the second main difference between the Avg. and B2S models is tokenization. The Avg. model uses sub-word tokens, the same as BERT (tokens of which it averages), while B2S models use a vocabulary of the full set of the most frequent words, 20 times the size of the BERT's. Due to this discrepancy, token aggregations for BERT and B2S models are applied differently, as idf statistics for both tokenizations are different; also, B2S does not have the equivalence of biased tokens such as BERT. However, it is very straightforward for B2S to filter out (do not use during averaging) a portion of the most frequent words. During our experiments, we varied this number and found that removing the 100 most frequent words works best, which improved the average Spearman correlation of STS tasks from 56.7 to 60.6 and the average accuracy of clustering tasks from 53.8 to 55.3 compared to full B2S, both with plain token averaging and no post-processing applied.

Despite the additional option for B2S to remove the 100 most frequent tokens, using additional token aggregation and post-processing techniques allows our simple Avg. model to surpass both B2S and B2S-100 (see Table 4). Note that the removal of the most frequent words for B2S works in a similar way to weighted token aggregations or post-processing, the gains are not additive. Now the best average STS Spearman correlation score becomes 69.5 for Avg. versus 66.4 of B2S and the best average clustering accuracy of 62.2 for Avg. versus 57.2 of B2S-100. Additional techniques here helped a much simpler Avg. model outperform a much more complex learned B2S.

The success of such a simple Avg. model inspired us to seek further improvement by combining it with the parent BERT model. For the combined BERT + Avg. model results, please see Section 4.3.

4.1.2 BERT versus Random Embeddings

The most dramatic increase in performance due to the two techniques is observed for the model of random embeddings. For STS tasks, the average Spearman correlation rises from 50.3 to 66.4, while for clustering tasks, it increases from 36.3 to 49.5. Although accuracy for the latter tasks is still very low, for STS, it is just a mere 3.4 points below what BERT managed to accomplish with both techniques applied.

If we were to look at the detailed performance on individual tasks in Appendix A.1, we would find several where random embeddings with the help of the two shaping techniques score higher than the also-shaped BERT representations. For STS14 (Table 8), it is 68.8 versus 68.3 Spearman correlation, while for the googleTS (Table 16), stackoverflow (Table 18), and tweet (Table 19) datasets, the clustering accuracy is, respectively, 69.5 vs. 68.5, 70.6 vs. 59.9, and 58.5 vs. 55.1. This may indicate a smaller complexity of these particular datasets, where the task can largely be solved based on a small set of keywords. Also, if the texts do not contain the natural language of the type the BERT was pre-trained on (e.g., they contain code), the model cannot properly contextualize the tokens, and the random embeddings without contexts work better. Yet it is important to note that it is achieved only with the help of aggregation and post-processing methods on top of the random embeddings.

The largest clustering accuracy difference between BERT and random embeddings is for the agnews and searchsnippets datasets, with the best scores of 86.8 versus 43.4 and 82.9 versus 36.6, respectively. We think that the observed performance gap may be related to the vocabulary and text sizes of the datasets. Stackoverflow and tweet datasets, with random embeddings ahead of the BERT, have the smallest vocabulary sizes of 7332 and 8091 (see Table 2), while agnews and searchsnippets have the largest vocabulary sizes of 16,140 and 16,334 unique tokens, respectively. Because random embeddings rely only on distinctness of tokens, with the increased amount of them, the probability of having exactly the same keywords in two texts drops. BERT embeddings having non-identical but semantically similar tokens and similar representations helps in this case.

Having shorter text lengths (average text sizes of 12 and 11 for stackoverflow and tweet, respectively, versus 26 and 24 for agnews and searchsnippets, respectively) may also help the random embeddings, because they do not average away so easily, and there is less context to consider.

In contrast, other static models have much more comparable scores. For example, the Avg. clustering accuracy is 83.8 (agnews) and 74.2 (searchsnippets).

4.1.3 Isotropy

Many post-processing methods have previously been proposed to improve the isotropy of representations. It was argued that representations from the BERT model fall into the narrow cone and, therefore, are anisotropic. Thus, by raising isotropy, one can improve regular task performance.

We calculate the IsoScore isotropy metric on the Wikitext-2 dataset after applying various post-processing and token aggregation methods for multiple models. We exclude whitening post-processing, as it always produces representations with the IsoScore at the maximum of 100%, just due to its working principle. Then, we compare the isotropy score to the STS, clustering, and classification task performance by calculating the Pearson correlation coefficient. For semantic textual similarity tasks, results are shown in Figure 1, and for clustering tasks, results are shown in Figure 2.

For each method, the baseline IsoScore is very low: always below 5%. After applying weighted token aggregation techniques or post-processing, it is always increased. The smallest improvements of up to 7-8% score are observed for templated models T0 and T4, while the highest, over 80%, are for random embeddings. For other models, the IsoScore reaches around 17%.

Why is the IsoScore of random embeddings improved so much compared to the other models? The answer may be related to the inner workings of RE. Once we generate the random embeddings, these token-level representations have a maximum possible isotropy. The RE model represents text sequences as averages of these vectors; thus, isotropy going from the token level to the document is reduced. However, token aggregation and post-processing techniques mitigate this isotropy loss, allowing for document embeddings to regain most of it back from the token-level ones. Other models start with already low isotropy for token-level representations and thus have less space to improve it.

IsoScore Correlation with Task Performance Six out of eight models have a moderate correlation (more than 54%) between Isoscore and the average Spearman correlation of semantic textual similarity tasks (see Figure 1). For clustering (see Figure 2), it is less apparent, with only four models reaching moderate Pearson correlation. However, for both STS and clustering tasks, the best score for each model is always reached by some improvement of IsoScore, compared to the isotropy score of the plain averaging setup.

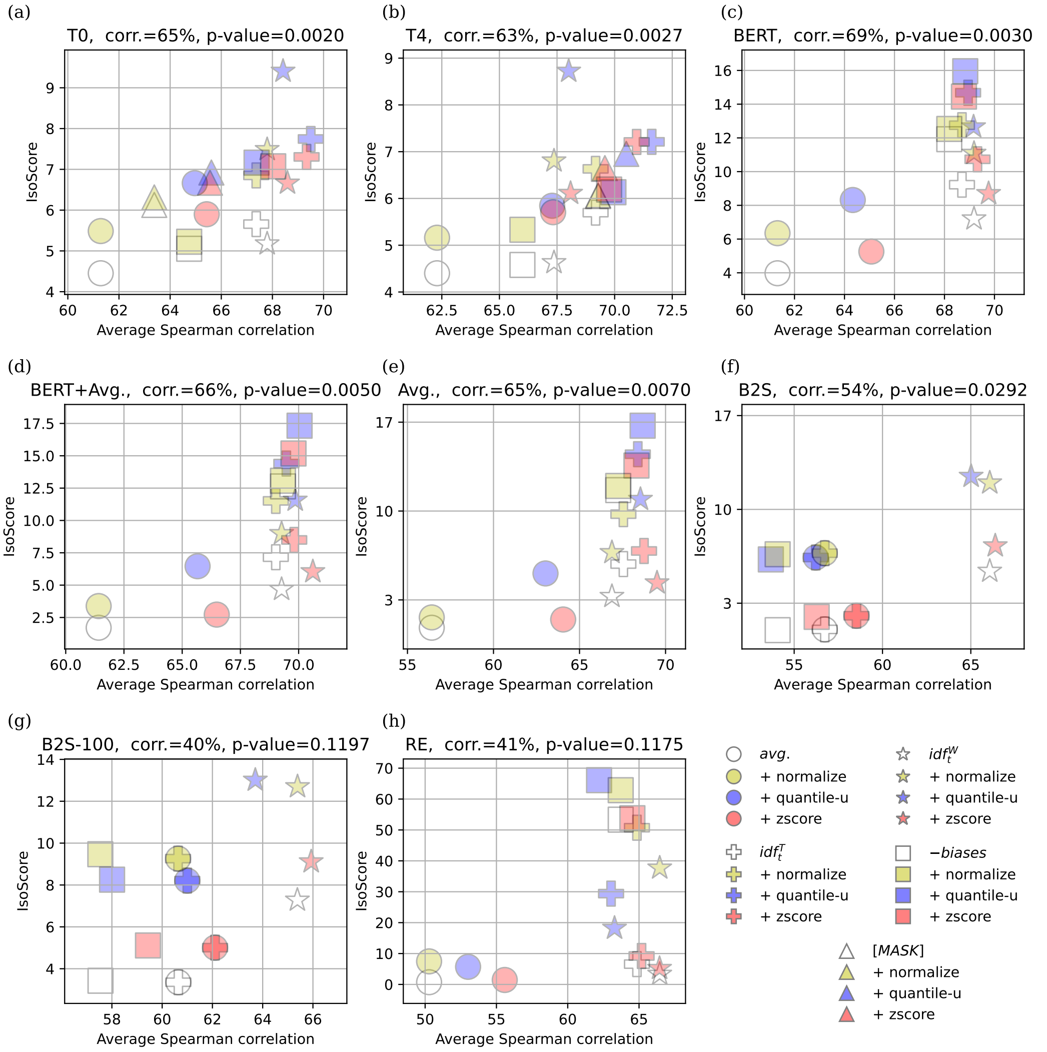

Figure 1: Relation between average Spearman correlation for STS tasks and IsoScore of Wikitext representations for each model. Pearson correlation coefficients are shown.

In contrast, the classification does not improve with token aggregation and post-processing techniques; therefore, we do not observe correlation (and do not show it here).

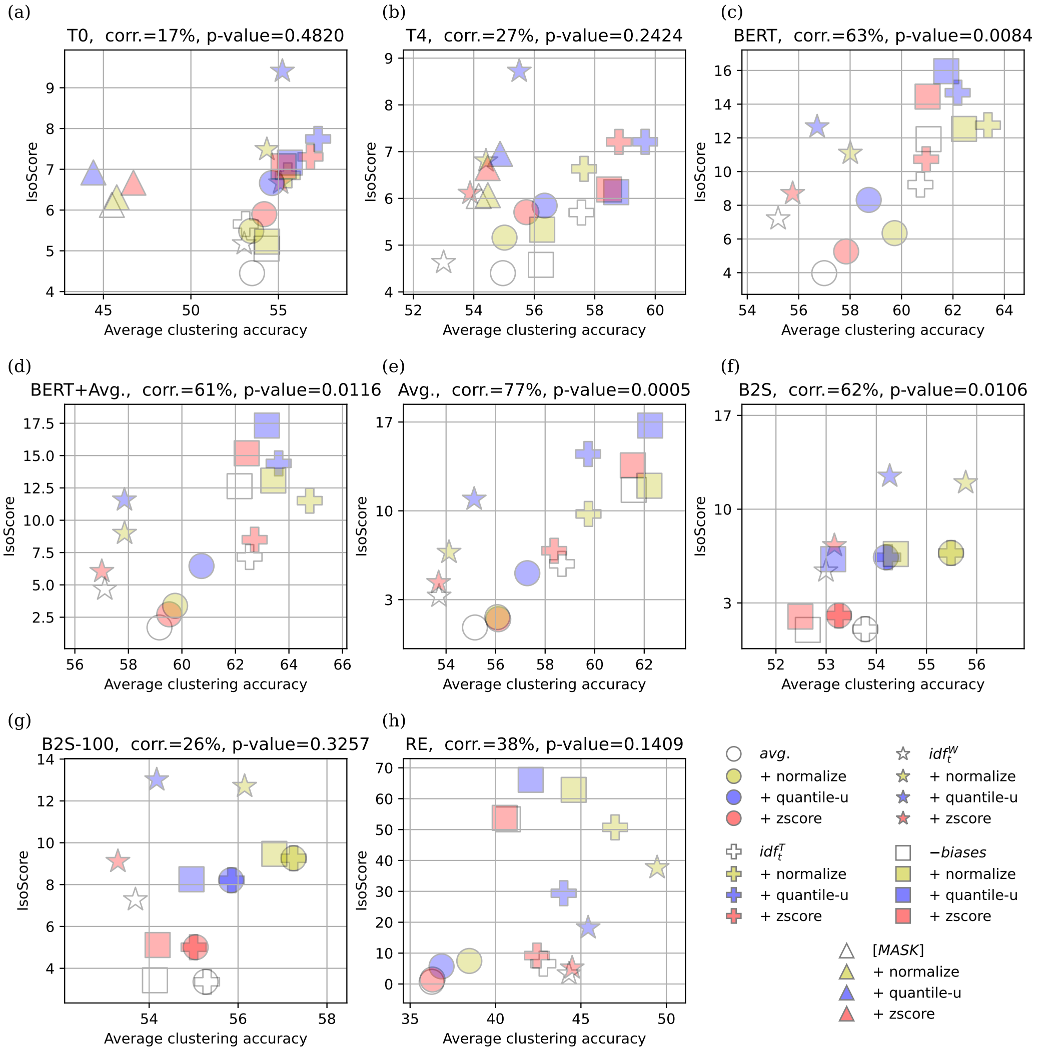

Figure 2: Relation between average clustering accuracy and IsoScore of Wikitext representations for each model. Pearson correlation coefficients are shown.

Table 4: Effect of different token aggregation and post-processing methods on multiple text embedding models. Both correlation and accuracy scores range from 0 to 100 (from the worst to the best). We also added results for SimCSE-BERT model [143], which is fine-tuned on NLI data supervision. The best result for each model is bolded, while underlined results are the best across all the models.

Model	**T0**	**T4**	**BERT**	**BERT + Avg.**	**Avg.**	**B2S**	**B2S-100**	**RE**
Layer	**Last**		**First + Last**			–		
Avg. Spearman correlation (%) of STS tasks								
avg.	61.3	62.3	61.3	61.4	56.4	56.7	60.6	50.3
idf_t^W	67.8	67.4	69.2	69.3	66.9	66.1	65.4	**66.4**
+ zscore	68.6	68.1	**69.8**	**70.6**	**69.5**	**66.4**	**65.9**	66.4
idf_t^T	67.4	69.2	68.7	69.0	67.5	56.7	60.6	64.9
+ quantile-u	**69.5**	**71.6**	68.9	69.5	68.4	56.2	61.0	63.0
-biases	64.7	66.0	68.2	69.3	67.3	54.1	57.5	63.7
`[MASK]`	63.4	69.3						
+ quantile-u	65.6	70.5						
SimCSE performance: 81.5								
Avg. accuracy (%) of clustering tasks								
avg.	53.5	55.0	57.0	59.2	55.2	53.8	55.3	36.3
idf_t^W	53.0	53.0	55.2	57.1	53.7	53.0	53.7	44.3
+ normalize	54.3	54.4	58.0	57.9	54.1	**55.8**	56.1	**49.5**
idf_t^T	53.1	57.6	60.7	62.5	58.7	53.8	55.3	42.8
+ quantile-u^W	**57.6**	**60.0**	63.1	64.4	60.2	54.6	56.0	44.4
+ normalize	55.5	57.6	**63.4**	**64.8**	59.8	55.5	**57.2**	47.0
-biases	54.3	56.2	61.1	62.2	61.6	52.6	54.1	40.7
+ normalize	54.4	56.3	62.4	63.4	**62.2**	54.4	56.8	44.6
`[MASK]`	45.5	54.2						
+ quantile-u	44.4	54.9						
SimCSE performance: 59.8 and 64.0 (avg. + quantile-u^W, first + last layers)								
Avg. accuracy (%) of classification tasks								
avg.	**80.5**	79.9	79.9	79.7	**77.3**	**76.9**	75.1	69.5
idf_t^W	73.1	67.1	78.3	77.9	75.3	76.8	74.1	67.6
idf_t^T	80.1	79.7	78.9	78.4	76.1	76.9	**75.1**	68.1
-biases	80.1	**79.9**	**80.1**	**79.8**	77.2	76.3	74.2	**70.3**
`[MASK]`	78.7	76.8						
SimCSE performance: 86.5								

Table 5: Performance of post-processing with plain token averaging. abtt-2 is the ABTT method with top 2 principal components removed. The best result for each model is bolded, while underlined results are the best across all the models.

Model	**T0**	**T4**	**BERT**	**BERT + Avg.**	**Avg.**	**B2S**	**B2S-100**	**RE**
Layer	**Last**		**First + Last**			–		
Avg. Spearman correlation (%) of STS tasks								
avg.	61.3	62.3	61.3	61.4	56.4	56.7	60.6	50.3
+ zscore	**65.4**	**67.3**	**65.1**	**66.5**	64.1	58.5	**62.1**	55.6
+ quantile-u	65.0	67.3	64.3	65.7	63.0	56.2	61.0	53.0
+ quantile-u^W	62.8	64.6	63.0	64.7	62.1	55.5	60.5	52.2
+ abtt-2	64.7	66.7	64.3	66.3	64.3	58.3	62.0	54.4
+ normalize	61.3	62.3	61.3	61.4	56.4	56.7	60.6	50.3
+ whiten	63.5	64.2	64.7	65.6	65.5	**62.1**	61.2	63.2
+ whiten^W	60.9	63.7	63.1	65.8	**67.0**	61.4	61.0	**64.3**
Avg. accuracy (%) of clustering tasks								
avg.	53.5	55.0	57.0	59.2	55.2	53.8	55.3	36.3
+ zscore	54.2	55.7	57.8	59.5	56.1	53.3	55.0	36.3
+ quantile-u	**54.6**	**56.3**	58.7	60.7	57.3	54.2	55.8	36.8
+ quantile-u^W	54.2	56.3	**60.1**	**61.1**	**58.6**	54.5	56.0	36.9
+ abtt-2	53.4	54.8	57.0	59.0	55.8	53.6	55.0	36.5
+ normalize	53.4	55.0	59.7	59.7	56.1	**55.5**	**57.2**	38.5
+ whiten	35.4	35.7	33.1	29.8	26.8	25.3	26.0	24.8
+ whiten^W	49.0	49.8	56.2	58.7	55.0	50.4	50.1	**41.5**

4.1.4 Alignment and Uniformity

We observed that token pooling and post-processing techniques do not improve the alignment and uniformity properties of the representations. Let us remind the reader that alignment is calculated with only those STS-B pairs with a similarity score of 5, while uniformity with all pairs, as defined in Section 3.5.5, and smaller values are better.

We can observe in Figure 3 that the z-score post-processing always makes alignment worse, while normalization of embeddings almost always does the same for the uniformity of the representations. Excluding the random embeddings model, the best alignment and uniformity properties are almost always with plain averaging and no post-processing applied.

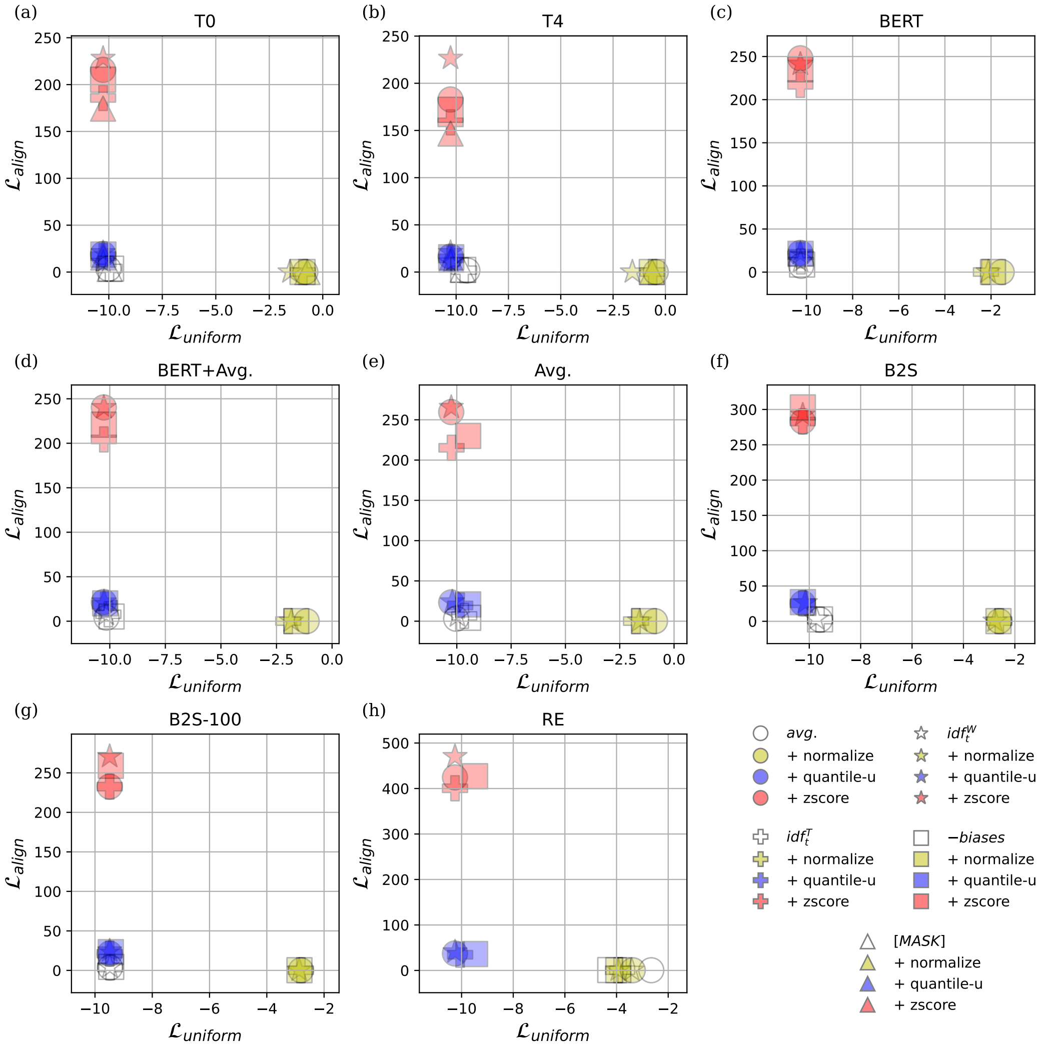

Figure 3: Alignment and uniformity of representations in relation to various token pooling and post-processing techniques. Lower values are better.

Both alignment and uniformity are sensitive to the scaling of the embedding vectors. Depending on the resulting scaling from the post-processing method, either one or the other is increased. However, decreasing them both using these methods is found to be difficult. This finding strengthens the reputation of these two metrics that only training the transformer model, as shown in [143], is capable of improving them.

As we mentioned, the outlier in its behavior here is the RE model, which improves its uniformity by different weighting of tokens or using quantile-uniform post-processing. Indeed, the latter post-processing is the least harmful for all models considered and disturbs alignment and uniformity properties less.

4.2 Using Prompts

The normal use of prompts, as presented in [15], is to place the sentence `[X]` inside a template as "`This sentence: "[X]" means [MASK]`" and use only the vector of `[MASK]` token from the final layer as the full representation of `[X]`. Our experiments show that in general, using prompts is beneficial; however, we found some ways to improve performance even more.

First, extracting only the vector of `[MASK]` is not necessarily the best. We found that a simple average of all tokens, including the ones from the prompt template, and also the `[MASK]` token, is still a valid approach. Even more, it outperforms only the `[MASK]` token approach for clustering and classification tasks, with corresponding 55.0 and 79.9 average accuracies versus 54.2 and 76.8 for using only the `[MASK]` token (see T4 model results in Table 4).

The use of the additional text template around the text enriches the target text representation. Let us compare the performance of 12th layer averaged token representations with and without a template (see Figure 4, T0 avg. and T4 avg. with a template versus BERT avg. without). For all 3 groups of tasks—semantic textual similarity, clustering, and classification—the T0 template achieves 61.3 average Spearman correlation, 53.5 clustering accuracy, and 80.5 classification accuracy; the T4 template reaches 62.3, 55.0, and 79.9, while a regular, non-templated text obtains just 53.2, 50.6, and 79.0, respectively. This shows that the use of the templates allows the model to enrich target text representation. However, this effect peaks at the last, 12th layer, and the achieved performance is similar to the first + last layer combination of the regular non-templated vectors.

Performance of using only the `[MASK]` token also peaks at the 12th layer, so we investigated what influence it has in the enrichment of templated target text representation. Could it be that this token is the main culprit for better performance of averaged templated text representations? To answer this question, we tried to omit the `[MASK]` token from averaging (see Figure 4 T0 avg. no `[MASK]` and T4 avg. no `[MASK]`). For the T4 template and 12th layer representations, averaging all tokens yields 62.3 average Spearman correlation, 55.0 clustering accuracy, and 79.9 classification accuracy; dropping the `[MASK]` token from averaging yields 58.7, 53.3, and 79.8; non-templated performance is 53.2, 50.6, and 79.0, respectively. We see that omitting the `[MASK]` token in averaging indeed hurts the performance. On the other hand, the results show that it is responsible only for approximately half of the improvements, with the other half coming from the other tokens in the templated text.

One good reason to use averaging instead of only the `[MASK]` token is that then token weighting can be also applied. As we already showed in Table 4), the T4 template together with idf_t^T token weighting and uniform quantile post-processing allowed us to reach the average Spearman correlation of 71.6, which was the best among the tried methods. Also, we observed that post-processing on text representations from `[MASK]` token was not as effective as from averaged ones. This also suggests that all tokens in templated texts have richer representations.

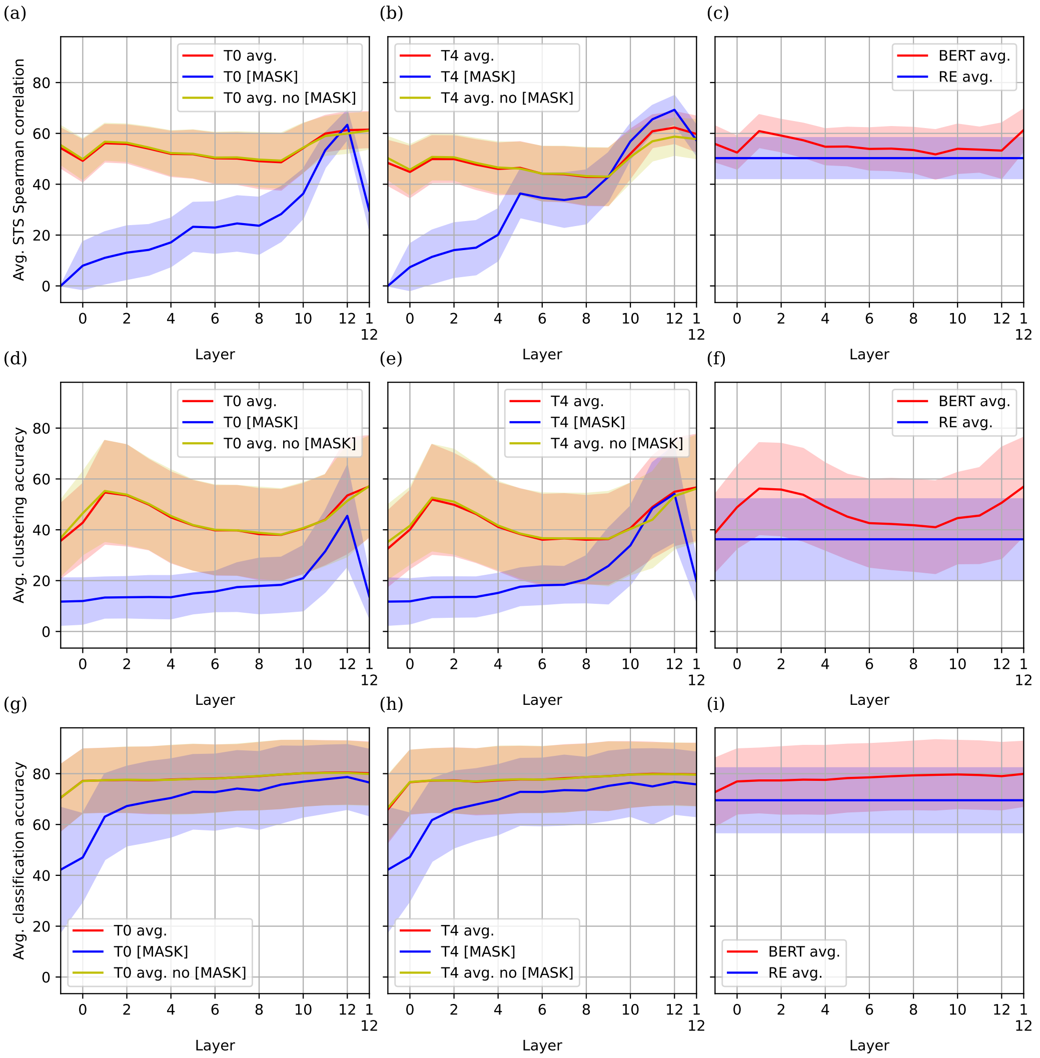

Figure 4: Layer-wise performance of templated models T0 (subfigures a, d, g) and T4 (b, e, h), as well as BERT versus RE with no weighting or post-processing (c, f, i). The average performance of STS (a, b, c), clustering (d, e, f), and classification tasks (g, h, i) is shown by the lines, while shadow areas correspond to the standard deviation. We also show first + last aggregation over layers as the last tick $\frac{1}{12}$ on the horizontal axis.

4.3 BERT + Avg. Model

One of the ways we sought to improve BERT transformer model representations is to combine embeddings of the regular BERT with the ones averaged over multiple contexts. That is, for each token in each BERT layer, we collected many different contexts, and the averaged vector became the vector of the Avg. model. We then combined BERT and Avg. model, according to the parameter w, which shows the fraction of Avg. model representations in the resulting vector v:

$$v = v_{BERT}(1 - w) + v_{Avg.}w. \tag{10}$$

We have also varied the w parameter to the negative values in (10) to see if subtracting (instead of adding) the context-average representations from the context-aware ones helps.

The impact of the w parameter and the choice of the layer to source the representations (same layer for both BERT and Avg.) is presented in Figure 5.

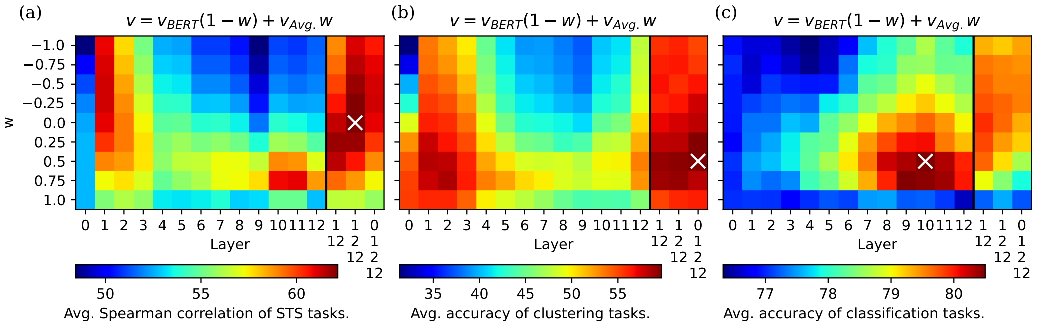

Figure 5: BERT + Avg. model performance dependence on the weight w of Avg. model and layer, from which (for both models) representations are used. To the right of the black line on the horizontal axis, average aggregation of multiple layers is also shown. Tokens are simply averaged and no post-processing is used. The horizontal line with $w = 0.0$ corresponds to a regular Bert (B) model, $w = 0.5$ is B + Avg., and $w = 1.0$ is the Avg. model. The white × marks the maximum value.

One can see that for clustering and classification tasks, the combination of both models in equal portions of $w = 0.5$ is better than these models alone ($w = 0.0$ corresponds to BERT, $w = 1.0$ is a single Avg. model). If we look at the best scores with the two techniques applied in Table 4, for the average clustering accuracy this translates to an improvement of 1.4% from the 63.4% of BERT to 64.8% of BERT + Avg.

Meanwhile, as we see in the same table, the average accuracy of classification tasks without additional techniques applied to the combination of the first and last layers is similar between only BERT (79.9%) and BERT + Avg. (79.7%). However, as we see in Figure 5, differently from the first + last optimal layer setting for BERT, BERT + Avg. has a sweet spot in the 10th layer with an average classification accuracy of 80.5%, surpassing that of BERT by 0.6%.

These results suggest a conclusion that for clustering and classification tasks, combining a regular BERT token with the same one but averaged over multiple contexts is beneficial.

As we already mentioned, STS tasks preferred the regular BERT ($w = 0$) to the BERT + Avg. on average. However, for several individual semantic textual similarity datasets, the best weights turned out to be even negative. This is the case for STS15 ($w = -0.25$) and STS16 ($w = -0.75$). The same negative weight of $w = -0.5$ preference was also observed for the searchsnippets clustering dataset. Although the Spearman correlation for three STS tasks was only up to several percent higher, for searchsnippets, the clustering accuracy increased to a staggering 80.2% from 72.2% of $w = 0.0$. This shows that the determined values of $w = 0.0$ for STS and $w = 0.5$ for clustering and classification are not universal, and for a small percentage of tasks, they can differ. For more BERT + Avg. task-wise details, see Appendix A.2 figures for STS (Figure 6), clustering (Figure 7), and classification (Figure 8) performance.

Layers

Figure 5 also depicts the task performance versus BERT layers. Our first observation is that for STS and clustering tasks, there is a strong preference for the first layers. As a result, the best combination of layers also involves the first ones; for semantic textual similarity tasks, it is the average of representations from one, two, and twelve layers, and for clustering, zero, one, two, and twelve layers. Even individual SICK-E, SICK-R, and STS-B classification tasks, which originate from semantic textual similarity ones, have strong first layers (see Figure 8 in Appendix A.2), although on average, the best layer for classification tasks is the tenth. This shows that a lot of performance for tasks that work on similarities between texts (STS and clustering) depends on the first layers, which, according to [234], have more generalized token representations. However, the last layer, which receives recreated token identities [234] is also important, as the best combinations (such as 1 + 2 + 12) also involve it. This first + last cooperation of layers can be distinctly observed as a U shape for clustering performance in layers (see Figure 5 and also Figure 4).

For most classification tasks (excluding those originating from STS), the U shape is inversed. This is very clearly seen for binary classification tasks (see Figure 8 in Appendix A.2), where the highest classification accuracy is concentrated between the ninth and the eleventh layers. This aligns with the explanations of the previous work [100], which argues that the last layer is over-specialized for the training objective.

5 Conclusions

We empirically evaluated the effects of various aggregation and post-processing techniques of token representations in a trained transformer and other models to form good text-, paragraph-, or sentence-level embeddings. We carried out the empirical evaluation of the embeddings on three classes of downstream text-level tasks: Semantic Textual Similarity (STS), clustering, and classification.

We found the techniques to benefit all models studied for the unsupervised STS (the best model average Spearman correlation increased from 62.3% to 71.6%) and clustering (the best model average clustering accuracy increased from 59.2% to 64.8%), while it had no positive effect on the supervised classification tasks (see Table 4).

We present a strong and very simple baseline model of Random Embeddings (RE), where every token is assigned a random vector as its embedding. Combined with token aggregation and post-processing techniques, it also almost matches the average STS performance of the BERT model with the techniques applied, with 66.4% versus 69.8% average Spearman correlation. It also shows very high performance for some tasks, like stackoverflow classification, where BERT token contextualization may not work well on code samples in the texts (see Section 4.1.2 for more details). We encourage future work to use RE as a baseline, due to its mid-level performance, simple implementation, and ability to separate the contribution to the performance of the learned contexts from the aggregation and post-processing techniques.

We found that the aggregation and post-processing techniques tried typically increase the isotropy of the representations, and the isotropy for most models is positively correlated (up to 69% Pearson correlation) with the Spearman correlation of STS tasks (Section 4.1.3). The highest isotropy improvement is observed for our Random Embeddings model, since its token representations have the maximal isotropy to start with. We did not find the token aggregation and post-processing techniques to improve the alignment and uniformity properties of representations.

We question the use of prompts (adding a sentence into a certain text template) for retrieving the representation of the sentence from only the [MASK] token. Our experiments show that averaging all the templated text tokens, with idf weighting and post-processing for the unsupervised tasks, is better. Meanwhile, the average increase in performance due to the added template is only obvious for the STS tasks, gives no improvement in clustering, and is very slight in classification (see Table 4, Section 4.2).

We presented a static vector model Avg., which simply contains BERT tokens averaged over multiple different contexts. Our experiments show that it outperforms a more complex BERT2Static [49], also a static word-level model, yet specially trained on BERT contexts. With the best post-processing and token aggregation techniques, the advantage for unsupervised STS tasks is 69.5 versus 66.4, and for clustering, it is 62.2 versus 57.2., with a negligible difference for classification tasks. Moreover, we show that combining Avg. with the parent BERT model can bring even further improvements. In particular, BERT + Avg. reached the highest average clustering accuracy of 64.8 out of all our considered models, as well as the classification accuracy of 80.5 (Section 4.3). We encourage future work to also use Avg. as a baseline, both due to its upper-level performance and simple implementation.

In our work, we also analyzed prompt and BERT + Avg. models layer-wise. We found that for the STS tasks, taking the representations from the first layers performs better, and for the clustering task, the performance profile forms a "U" shape with tops at the first and last layers. Therefore, for these two task groups, we mostly use the average of first + last layers, harnessing both of the tops for the best performance. On the other hand, classification tasks have an inverted "U" shape with the top in the 10th layer (Figure 5). We did not find token aggregation and post-processing techniques to change such profile curvature.

In this research, we used a pre-trained BERT as a manageable representative of transformer models, but the findings should be transferable to other types of transformers, including large language models. We also specifically did not perform task-specific fine-tuning of the model to keep it universal. This enables the same model to be used in multiple ways. For example, given a prompt, its text-level embedding can be extracted from the model using the techniques investigated here, and this embedding can be used to search an external database for related information to add the prompt to the same model, i.e., we can use the same model for both query encoding and generation in retrieval-augmented generation [235]. Alternatively, one of the simpler baseline models proposed here could be used as the query encoder.

References

[1] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in Neural Information Processing Systems*, C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger, Eds., vol. 26. Curran Associates, Inc., 2013. [Online]. Available: https://proceedings.neurips.cc/paper/2013/file/9aa42b31882ec039965f3c4923ce901b-Paper.pdf

[2] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[3] B. Hart and T. Risley, "The early catastrophe: The 30 million word gap by age 3," *American Educator*, vol. 27, pp. 4–9, 01 2003.

[4] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: http://arxiv.org/abs/1409.0473

[5] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 4171–4186. [Online]. Available: https://aclanthology.org/N19-1423

[6] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," in *International Conference on Learning Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=YicbFdNTTy

[7] A. Gulati, J. Qin, C.-C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Z. Zhang, Y. Wu, and R. Pang, "Conformer: Convolution-augmented transformer for speech recognition," 2020. [Online]. Available: https://arxiv.org/abs/2005.08100

[8] L. Chen, K. Lu, A. Rajeswaran, K. Lee, A. Grover, M. Laskin, P. Abbeel, A. Srinivas, and I. Mordatch, "Decision transformer: Reinforcement learning via sequence modeling," in *Advances in Neural Information Processing Systems*, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., 2021. [Online]. Available: https://openreview.net/forum?id=a7APmM4B9d

[9] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence embeddings using Siamese BERT-networks," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 3982–3992. [Online]. Available: https://aclanthology.org/D19-1410

[10] S. Narang, H. W. Chung, Y. Tay, L. Fedus, T. Fevry, M. Matena, K. Malkan, N. Fiedel, N. Shazeer, Z. Lan, Y. Zhou, W. Li, N. Ding, J. Marcus, A. Roberts, and C. Raffel, "Do transformer modifications transfer across implementations and applications?" in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 5758–5773. [Online]. Available: https://aclanthology.org/2021.emnlp-main.465

[11] Y. Belinkov, "Probing Classifiers: Promises, Shortcomings, and Advances," *Computational Linguistics*, vol. 48, no. 1, pp. 207–219, 04 2022. [Online]. Available: https://doi.org/10.1162/coli_a_00422

[12] A. Conneau, D. Kiela, H. Schwenk, L. Barrault, and A. Bordes, "Supervised learning of universal sentence representations from natural language inference data," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark: Association for Computational Linguistics, Sep. 2017, pp. 670–680. [Online]. Available: https://www.aclweb.org/anthology/D17-1070

[13] J. Su, J. Cao, W. Liu, and Y. Ou, "Whitening sentence representations for better semantics and faster retrieval," 2021. [Online]. Available: https://arxiv.org/abs/2103.15316

[14] J. Huang, D. Tang, W. Zhong, S. Lu, L. Shou, M. Gong, D. Jiang, and N. Duan, "WhiteningBERT: An easy unsupervised sentence embedding approach," in *Findings of the Association for Computational Linguistics: EMNLP 2021*. Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 238–244. [Online]. Available: https://aclanthology.org/2021.findings-emnlp.23

[15] T. Jiang, J. Jiao, S. Huang, Z. Zhang, D. Wang, F. Zhuang, F. Wei, H. Huang, D. Deng, and Q. Zhang, "PromptBERT: Improving BERT sentence embeddings with prompts," in *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. Abu Dhabi, United Arab Emirates: Association for Computational Linguistics, Dec. 2022, pp. 8826–8837. [Online]. Available: https://aclanthology.org/2022.emnlp-main.603

[16] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013. [Online]. Available: https://arxiv.org/abs/1301.3781

[17] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543. [Online]. Available: http://www.aclweb.org/anthology/D14-1162

[18] F. J. Pelletier, "The principle of semantic compositionality," *Topoi*, vol. 13, no. 1, pp. 11–24, 1994.

[19] J. Bos, "Wide-coverage semantic analysis with Boxer," in *Semantics in Text Processing. STEP 2008 Conference Proceedings*. College Publications, 2008, pp. 277–286. [Online]. Available: https://aclanthology.org/W08-2222

[20] R. MONTAGUE, "Universal grammar," *Theoria*, vol. 36, no. 3, pp. 373–398, 1970. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1755-2567.1970.tb00434.x

[21] J. Bos, "A survey of computational semantics: Representation, inference and knowledge in wide-coverage text understanding," *Language and Linguistics Compass*, vol. 5, no. 6, pp. 336–366, 2011. [Online]. Available: https://compass.onlinelibrary.wiley.com/doi/abs/10.1111/j.1749-818X.2011.00284.x

[22] K. Erk, "Vector space models of word meaning and phrase meaning: A survey," *Language and Linguistics Compass*, vol. 6, no. 10, pp. 635–653, 2012. [Online]. Available: https://compass.onlinelibrary.wiley.com/doi/abs/10.1002/lnco.362

[23] M. Yoshikawa, K. Mineshima, H. Noji, and D. Bekki, "Combining axiom injection and knowledge base completion for efficient natural language inference," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, Jul. 2019, pp. 7410–7417. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/4730

[24] S. Clark and S. G. Pulman, "Combining symbolic and distributional models of meaning," in *AAAI Spring Symposium: Quantum Interaction*, 2007.

[25] J. Bjerva, J. Bos, R. van der Goot, and M. Nissim, "The meaning factory: Formal semantics for recognizing textual entailment and determining semantic similarity," in *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*. Dublin, Ireland: Association for Computational Linguistics, Aug. 2014, pp. 642–646. [Online]. Available: https://aclanthology.org/S14-2114

[26] P. Smolensky, "Tensor product variable binding and the representation of symbolic structures in connectionist systems," *Artificial Intelligence*, vol. 46, no. 1, pp. 159–216, 1990. [Online]. Available: https://www.sciencedirect.com/science/article/pii/000437029090007M

[27] J. Mitchell and M. Lapata, "Composition in distributional models of semantics," *Cognitive Science*, vol. 34, no. 8, pp. 1388–1429, 2010. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1551-6709.2010.01106.x

[28] D. Milajevs, D. Kartsaklis, M. Sadrzadeh, and M. Purver, "Evaluating neural word representations in tensor-based compositional settings," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 708–719. [Online]. Available: https://aclanthology.org/D14-1079

[29] M. N. Jones and D. J. Mewhort, "Representing word meaning and order information in a composite holographic lexicon." *Psychological review*, vol. 114, no. 1, p. 1, 2007.

[30] D. Widdows, "Semantic vector products: Some initial investigations," in *Proceedings of the Second AAAI Symposium on Quantum Interaction*, 2008.

[31] T. Plate, "Holographic reduced representations," *IEEE Transactions on Neural Networks*, vol. 6, no. 3, pp. 623–641, 1995.

[32] M. Baroni and R. Zamparelli, "Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space," in *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*. Cambridge, MA: Association for Computational Linguistics, Oct. 2010, pp. 1183–1193. [Online]. Available: https://aclanthology.org/D10-1115

[33] E. Guevara, "A regression model of adjective-noun compositionality in distributional semantics," in *Proceedings of the 2010 Workshop on GEometrical Models of Natural Language Semantics*. Uppsala, Sweden: Association for Computational Linguistics, Jul. 2010, pp. 33–37. [Online]. Available: https://aclanthology.org/W10-2805

[34] E. Grefenstette, G. Dinu, Y. Zhang, M. Sadrzadeh, and M. Baroni, "Multi-step regression learning for compositional distributional semantics," in *Proceedings of the 10th International Conference on Computational Semantics (IWCS 2013) – Long Papers*. Potsdam, Germany: Association for Computational Linguistics, Mar. 2013, pp. 131–142. [Online]. Available: https://aclanthology.org/W13-0112

[35] M. Baroni, R. Bernardi, and R. Zamparelli, "Frege in space: A program for composition distributional semantics," in *Linguistic Issues in Language Technology, Volume 9, 2014 - Perspectives on Semantic Representations for Textual Inference*. CSLI Publications, 2014. [Online]. Available: https://aclanthology.org/2014.lilt-9.5

[36] C. Xing, D. Wang, X. Zhang, and C. Liu, "Document classification with distributions of word vectors," in *Signal and Information Processing Association Annual Summit and Conference (APSIPA), 2014 Asia-Pacific*, 2014, pp. 1–5.

[37] L. Yu, K. M. Hermann, P. Blunsom, and S. Pulman, "Deep learning for answer sentence selection," 2014. [Online]. Available: https://arxiv.org/abs/1412.1632

[38] G. Lev, B. Klein, and L. Wolf, "In defense of word embedding for generic text representation," in *Natural Language Processing and Information Systems*, C. Biemann, S. Handschuh, A. Freitas, F. Meziane, and E. Métais, Eds. Cham: Springer International Publishing, 2015, pp. 35–50.

[39] S. Ritter, C. Long, D. Paperno, M. Baroni, M. Botvinick, and A. Goldberg, "Leveraging preposition ambiguity to assess compositional distributional models of semantics," in *Proceedings of the Fourth Joint Conference on Lexical and Computational Semantics*. Denver, Colorado: Association for Computational Linguistics, Jun. 2015, pp. 199–204. [Online]. Available: https://aclanthology.org/S15-1023

[40] L. White, R. Togneri, W. Liu, and M. Bennamoun, "How well sentence embeddings capture meaning," in *Proceedings of the 20th Australasian Document Computing Symposium*, ser. ADCS '15. New York, NY, USA: Association for Computing Machinery, 2015. [Online]. Available: https://doi.org/10.1145/2838931.2838932

[41] D. Shen, G. Wang, W. Wang, M. R. Min, Q. Su, Y. Zhang, C. Li, R. Henao, and L. Carin, "Baseline needs more love: On simple word-embedding-based models and associated pooling mechanisms," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia: Association for Computational Linguistics, Jul. 2018, pp. 440–450. [Online]. Available: https://aclanthology.org/P18-1041

[42] J. Wieting, M. Bansal, K. Gimpel, and K. Livescu, "Towards universal paraphrastic sentence embeddings," 2015. [Online]. Available: https://arxiv.org/abs/1511.08198

[43] H. Aldarmaki and M. Diab, "Evaluation of unsupervised compositional representations," in *Proceedings of the 27th International Conference on Computational Linguistics*. Santa Fe, New Mexico, USA: Association for Computational Linguistics, Aug. 2018, pp. 2666–2677. [Online]. Available: https://aclanthology.org/C18-1226

[44] M. Iyyer, V. Manjunatha, J. Boyd-Graber, and H. Daumé III, "Deep unordered composition rivals syntactic methods for text classification," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Beijing, China: Association for Computational Linguistics, Jul. 2015, pp. 1681–1691. [Online]. Available: https://aclanthology.org/P15-1162

[45] N. T. Pham, G. Kruszewski, A. Lazaridou, and M. Baroni, "Jointly optimizing word representations for lexical and sentential tasks with the C-PHRASE model," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Beijing, China: Association for Computational Linguistics, Jul. 2015, pp. 971–981. [Online]. Available: https://aclanthology.org/P15-1094

[46] J. Wieting and K. Gimpel, "Revisiting recurrent networks for paraphrastic sentence embeddings," 2017. [Online]. Available: https://arxiv.org/abs/1705.00364

[47] T. Kenter, A. Borisov, and M. de Rijke, "Siamese CBOW: Optimizing word embeddings for sentence representations," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 941–951. [Online]. Available: https://aclanthology.org/P16-1089

[48] M. Pagliardini, P. Gupta, and M. Jaggi, "Unsupervised learning of sentence embeddings using compositional n-gram features," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, Jun. 2018, pp. 528–540. [Online]. Available: https://aclanthology.org/N18-1049

[49] P. Gupta and M. Jaggi, "Obtaining better static word embeddings using contextual embedding models," in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Online: Association for Computational Linguistics, Aug. 2021, pp. 5241–5253. [Online]. Available: https://aclanthology.org/2021.acl-long.408

[50] F. Hill, K. Cho, A. Korhonen, and Y. Bengio, "Learning to understand phrases by embedding the dictionary," *Transactions of the Association for Computational Linguistics*, vol. 4, pp. 17–30, 2016. [Online]. Available: https://aclanthology.org/Q16-1002

[51] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, "Bag of tricks for efficient text classification," in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. Valencia, Spain: Association for Computational Linguistics, Apr. 2017, pp. 427–431. [Online]. Available: https://aclanthology.org/E17-2068

[52] H. Choi, J. Kim, S. Joe, and Y. Gwon, "Evaluation of bert and albert sentence embedding performance on downstream nlp tasks," in *2020 25th International Conference on Pattern Recognition (ICPR)*, 2021, pp. 5482–5487.

[53] R. Bommasani, K. Davis, and C. Cardie, "Interpreting Pretrained Contextualized Representations via Reductions to Static Embeddings," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, Jul. 2020, pp. 4758–4781. [Online]. Available: https://aclanthology.org/2020.acl-main.431

[54] W. Yin and H. Schütze, "Learning word meta-embeddings," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 1351–1360. [Online]. Available: https://aclanthology.org/P16-1128

[55] J. Coates and D. Bollegala, “Frustratingly easy meta-embedding – computing meta-embeddings by averaging source word embeddings,” in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, Jun. 2018, pp. 194–198. [Online]. Available: https://aclanthology.org/N18-2031

[56] Y. Adi, E. Kermany, Y. Belinkov, O. Lavi, and Y. Goldberg, “Fine-grained analysis of sentence embeddings using auxiliary prediction tasks,” in *5th International Conference on Learning Representations*, 2017. [Online]. Available: https://openreview.net/forum?id=BJh6Ztuxl

[57] C. E. Shannon, “A mathematical theory of communication,” *The Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, 1948.

[58] B. Riedel, I. Augenstein, G. P. Spithourakis, and S. Riedel, “A simple but tough-to-beat baseline for the fake news challenge stance detection task,” 2017. [Online]. Available: https://arxiv.org/abs/1707.03264

[59] P. Singh and A. Mukerjee, “Words are not equal: Graded weighting model for building composite document vectors,” in *Proceedings of the 12th International Conference on Natural Language Processing*. Trivandrum, India: NLP Association of India, Dec. 2015, pp. 11–19. [Online]. Available: https://aclanthology.org/W15-5903

[60] S. Arora, Y. Liang, and T. Ma, “A simple but tough-to-beat baseline for sentence embeddings,” in *5th International Conference on Learning Representations, ICLR 2017*, 2017.

[61] K. Ethayarajh, “Unsupervised random walk sentence embeddings: A strong but simple baseline,” in *Proceedings of The Third Workshop on Representation Learning for NLP*. Melbourne, Australia: Association for Computational Linguistics, Jul. 2018, pp. 91–100. [Online]. Available: https://aclanthology.org/W18-3012

[62] L. Stankevičius and M. Lukoševičius, “Testing pre-trained transformer models for lithuanian news clustering,” in *CEUR workshop proceeding: IVUS 2020*, vol. 2698. CEUR-WS, 2020, pp. 46–53.

[63] Y. Yan, R. Li, S. Wang, F. Zhang, W. Wu, and W. Xu, “ConSERT: A contrastive framework for self-supervised sentence representation transfer,” in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Online: Association for Computational Linguistics, Aug. 2021, pp. 5065–5075. [Online]. Available: https://aclanthology.org/2021.acl-long.393

[64] Y. An, A. Kalinowski, and J. Greenberg, “Clustering and network analysis for the embedding spaces of sentences and sub-sentences,” in *2021 Second International Conference on Intelligent Data Science Technologies and Applications (IDSTA)*, 2021, pp. 138–145.

[65] Z. Yang, C. Zhu, and W. Chen, “Parameter-free sentence embedding via orthogonal basis,” in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 638–648. [Online]. Available: https://aclanthology.org/D19-1059

[66] B. Wang and C.-C. J. Kuo, “Sbert-wk: A sentence embedding method by dissecting bert-based word models,” *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, vol. 28, p. 2146–2157, jan 2020. [Online]. Available: https://doi.org/10.1109/TASLP.2020.3008390

[67] A. M. J. Schakel and B. J. Wilson, “Measuring word significance using distributed representations of words,” 2015. [Online]. Available: https://arxiv.org/abs/1508.02297

[68] N. Arefyev, P. Ermolaev, and A. Panchenko, “How much does a word weigh? weighting word embeddings for word sense induction,” 2018. [Online]. Available: https://arxiv.org/abs/1805.09209

[69] H. P. Luhn, “The automatic creation of literature abstracts,” *IBM Journal of Research and Development*, vol. 2, no. 2, pp. 159–165, 1958.

[70] S. Yokoi, R. Takahashi, R. Akama, J. Suzuki, and K. Inui, “Word rotator’s distance,” in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics, Nov. 2020, pp. 2944–2960. [Online]. Available: https://aclanthology.org/2020.emnlp-main.236

[71] H. Amiri and M. Mohtarami, “Vector of locally aggregated embeddings for text representation,” in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 1408–1414. [Online]. Available: https://aclanthology.org/N19-1143

[72] V. Gupta, H. Karnick, A. Bansal, and P. Jhala, “Product classification in E-commerce using distributional semantics,” in *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. Osaka, Japan: The COLING 2016 Organizing Committee, Dec. 2016, pp. 536–546. [Online]. Available: https://aclanthology.org/C16-1052

[73] H. K. Kim, H. Kim, and S. Cho, "Bag-of-concepts: Comprehending document representation through clustering words in distributed representation," *Neurocomputing*, vol. 266, pp. 336–352, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231217308962

[74] S. Guo and N. Yao, "Document vector extension for documents classification," *IEEE Transactions on Knowledge and Data Engineering*, vol. 33, no. 8, pp. 3062–3074, 2021.

[75] M. Li, H. Bai, L. Tan, K. Xiong, M. Li, and J. Lin, "Latte-mix: Measuring sentence semantic similarity with latent categorical mixtures," 2020. [Online]. Available: https://arxiv.org/abs/2010.11351

[76] D. Mekala, V. Gupta, B. Paranjape, and H. Karnick, "SCDV : Sparse composite document vectors using soft clustering over distributional representations," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark: Association for Computational Linguistics, Sep. 2017, pp. 659–669. [Online]. Available: https://aclanthology.org/D17-1069

[77] V. Gupta, A. Saw, P. Nokhiz, P. Netrapalli, P. Rai, and P. Talukdar, "P-sif: Document embeddings using partition averaging," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 05, 2020, pp. 7863–7870.

[78] M. Aharon, M. Elad, and A. Bruckstein, "K-svd: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Transactions on Signal Processing*, vol. 54, no. 11, pp. 4311–4322, 2006.

[79] V. Gupta, A. Saw, P. Nokhiz, H. Gupta, and P. Talukdar, "Improving document classification with multi-sense embeddings," in *Proceedings of the European Conference on Artificial Intelligence*, 2020.

[80] A. Gupta and V. Gupta, "Unsupervised contextualized document representation," in *Proceedings of the Second Workshop on Simple and Efficient Natural Language Processing*. Virtual: Association for Computational Linguistics, Nov. 2021, pp. 166–173. [Online]. Available: https://aclanthology.org/2021.sustainlp-1.17

[81] D. Mekala and J. Shang, "Contextualized weak supervision for text classification," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, Jul. 2020, pp. 323–333. [Online]. Available: https://aclanthology.org/2020.acl-main.30

[82] N. Ahmed, T. Natarajan, and K. Rao, "Discrete cosine transform," *IEEE Transactions on Computers*, vol. C-23, no. 1, pp. 90–93, 1974.

[83] N. Almarwani, H. Aldarmaki, and M. Diab, "Efficient sentence embedding using discrete cosine transform," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 3672–3678. [Online]. Available: https://aclanthology.org/D19-1380

[84] N. Almarwani and M. Diab, "Discrete cosine transform as universal sentence encoder," in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. Online: Association for Computational Linguistics, Aug. 2021, pp. 419–426. [Online]. Available: https://aclanthology.org/2021.acl-short.53

[85] S. Kayal and G. Tsatsaronis, "EigenSent: Spectral sentence embeddings using higher-order dynamic mode decomposition," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 4536–4546. [Online]. Available: https://aclanthology.org/P19-1445

[86] T. Kim, K. M. Yoo, and S.-g. Lee, "Self-guided contrastive learning for BERT sentence representations," in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Online: Association for Computational Linguistics, Aug. 2021, pp. 2528–2540. [Online]. Available: https://aclanthology.org/2021.acl-long.197

[87] H. Tanaka, H. Shinnou, R. Cao, J. Bai, and W. Ma, "Document classification by word embeddings of bert," in *Computational Linguistics*, L.-M. Nguyen, X.-H. Phan, K. Hasida, and S. Tojo, Eds. Singapore: Springer Singapore, 2020, pp. 145–154.

[88] X. Ma, Z. Wang, P. Ng, R. Nallapati, and B. Xiang, "Universal text representation from bert: An empirical study," 2019. [Online]. Available: https://arxiv.org/abs/1910.07973

[89] O. Kovaleva, A. Romanov, A. Rogers, and A. Rumshisky, "Revealing the dark secrets of BERT," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 4365–4374. [Online]. Available: https://aclanthology.org/D19-1445

[90] P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, and G. Neubig, "Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing," 2021. [Online]. Available: https://arxiv.org/abs/2107.13586

[91] Z. Zhong, D. Friedman, and D. Chen, "Factual probing is [MASK]: Learning vs. learning to recall," in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Online: Association for Computational Linguistics, Jun. 2021, pp. 5017–5033. [Online]. Available: https://aclanthology.org/2021.naacl-main.398

[92] H. Wang, Y. Li, Z. Huang, Y. Dou, L. Kong, and J. Shao, "Sncse: Contrastive learning for unsupervised sentence embedding with soft negative samples," 2022. [Online]. Available: https://arxiv.org/abs/2201.05979

[93] F. Dalvi, A. R. Khan, F. Alam, N. Durrani, J. Xu, and H. Sajjad, "Discovering latent concepts learned in BERT," in *International Conference on Learning Representations*, 2022. [Online]. Available: https://openreview.net/forum?id=POTMtpYI1xH

[94] N. F. Liu, M. Gardner, Y. Belinkov, M. E. Peters, and N. A. Smith, "Linguistic knowledge and transferability of contextual representations," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 1073–1094. [Online]. Available: https://aclanthology.org/N19-1112

[95] I. Tenney, P. Xia, B. Chen, A. Wang, A. Poliak, R. T. McCoy, N. Kim, B. V. Durme, S. Bowman, D. Das, and E. Pavlick, "What do you learn from context? probing for sentence structure in contextualized word representations," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=SJzSgnRcKX

[96] B. Muller, Y. Elazar, B. Sagot, and D. Seddah, "First align, then predict: Understanding the cross-lingual ability of multilingual BERT," in *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*. Online: Association for Computational Linguistics, Apr. 2021, pp. 2214–2231. [Online]. Available: https://aclanthology.org/2021.eacl-main.189

[97] K. Hämmerl, J. Libovický, and A. Fraser, "Combining static and contextualised multilingual embeddings," in *Findings of the Association for Computational Linguistics: ACL 2022*. Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 2316–2329. [Online]. Available: https://aclanthology.org/2022.findings-acl.182

[98] K. Ethayarajh, "How contextual are contextualized word representations? Comparing the geometry of BERT, ELMo, and GPT-2 embeddings," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 55–65. [Online]. Available: https://aclanthology.org/D19-1006

[99] F. Carlsson, A. C. Gyllensten, E. Gogoulou, E. Y. Hellqvist, and M. Sahlgren, "Semantic re-tuning with contrastive tension," in *International Conference on Learning Representations*, 2020.

[100] H. W. Chung, T. Fevry, H. Tsai, M. Johnson, and S. Ruder, "Rethinking embedding coupling in pre-trained language models," in *International Conference on Learning Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=xpFFI_NtgpW

[101] A. Rogers, O. Kovaleva, and A. Rumshisky, "A primer in BERTology: What we know about how BERT works," *Transactions of the Association for Computational Linguistics*, vol. 8, pp. 842–866, 2020. [Online]. Available: https://aclanthology.org/2020.tacl-1.54

[102] W. Timkey and M. van Schijndel, "All bark and no bite: Rogue dimensions in transformer language models obscure representational quality," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 4527–4546. [Online]. Available: https://aclanthology.org/2021.emnlp-main.372

[103] H. Sajjad, F. Alam, F. Dalvi, and N. Durrani, "Effect of post-processing on contextualized word representations," 2021. [Online]. Available: https://arxiv.org/abs/2104.07456

[104] H. Zhao, Z. Lu, and P. Poupart, "Self-adaptive hierarchical sentence model," in *Proceedings of the 24th International Conference on Artificial Intelligence*, ser. IJCAI'15. AAAI Press, 2015, p. 4069–4076.

[105] J. Yang and hai zhao, "Deepening hidden representations from pre-trained language models," 2021. [Online]. Available: https://openreview.net/forum?id=a9nIWs-Orh

[106] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, Jun. 2018, pp. 2227–2237. [Online]. Available: https://aclanthology.org/N18-1202

[107] J. Michael, J. A. Botha, and I. Tenney, "Asking without telling: Exploring latent ontologies in contextual representations," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics, Nov. 2020, pp. 6792–6812. [Online]. Available: https://aclanthology.org/2020.emnlp-main.552

[108] W. Shi, M. Chen, P. Zhou, and K.-W. Chang, "Retrofitting contextualized word embeddings with paraphrases," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 1198–1203. [Online]. Available: https://aclanthology.org/D19-1113

[109] M. Artetxe and H. Schwenk, "Massively Multilingual Sentence Embeddings for Zero-Shot Cross-Lingual Transfer and Beyond," *Transactions of the Association for Computational Linguistics*, vol. 7, pp. 597–610, 09 2019. [Online]. Available: https://doi.org/10.1162/tacl_a_00288

[110] S. Toshniwal, H. Shi, B. Shi, L. Gao, K. Livescu, and K. Gimpel, "A cross-task analysis of text span representations," in *Proceedings of the 5th Workshop on Representation Learning for NLP*. Online: Association for Computational Linguistics, Jul. 2020, pp. 166–176. [Online]. Available: https://aclanthology.org/2020.repl4nlp-1.20

[111] R. Socher, J. Pennington, E. H. Huang, A. Y. Ng, and C. D. Manning, "Semi-supervised recursive autoencoders for predicting sentiment distributions," in *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*. Edinburgh, Scotland, UK.: Association for Computational Linguistics, Jul. 2011, pp. 151–161. [Online]. Available: https://aclanthology.org/D11-1014

[112] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Ng, and C. Potts, "Recursive deep models for semantic compositionality over a sentiment treebank," in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Seattle, Washington, USA: Association for Computational Linguistics, Oct. 2013, pp. 1631–1642. [Online]. Available: https://aclanthology.org/D13-1170

[113] G. Salton, E. A. Fox, and H. Wu, "Extended boolean information retrieval," *Commun. ACM*, vol. 26, no. 11, p. 1022–1036, nov 1983. [Online]. Available: https://doi.org/10.1145/182.358466

[114] A. Rücklé, S. Eger, M. Peyrard, and I. Gurevych, "Concatenated power mean word embeddings as universal cross-lingual sentence representations," 2018. [Online]. Available: https://arxiv.org/abs/1803.01400

[115] Z. Lin, M. Feng, C. N. dos Santos, M. Yu, B. Xiang, B. Zhou, and Y. Bengio, "A STRUCTURED SELF-ATTENTIVE SENTENCE EMBEDDING," in *International Conference on Learning Representations*, 2017. [Online]. Available: https://openreview.net/forum?id=BJC_jUqxe

[116] S. Eger, A. Rücklé, and I. Gurevych, "Pitfalls in the evaluation of sentence embeddings," in *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*. Florence, Italy: Association for Computational Linguistics, Aug. 2019, pp. 55–60. [Online]. Available: https://aclanthology.org/W19-4308

[117] W. Rudman, N. Gillman, T. Rayne, and C. Eickhoff, "IsoScore: Measuring the uniformity of embedding space utilization," in *Findings of the Association for Computational Linguistics: ACL 2022*. Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 3325–3339. [Online]. Available: https://aclanthology.org/2022.findings-acl.262

[118] J. Mu and P. Viswanath, "All-but-the-top: Simple and effective postprocessing for word representations," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=HkuGJ3kCb

[119] T. Zhou, J. Sedoc, and J. Rodu, "Getting in shape: Word embedding subspaces," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*. International Joint Conferences on Artificial Intelligence Organization, 7 2019, pp. 5478–5484. [Online]. Available: https://doi.org/10.24963/ijcai.2019/761

[120] J. Gao, D. He, X. Tan, T. Qin, L. Wang, and T. Liu, "Representation degeneration problem in training natural language generation models," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=SkEYojRqtm

[121] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," 2019.

[122] T. Schnabel, I. Labutov, D. Mimno, and T. Joachims, "Evaluation methods for unsupervised word embeddings," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, Sep. 2015, pp. 298–307. [Online]. Available: https://aclanthology.org/D15-1036

[123] B. Li, H. Zhou, J. He, M. Wang, Y. Yang, and L. Li, "On the sentence embeddings from pre-trained language models," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics, Nov. 2020, pp. 9119–9130. [Online]. Available: https://aclanthology.org/2020.emnlp-main.733

[124] D. Biś, M. Podkorytov, and X. Liu, "Too much in common: Shifting of embeddings in transformer language models and its implications," in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Online: Association for Computational Linguistics, Jun. 2021, pp. 5117–5130. [Online]. Available: https://aclanthology.org/2021.naacl-main.403

[125] S. Rajaee and M. T. Pilehvar, "An isotropy analysis in the multilingual BERT embedding space," in *Findings of the Association for Computational Linguistics: ACL 2022*. Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 1309–1316. [Online]. Available: https://aclanthology.org/2022.findings-acl.103

[126] T. Schick and H. Schütze, "Rare words: A major problem for contextualized embeddings and how to fix it by attentive mimicking," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 05, Apr. 2020, pp. 8766–8774. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/6403

[127] G. A. Miller, "Wordnet: A lexical database for english," *Commun. ACM*, vol. 38, no. 11, p. 39–41, nov 1995. [Online]. Available: https://doi.org/10.1145/219717.219748

[128] Y.-Y. Lee, H. Ke, H.-H. Huang, and H.-H. Chen, "Less is more: Filtering abnormal dimensions in glove," in *Proceedings of the 25th International Conference Companion on World Wide Web*, ser. WWW '16 Companion. Republic and Canton of Geneva, CHE: International World Wide Web Conferences Steering Committee, 2016, p. 71–72. [Online]. Available: https://doi.org/10.1145/2872518.2889381

[129] O. Kovaleva, S. Kulshreshtha, A. Rogers, and A. Rumshisky, "BERT busters: Outlier dimensions that disrupt transformers," in *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*. Online: Association for Computational Linguistics, Aug. 2021, pp. 3392–3405. [Online]. Available: https://aclanthology.org/2021.findings-acl.300

[130] Z. Luo, A. Kulmizev, and X. Mao, "Positional artefacts propagate through masked language model embeddings," in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Online: Association for Computational Linguistics, Aug. 2021, pp. 5312–5327. [Online]. Available: https://aclanthology.org/2021.acl-long.413

[131] Y. Ding, K. Martinkus, D. Pascual, S. Clematide, and R. Wattenhofer, "On isotropy calibration of transformer models," in *Proceedings of the Third Workshop on Insights from Negative Results in NLP*. Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 1–9. [Online]. Available: https://aclanthology.org/2022.insights-1.1

[132] X. Cai, J. Huang, Y. Bian, and K. Church, "Isotropy in the contextual embedding space: Clusters and manifolds," in *International Conference on Learning Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=xYGNO86OWDH

[133] S. Rajaee and M. T. Pilehvar, "A cluster-based approach for improving isotropy in contextual embedding space," in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. Online: Association for Computational Linguistics, Aug. 2021, pp. 575–584. [Online]. Available: https://aclanthology.org/2021.acl-short.73

[134] ——, "How does fine-tuning affect the geometry of embedding space: A case study on isotropy," in *Findings of the Association for Computational Linguistics: EMNLP 2021*. Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 3042–3049. [Online]. Available: https://aclanthology.org/2021.findings-emnlp.261

[135] M. Artetxe, G. Labaka, I. Lopez-Gazpio, and E. Agirre, "Uncovering divergent linguistic information in word embeddings with lessons for intrinsic and extrinsic evaluation," in *Proceedings of the 22nd Conference on Computational Natural Language Learning*. Brussels, Belgium: Association for Computational Linguistics, Oct. 2018, pp. 282–291. [Online]. Available: https://aclanthology.org/K18-1028

[136] B. Wang, F. Chen, A. Wang, and C.-C. J. Kuo, "Post-processing of word representations via variance normalization and dynamic embedding," in *2019 IEEE International Conference on Multimedia and Expo (ICME)*, 2019, pp. 718–723.

[137] T. Liu, L. Ungar, and J. a. Sedoc, "Unsupervised post-processing of word vectors via conceptor negation," in *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence*, ser. AAAI'19/IAAI'19/EAAI'19. AAAI Press, 2019. [Online]. Available: https://doi.org/10.1609/aaai.v33i01.33016778

[138] H. Jaeger, "Controlling recurrent neural networks by conceptors," Jacobs University Bremen, Tech. Rep. No. 31, 2014. [Online]. Available: https://doi.org/10.48550/arXiv.1403.3369

[139] S. Karve, L. Ungar, and J. Sedoc, "Conceptor debiasing of word representations evaluated on WEAT," in *Proceedings of the First Workshop on Gender Bias in Natural Language Processing*. Florence, Italy: Association for Computational Linguistics, Aug. 2019, pp. 40–48. [Online]. Available: https://aclanthology.org/W19-3806

[140] Y. Liang, R. Cao, J. Zheng, J. Ren, and L. Gao, "Learning to remove: Towards isotropic pre-trained bert embedding," in *Artificial Neural Networks and Machine Learning – ICANN*

2021: 30th International Conference on Artificial Neural Networks, Bratislava, Slovakia, September 14–17, 2021, Proceedings, Part V. Berlin, Heidelberg: Springer-Verlag, 2021, p. 448–459. [Online]. Available: https://doi.org/10.1007/978-3-030-86383-8_36

[141] V. Raunak, V. Gupta, and F. Metze, "Effective dimensionality reduction for word embeddings," in *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*. Florence, Italy: Association for Computational Linguistics, Aug. 2019, pp. 235–243. [Online]. Available: https://aclanthology.org/W19-4328

[142] M. Artetxe, G. Labaka, and E. Agirre, "Generalizing and improving bilingual word embedding mappings with a multi-step framework of linear transformations," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, Apr. 2018. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/11992

[143] T. Gao, X. Yao, and D. Chen, "SimCSE: Simple contrastive learning of sentence embeddings," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 6894–6910. [Online]. Available: https://aclanthology.org/2021.emnlp-main.552

[144] T. Wang and P. Isola, "Understanding contrastive representation learning through alignment and uniformity on the hypersphere," in *Proceedings of the 37th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, H. D. III and A. Singh, Eds., vol. 119. PMLR, 13–18 Jul 2020, pp. 9929–9939. [Online]. Available: https://proceedings.mlr.press/v119/wang20k.html

[145] B. Wang, C.-C. J. Kuo, and H. Li, "Just rank: Rethinking evaluation with word and sentence similarities," in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 6060–6077. [Online]. Available: https://aclanthology.org/2022.acl-long.419

[146] M. Faruqui, J. Dodge, S. K. Jauhar, C. Dyer, E. Hovy, and N. A. Smith, "Retrofitting word vectors to semantic lexicons," in *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Denver, Colorado: Association for Computational Linguistics, May–Jun. 2015, pp. 1606–1615. [Online]. Available: https://aclanthology.org/N15-1184

[147] Z. Yu, T. Cohen, B. Wallace, E. Bernstam, and T. Johnson, "Retrofitting word vectors of MeSH terms to improve semantic similarity measures," in *Proceedings of the Seventh International Workshop on Health Text Mining and Information Analysis*. Auxtin, TX: Association for Computational Linguistics, Nov. 2016, pp. 43–51. [Online]. Available: https://aclanthology.org/W16-6106

[148] M. Zhang, Y. Fujinuma, M. J. Paul, and J. Boyd-Graber, "Why overfitting isn't always bad: Retrofitting cross-lingual word embeddings to dictionaries," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, Jul. 2020, pp. 2214–2220. [Online]. Available: https://aclanthology.org/2020.acl-main.201

[149] G. Glavaš and I. Vulić, "Explicit retrofitting of distributional word vectors," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia: Association for Computational Linguistics, Jul. 2018, pp. 34–45. [Online]. Available: https://aclanthology.org/P18-1004

[150] J. Zheng, Y. Wang, G. Wang, J. Xia, Y. Huang, G. Zhao, Y. Zhang, and S. Li, "Using context-to-vector with graph retrofitting to improve word embeddings," in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 8154–8163. [Online]. Available: https://aclanthology.org/2022.acl-long.561

[151] L. Wang, J. Huang, K. Huang, Z. Hu, G. Wang, and Q. Gu, "Improving neural language generation with spectrum control," in *International Conference on Learning Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=ByxY8CNtvr

[152] A. Tamkin, D. Jurafsky, and N. Goodman, "Language through a prism: A spectral approach for multiscale language representations," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 5492–5504. [Online]. Available: https://proceedings.neurips.cc/paper/2020/file/3acb2a202ae4bea8840224e6fce16fd0-Paper.pdf

[153] V. Zhelezniak, A. Savkov, and N. Hammerla, "Estimating mutual information between dense word embeddings," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, Jul. 2020, pp. 8361–8371. [Online]. Available: https://aclanthology.org/2020.acl-main.741

[154] A. Kraskov, H. Stögbauer, and P. Grassberger, "Estimating mutual information," *Phys. Rev. E*, vol. 69, p. 066138, Jun 2004. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevE.69.066138

[155] V. Zhelezniak, A. Savkov, A. Shen, and N. Hammerla, "Correlation coefficients and semantic textual similarity," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 951–962. [Online]. Available: https://aclanthology.org/N19-1100

[156] V. Zhelezniak, A. Savkov, A. Shen, F. Moramarco, J. Flann, and N. Y. Hammerla, "Don't settle for average, go for the max: Fuzzy sets and max-pooled word vectors," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=SkxXg2C5FX

[157] C. Min, Y. Chu, L. Yang, B. Xu, and H. Lin, "Locality preserving sentence encoding," in *Findings of the Association for Computational Linguistics: EMNLP 2021*. Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 3050–3060. [Online]. Available: https://aclanthology.org/2021.findings-emnlp.262

[158] S. Kayal, "Unsupervised sentence-embeddings by manifold approximation and projection," in *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*. Online: Association for Computational Linguistics, Apr. 2021, pp. 1–11. [Online]. Available: https://aclanthology.org/2021.eacl-main.1

[159] M. Kusner, Y. Sun, N. Kolkin, and K. Weinberger, "From word embeddings to document distances," in *Proceedings of the 32nd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, F. Bach and D. Blei, Eds., vol. 37. Lille, France: PMLR, 07–09 Jul 2015, pp. 957–966. [Online]. Available: https://proceedings.mlr.press/v37/kusnerb15.html

[160] L. Wu, I. E.-H. Yen, K. Xu, F. Xu, A. Balakrishnan, P.-Y. Chen, P. Ravikumar, and M. J. Witbrock, "Word mover's embedding: From Word2Vec to document embedding," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, Oct.-Nov. 2018, pp. 4524–4534. [Online]. Available: https://aclanthology.org/D18-1482

[161] Z. Wang, Y. Zhang, and H. Wu, "Structural-aware sentence similarity with recursive optimal transport," 2020. [Online]. Available: https://arxiv.org/abs/2002.00745

[162] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *Proceedings of the 31st International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, E. P. Xing and T. Jebara, Eds., vol. 32, no. 2. Bejing, China: PMLR, 22–24 Jun 2014, pp. 1188–1196. [Online]. Available: https://proceedings.mlr.press/v32/le14.html

[163] J. H. Lau and T. Baldwin, "An empirical evaluation of doc2vec with practical insights into document embedding generation," in *Proceedings of the 1st Workshop on Representation Learning for NLP*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 78–86. [Online]. Available: https://aclanthology.org/W16-1609

[164] F. Hill, K. Cho, and A. Korhonen, "Learning distributed representations of sentences from unlabelled data," in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. San Diego, California: Association for Computational Linguistics, Jun. 2016, pp. 1367–1377. [Online]. Available: https://aclanthology.org/N16-1162

[165] M. Chen, "Efficient vector representation for documents through corruption," in *International Conference on Learning Representations*, 2017. [Online]. Available: https://openreview.net/forum?id=B1Igu2ogg

[166] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in Neural Information Processing Systems*, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, Eds., vol. 27. Curran Associates, Inc., 2014. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2014/file/a14ac55a4f27472c5d894ec1c3c743d2-Paper.pdf

[167] M. Schuster and K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[168] R. Kiros, Y. Zhu, R. R. Salakhutdinov, R. Zemel, R. Urtasun, A. Torralba, and S. Fidler, "Skip-thought vectors," in *Advances in Neural Information Processing Systems*, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, Eds., vol. 28. Curran Associates, Inc., 2015. [Online]. Available: https://proceedings.neurips.cc/paper/2015/file/f442d33fa06832082290ad8544a8da27-Paper.pdf

[169] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder–decoder for statistical machine translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1724–1734. [Online]. Available: https://aclanthology.org/D14-1179

[170] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 11 1997. [Online]. Available: https://doi.org/10.1162/neco.1997.9.8.1735

[171] S. R. Bowman, G. Angeli, C. Potts, and C. D. Manning, "A large annotated corpus for learning natural language inference," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, Sep. 2015, pp. 632–642. [Online]. Available: https://aclanthology.org/D15-1075

[172] L. Logeswaran and H. Lee, "An efficient framework for learning sentence representations," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=rJvJXZb0W

[173] S. Subramanian, A. Trischler, Y. Bengio, and C. J. Pal, "Learning general purpose distributed sentence representations via large scale multi-task learning," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=B18WgG-CZ

[174] A. Nie, E. Bennett, and N. Goodman, "DisSent: Learning sentence representations from explicit discourse relations," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 4497–4510. [Online]. Available: https://aclanthology.org/P19-1442

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

[176] M. Henderson, R. Al-Rfou, B. Strope, Y. hsuan Sung, L. Lukacs, R. Guo, S. Kumar, B. Miklos, and R. Kurzweil, "Efficient natural language response suggestion for smart reply," 2017.

[177] L. Cao, E. Larsson, V. von Ehrenheim, D. D. Cavalcanti Rocha, A. Martin, and S. Horn, "PAUSE: Positive and annealed unlabeled sentence embedding," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 10 096–10 107. [Online]. Available: https://aclanthology.org/2021.emnlp-main.791

[178] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 2, 2006, pp. 1735–1742.

[179] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014. [Online]. Available: http://jmlr.org/papers/v15/srivastava14a.html

[180] F. Liu, I. Vulić, A. Korhonen, and N. Collier, "Fast, effective, and self-supervised: Transforming masked language models into universal lexical and sentence encoders," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 1442–1459. [Online]. Available: https://aclanthology.org/2021.emnlp-main.109

[181] X. Wu, C. Gao, L. Zang, J. Han, Z. Wang, and S. Hu, "Esimcse: Enhanced sample building method for contrastive learning of unsupervised sentence embedding," 2021. [Online]. Available: https://arxiv.org/abs/2109.04380

[182] T. Klein and M. Nabi, "Scd: Self-contrastive decorrelation for sentence embeddings," 2022.

[183] D. Shen, M. Zheng, Y. Shen, Y. Qu, and W. Chen, "A simple but tough-to-beat data augmentation approach for natural language understanding and generation," 2020. [Online]. Available: https://arxiv.org/abs/2009.13818

[184] A. Rozsa, E. M. Rudd, and T. E. Boult, "Adversarial diversity and hard positive generation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2016, pp. 25–32.

[185] R. Cao, Y. Wang, Y. Liang, L. Gao, J. Zheng, J. Ren, and Z. Wang, "Exploring the impact of negative samples of contrastive learning: A case study of sentence embedding," 2022. [Online]. Available: https://arxiv.org/abs/2202.13093

[186] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: http://arxiv.org/abs/1412.6572

[187] J. Wei and K. Zou, "EDA: Easy data augmentation techniques for boosting performance on text classification tasks," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 6382–6388. [Online]. Available: https://aclanthology.org/D19-1670

[188] P. Su, Y. Peng, and K. Vijay-Shanker, "Improving BERT model using contrastive learning for biomedical relation extraction," in *Proceedings of the 20th Workshop on Biomedical Language Processing*. Online: Association for Computational Linguistics, Jun. 2021, pp. 1–10. [Online]. Available: https://aclanthology.org/2021.bionlp-1.1

[189] H. Fang, S. Wang, M. Zhou, J. Ding, and P. Xie, "Cert: Contrastive self-supervised learning for language understanding," 2020. [Online]. Available: https://arxiv.org/abs/2005.12766

[190] S. Edunov, M. Ott, M. Auli, and D. Grangier, "Understanding back-translation at scale," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, Oct.-Nov. 2018, pp. 489–500. [Online]. Available: https://aclanthology.org/D18-1045

[191] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. Bowman, "GLUE: A multi-task benchmark and analysis platform for natural language understanding," in *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*. Brussels, Belgium: Association for Computational Linguistics, Nov. 2018, pp. 353–355. [Online]. Available: https://aclanthology.org/W18-5446

[192] Z. Wu, S. Wang, J. Gu, M. Khabsa, F. Sun, and H. Ma, "Clear: Contrastive learning for sentence representation," 2020.

[193] J. Giorgi, O. Nitski, B. Wang, and G. Bader, "DeCLUTR: Deep contrastive learning for unsupervised textual representations," in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Online: Association for Computational Linguistics, Aug. 2021, pp. 879–895. [Online]. Available: https://aclanthology.org/2021.acl-long.72

[194] A. Conneau and D. Kiela, "SentEval: An evaluation toolkit for universal sentence representations," in *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*. Miyazaki, Japan: European Language Resources Association (ELRA), May 2018. [Online]. Available: https://aclanthology.org/L18-1269

[195] Q. Wu, C. Tao, T. Shen, C. Xu, X. Geng, and D. Jiang, "Pcl: Peer-contrastive learning with diverse augmentations for unsupervised sentence embeddings," 2022. [Online]. Available: https://arxiv.org/abs/2201.12093

[196] X. Wu, C. Gao, J. Wang, L. Zang, Z. Wang, and S. Hu, "Disco: Effective knowledge distillation for contrastive learning of sentence embeddings," 2021. [Online]. Available: https://arxiv.org/abs/2112.05638

[197] A. Williams, N. Nangia, and S. Bowman, "A broad-coverage challenge corpus for sentence understanding through inference," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, Jun. 2018, pp. 1112–1122. [Online]. Available: https://aclanthology.org/N18-1101

[198] D. Zhang, S.-W. Li, W. Xiao, H. Zhu, R. Nallapati, A. O. Arnold, and B. Xiang, "Pairwise supervised contrastive learning of sentence representations," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 5786–5798. [Online]. Available: https://aclanthology.org/2021.emnlp-main.467

[199] S. Li, X. Hu, L. Lin, and L. Wen, "Pair-level supervised contrastive learning for natural language inference," 2022. [Online]. Available: https://arxiv.org/abs/2201.10927

[200] J. Ni, G. H. Ábrego, N. Constant, J. Ma, K. B. Hall, D. Cer, and Y. Yang, "Sentence-t5: Scalable sentence encoders from pre-trained text-to-text models," 2021. [Online]. Available: https://arxiv.org/abs/2108.08877

[201] S. Merity, C. Xiong, J. Bradbury, and R. Socher, "Pointer sentinel mixture models," in *International Conference on Learning Representations*, 2017. [Online]. Available: https://openreview.net/forum?id=Byj72udxe

[202] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. Le Scao, S. Gugger, M. Drame, Q. Lhoest, and A. Rush, "Transformers: State-of-the-art natural language processing," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Online: Association for Computational Linguistics, Oct. 2020, pp. 38–45. [Online]. Available: https://aclanthology.org/2020.emnlp-demos.6

[203] Y. Zhu, R. Kiros, R. Zemel, R. Salakhutdinov, R. Urtasun, A. Torralba, and S. Fidler, "Aligning books and movies: Towards story-like visual explanations by watching movies and reading books," in *2015 IEEE International Conference on Computer Vision (ICCV)*. Los Alamitos, CA, USA: IEEE Computer Society, dec 2015, pp. 19–27. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/ICCV.2015.11

[204] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, Łukasz Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. Corrado, M. Hughes, and J. Dean, "Google's neural machine translation system: Bridging the gap between human and machine translation," 2016.

[205] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," 2016.

[206] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014. [Online]. Available: http://jmlr.org/papers/v15/srivastava14a.html

[207] W. L. Taylor, ""cloze procedure": A new tool for measuring readability," *Journalism Quarterly*, vol. 30, no. 4, pp. 415–433, 1953. [Online]. Available: https://doi.org/10.1177/107769905303000401

[208] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[209] B. Bolstad, R. Irizarry, M. Åstrand, and T. Speed, "A comparison of normalization methods for high density oligonucleotide array data based on variance and bias," *Bioinformatics*, vol. 19, no. 2, pp. 185–193, 01 2003. [Online]. Available: https://doi.org/10.1093/bioinformatics/19.2.185

[210] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[211] X. Zhang and Y. LeCun, "Text understanding from scratch," 2015. [Online]. Available: https://arxiv.org/abs/1502.01710

[212] M. R. H. Rakib, N. Zeh, M. Jankowska, and E. Milios, "Enhancement of short text clustering by iterative classification," in *Natural Language Processing and Information Systems*, E. Métais, F. Meziane, H. Horacek, and P. Cimiano, Eds. Cham: Springer International Publishing, 2020, pp. 105–117.

[213] J. Xu, B. Xu, P. Wang, S. Zheng, G. Tian, J. Zhao, and B. Xu, "Self-taught convolutional neural networks for short text clustering," *Neural Networks*, vol. 88, pp. 22–31, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0893608016301976

[214] J. Yin and J. Wang, "A model-based approach for text clustering with outlier detection," in *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*, 2016, pp. 625–636.

[215] X.-H. Phan, L.-M. Nguyen, and S. Horiguchi, "Learning to classify short and sparse text & web with hidden topics from large-scale data collections," in *Proceedings of the 17th International Conference on World Wide Web*, ser. WWW '08. New York, NY, USA: Association for Computing Machinery, 2008, p. 91–100. [Online]. Available: https://doi.org/10.1145/1367497.1367510

[216] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and Édouard Duchesnay, "Scikit-learn: Machine learning in python," *Journal of Machine Learning Research*, vol. 12, no. 85, pp. 2825–2830, 2011. [Online]. Available: http://jmlr.org/papers/v12/pedregosa11a.html

[217] J. Munkres, "Algorithms for the assignment and transportation problems," *Journal of the Society for Industrial and Applied Mathematics*, vol. 5, no. 1, pp. 32–38, 1957. [Online]. Available: http://www.jstor.org/stable/2098689

[218] E. Agirre, D. Cer, M. Diab, and A. Gonzalez-Agirre, "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)*. Montréal, Canada: Association for Computational Linguistics, 7-8 Jun. 2012, pp. 385–393. [Online]. Available: https://aclanthology.org/S12-1051

[219] E. Agirre, D. Cer, M. Diab, A. Gonzalez-Agirre, and W. Guo, "*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*. Atlanta, Georgia, USA: Association for Computational Linguistics, Jun. 2013, pp. 32–43. [Online]. Available: https://aclanthology.org/S13-1004

[220] E. Agirre, C. Banea, C. Cardie, D. Cer, M. Diab, A. Gonzalez-Agirre, W. Guo, R. Mihalcea, G. Rigau, and J. Wiebe, "SemEval-2014 task 10: Multilingual semantic textual similarity," in *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*. Dublin, Ireland: Association for Computational Linguistics, Aug. 2014, pp. 81–91. [Online]. Available: https://aclanthology.org/S14-2010

[221] E. Agirre, C. Banea, C. Cardie, D. Cer, M. Diab, A. Gonzalez-Agirre, W. Guo, I. Lopez-Gazpio, M. Maritxalar, R. Mihalcea, G. Rigau, L. Uria, and J. Wiebe, "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). Denver, Colorado: Association for Computational Linguistics, Jun. 2015, pp. 252–263. [Online]. Available: https://aclanthology.org/S15-2045

[222] E. Agirre, C. Banea, D. Cer, M. Diab, A. Gonzalez-Agirre, R. Mihalcea, G. Rigau, and J. Wiebe, "SemEval-2016 task 1: Semantic textual similarity, monolingual and cross-lingual evaluation," in *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*. San Diego, California: Association for Computational Linguistics, Jun. 2016, pp. 497–511. [Online]. Available: https://aclanthology.org/S16-1081

[223] D. Cer, M. Diab, E. Agirre, I. Lopez-Gazpio, and L. Specia, "SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation," in *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. Vancouver, Canada: Association for Computational Linguistics, Aug. 2017, pp. 1–14. [Online]. Available: https://aclanthology.org/S17-2001

[224] M. Marelli, S. Menini, M. Baroni, L. Bentivogli, R. Bernardi, and R. Zamparelli, "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)*. Reykjavik, Iceland: European Language Resources Association (ELRA), May 2014, pp. 216–223. [Online]. Available: http://www.lrec-conf.org/proceedings/lrec2014/pdf/363_Paper.pdf

[225] M. Abdalla, K. Vishnubhotla, and S. M. Mohammad, "What makes sentences semantically related: A textual relatedness dataset and empirical study," in *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*. Dubrovnik, Croatia: Association for Computational Linguistics, 2023.

[226] B. Pang and L. Lee, "Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales," in *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*. Ann Arbor, Michigan: Association for Computational Linguistics, Jun. 2005, pp. 115–124. [Online]. Available: https://aclanthology.org/P05-1015

[227] M. Hu and B. Liu, "Mining and summarizing customer reviews," in *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '04. New York, NY, USA: Association for Computing Machinery, 2004, p. 168–177. [Online]. Available: https://doi.org/10.1145/1014052.1014073

[228] B. Pang and L. Lee, "Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales," in *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ser. ACL '05. USA: Association for Computational Linguistics, 2005, p. 115–124. [Online]. Available: https://doi.org/10.3115/1219840.1219855

[229] J. Wiebe, T. Wilson, and C. Cardie, "Annotating expressions of opinions and emotions in language," *Language resources and evaluation*, vol. 39, pp. 165–210, 2005. [Online]. Available: https://doi.org/10.1007/s10579-005-7880-9

[230] L. Deng and J. Wiebe, "MPQA 3.0: An entity/event-level sentiment corpus," in *Proceedings of the 2015 conference of the North American chapter of the association for computational linguistics: human language technologies*, 2015, pp. 1323–1328. [Online]. Available: https://aclanthology.org/N15-1146.pdf

[231] B. Dolan, C. Quirk, and C. Brockett, "Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources," in *COLING 2004: Proceedings of the 20th International Conference on Computational Linguistics*. Geneva, Switzerland: COLING, aug 23–aug 27 2004, pp. 350–356. [Online]. Available: https://aclanthology.org/C04-1051

[232] A. Cohan, W. Ammar, M. van Zuylen, and F. Cady, "Structural scaffolds for citation intent classification in scientific publications," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 3586–3596. [Online]. Available: https://aclanthology.org/N19-1361

[233] X. Li and D. Roth, "Learning question classifiers," in *COLING 2002: The 19th International Conference on Computational Linguistics*, 2002. [Online]. Available: https://aclanthology.org/C02-1150

[234] E. Voita, R. Sennrich, and I. Titov, "The bottom-up evolution of representations in the transformer: A study with machine translation and language modeling objectives," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 4396–4406. [Online]. Available: https://aclanthology.org/D19-1448

[235] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, S. Riedel, and D. Kiela, "Retrieval-augmented generation for knowledge-intensive nlp tasks," in *Proceedings of the 34th International Conference on Neural Information Processing Systems*, ser. NIPS '20. Red Hook, NY, USA: Curran Associates Inc., 2020.

A Detailed Performance on Individual Tasks

A.1 Token Aggregation and Post-Processing

Table 6: Spearman correlation dependence on token aggregation and post-processing techniques for semantic textual similarity STS12 task. The best result for each model is bolded, while underlined result is the best across all the models.

Model	**T0**	**T4**	**BERT**	**BERT + Avg.**	**Avg.**	**B2S**	**B2S-100**	**RE**
Layer	**Last**		**First + Last**			–		
avg.	48.7	53.9	45.1	48.7	48.5	47.2	54.2	34.7
+ normalize	48.7	53.9	45.1	48.7	48.5	47.2	54.2	34.7
+ quantile-u	52.1	56.2	48.7	51.9	51.5	45.8	52.4	40.1
+ quantile-u^W	50.2	55.1	46.6	50.7	50.3	44.6	52.5	36.9
+ whiten	41.9	42.9	43.2	44.5	46.0	45.0	44.6	44.1
+ whiten^W	46.2	50.4	47.3	52.8	57.7	54.7	55.0	54.2
+ zscore	51.5	54.1	50.2	53.4	53.1	49.1	54.7	43.6
idf$_t^W$	56.2	56.8	57.5	58.7	58.3	57.6	**56.9**	55.4
+ normalize	56.2	56.8	57.5	58.7	58.3	**57.7**	56.9	55.4
+ quantile-u	56.2	56.5	56.4	57.9	57.5	54.5	53.6	52.0
+ quantile-u^W	57.0	57.5	57.1	58.7	58.4	55.4	54.4	52.6
+ whiten	43.7	43.8	46.1	46.7	47.2	45.3	44.7	44.6
+ whiten^W	52.6	53.3	57.5	58.8	58.7	57.5	56.8	55.5
+ zscore	56.1	55.9	57.4	58.7	58.2	56.9	56.4	55.6
idf$_t^T$	**59.2**	64.4	57.3	59.0	59.0	47.2	54.2	55.1
+ normalize	59.2	**64.4**	57.3	59.0	59.0	47.2	54.2	55.1
+ quantile-u	58.5	62.6	56.5	58.2	58.1	45.8	52.4	52.3
+ quantile-u^W	59.1	63.9	57.0	58.8	58.9	44.6	52.5	52.9
+ whiten	43.8	45.3	46.0	46.7	47.4	45.0	44.6	44.6
+ whiten^W	52.6	56.3	56.9	58.2	58.4	54.7	55.0	55.1
+ zscore	56.9	59.7	57.7	59.3	59.3	49.1	54.7	55.6
-biases	55.1	58.2	58.6	**62.2**	**62.9**	45.1	51.9	57.5
+ normalize	55.1	58.2	**58.6**	62.2	62.8	45.1	51.9	57.5
+ quantile-u	56.0	59.6	57.9	60.9	61.2	44.0	50.3	54.6
+ quantile-u^W	54.7	59.1	58.2	61.5	61.6	42.7	50.3	55.1
+ whiten^W	51.8	54.0	57.2	60.2	61.6	53.2	53.3	58.1
+ whiten	42.5	43.2	45.1	46.2	47.6	44.2	44.2	46.2
+ zscore	55.1	57.5	57.8	60.4	60.6	47.4	52.7	**58.2**
[MASK]	55.2	60.6						
+ normalize	55.2	60.6						
+ quantile-u	56.0	60.8						
+ quantile-u^W	56.6	61.7						
+ whiten	47.7	48.0						
+ whiten^W	52.5	57.7						
+ zscore	55.4	59.8						

Table 7: Spearman correlation dependence on token aggregation and post-processing techniques for semantic textual similarity STS13 task. The best result for each model is bolded, while underlined result is the best across all the models.

Model	**T0**	**T4**	**BERT**	**BERT + Avg.**	**Avg.**	**B2S**	**B2S-100**	**RE**
Layer	**Last**		**First + Last**			–		
avg.	62.6	64.7	64.3	63.4	56.6	61.4	66.2	48.8
+ normalize	62.6	64.7	64.3	63.4	56.6	61.4	66.2	48.8
+ quantile-u	68.4	70.9	68.3	69.0	65.5	60.4	68.9	54.8
+ quantile-u^W	64.0	66.5	65.7	66.8	64.0	59.2	67.2	53.3
+ whiten	76.0	76.6	77.6	78.3	78.0	76.4	76.1	**75.1**
+ whiten^W	62.2	66.5	64.8	65.8	66.2	60.6	60.8	63.7
+ zscore	70.7	72.5	70.4	70.7	67.1	62.8	68.9	55.9
idf$_t^W$	75.4	74.7	77.2	79.1	77.8	**77.5**	77.3	72.5
+ normalize	75.4	74.7	77.2	79.1	77.8	77.5	77.3	72.5
+ quantile-u	76.7	76.0	77.7	79.3	78.7	77.1	76.3	73.4
+ quantile-u^W	75.6	74.7	77.1	79.3	79.1	77.2	76.7	74.2
+ whiten	76.8	76.0	**78.3**	78.6	77.6	77.1	76.7	74.6
+ whiten^W	68.1	68.1	74.1	75.2	75.4	73.1	73.9	72.7
+ zscore	76.7	76.4	78.2	**79.7**	**79.3**	77.5	**78.0**	73.0
idf$_t^T$	70.8	73.7	74.8	76.5	76.0	61.4	66.2	68.3
+ normalize	70.8	73.7	74.8	76.5	76.0	61.4	66.2	68.3
+ quantile-u	74.7	77.0	76.0	77.2	76.7	60.4	68.9	71.9
+ quantile-u^W	72.0	74.6	74.8	76.4	76.0	59.2	67.2	71.5
+ whiten	**77.1**	**78.0**	78.1	78.3	77.3	76.4	76.1	74.0
+ whiten^W	67.2	71.0	70.3	70.4	70.1	60.6	60.8	68.1
+ zscore	75.9	77.7	76.2	76.9	75.8	62.8	68.9	69.8
-biases	66.5	68.7	68.4	69.9	68.1	56.3	60.5	61.3
+ normalize	66.5	68.7	68.4	69.9	68.1	56.3	60.5	61.3
+ quantile-u	70.9	73.5	70.8	71.9	70.3	55.0	63.2	66.2
+ quantile-u^W	67.4	70.3	68.6	69.9	68.2	53.9	61.3	64.4
+ whiten^W	63.1	67.1	65.2	65.7	65.9	56.6	56.7	63.0
+ whiten	76.2	76.8	77.9	78.5	78.1	73.3	73.1	74.7
+ zscore	73.0	74.9	72.5	72.9	70.6	58.5	64.0	64.6
`[MASK]`	63.4	76.2						
+ normalize	63.4	76.2						
+ quantile-u	68.1	77.3						
+ quantile-u^W	65.2	76.0						
+ whiten	73.1	77.6						
+ whiten^W	61.7	70.9						
+ zscore	69.4	76.9						

Table 8: Spearman correlation dependence on token aggregation and post-processing techniques for semantic textual similarity STS14 task. The best result for each model is bolded, while underlined result is the best across all the models.

Model	**T0**	**T4**	**BERT**	**BERT + Avg.**	**Avg.**	**B2S**	**B2S-100**	**RE**
Layer	**Last**		**First + Last**			–		
avg.	53.9	54.5	54.6	55.4	51.7	55.9	61.2	48.2
+ normalize	53.9	54.5	54.6	55.4	51.7	55.9	61.2	48.2
+ quantile-u	57.8	60.1	58.1	60.0	58.2	53.5	62.9	52.3
+ quantile-u^W	55.6	57.1	57.0	59.4	58.1	53.5	62.2	52.1
+ whiten	64.0	65.0	66.2	68.1	69.4	67.7	68.5	68.3
+ whiten^W	53.5	57.0	57.6	61.1	64.5	60.7	61.3	64.6
+ zscore	58.6	61.3	58.7	61.2	59.6	55.7	61.3	53.5
idf$_t^W$	65.5	65.9	65.2	67.7	67.8	67.2	67.8	67.6
+ normalize	65.5	65.9	65.2	67.7	67.8	67.2	67.8	67.6
+ quantile-u	**66.7**	66.7	66.5	69.0	69.3	67.2	68.1	66.4
+ quantile-u^W	66.5	66.6	66.2	68.9	69.3	67.1	68.0	67.0
+ whiten	65.4	65.4	67.6	68.5	68.8	**68.3**	**68.7**	67.9
+ whiten^W	59.2	59.4	64.7	67.0	68.2	66.6	67.4	68.0
+ zscore	65.7	65.3	66.2	68.8	69.2	66.8	68.0	67.8
idf$_t^T$	61.6	63.9	64.3	66.7	67.1	55.9	61.2	65.5
+ normalize	61.6	63.9	64.3	66.7	67.1	55.9	61.2	65.5
+ quantile-u	64.2	66.5	65.8	67.8	68.0	53.5	62.9	65.3
+ quantile-u^W	62.9	65.4	65.3	67.5	67.8	53.5	62.2	65.7
+ whiten	65.2	**66.8**	67.5	68.3	68.5	67.7	68.5	67.4
+ whiten^W	58.0	61.6	63.0	64.6	65.3	60.7	61.3	65.3
+ zscore	64.5	66.8	65.1	67.1	67.2	55.7	61.3	65.7
-biases	60.7	62.0	63.8	66.9	66.9	53.1	58.1	64.7
+ normalize	60.7	62.0	63.8	66.9	66.9	53.1	58.1	64.7
+ quantile-u	63.4	65.9	65.2	67.9	67.8	50.8	59.8	65.2
+ quantile-u^W	61.9	64.3	64.8	67.7	67.6	50.9	59.2	65.3
+ whiten^W	57.3	60.4	62.8	65.5	67.3	58.5	58.9	66.2
+ whiten	64.9	66.2	**68.3**	**69.6**	**70.4**	65.8	66.4	**68.8**
+ zscore	64.1	66.3	64.8	67.4	67.2	53.3	58.6	65.3
`[MASK]`	53.9	63.9						
+ normalize	53.9	63.9						
+ quantile-u	56.1	65.1						
+ quantile-u^W	54.8	64.7						
+ whiten	60.8	66.3						
+ whiten^W	51.5	61.3						
+ zscore	56.7	65.1						

Table 9: Spearman correlation dependence on token aggregation and post-processing techniques for semantic textual similarity STS15 task. The best result for each model is bolded, while underlined result is the best across all the models.

Model	**T0**	**T4**	**BERT**	**BERT + Avg.**	**Avg.**	**B2S**	**B2S-100**	**RE**
Layer	**Last**		**First + Last**			–		
avg.	70.2	70.6	70.5	70.0	64.2	69.8	74.2	62.1
+ normalize	70.2	70.6	70.5	70.0	64.2	69.8	74.2	62.1
+ quantile-u	71.8	73.7	72.7	74.3	72.4	68.9	73.6	61.4
+ quantile-u^W	71.5	72.5	72.3	74.1	72.6	69.4	74.4	62.7
+ whiten	62.6	62.9	64.0	65.9	69.3	69.3	68.8	67.9
+ whiten^W	70.2	71.8	71.9	74.9	77.0	74.7	75.2	74.7
+ zscore	71.9	73.0	72.6	74.3	72.4	69.3	73.9	64.3
idf$_t^W$	75.4	74.4	75.8	75.4	71.5	73.7	74.4	74.4
+ normalize	75.4	74.4	75.8	75.4	71.5	73.7	74.4	74.4
+ quantile-u	75.1	74.2	75.3	75.7	73.7	73.4	71.9	68.5
+ quantile-u^W	**76.5**	75.6	76.3	76.9	74.8	74.6	73.7	71.1
+ whiten	62.7	61.4	64.3	65.2	66.3	66.8	66.7	65.0
+ whiten^W	72.2	71.6	75.0	76.5	76.6	**75.4**	**75.5**	74.4
+ zscore	74.2	73.1	75.2	76.2	75.0	73.8	74.4	73.2
idf$_t^T$	74.1	75.2	76.2	76.8	75.1	69.8	74.2	73.8
+ normalize	74.1	75.2	76.2	76.8	75.1	69.8	74.2	73.8
+ quantile-u	75.2	**77.1**	76.0	76.8	75.7	68.9	73.6	69.5
+ quantile-u^W	75.4	77.0	**76.9**	77.8	76.7	69.4	74.4	71.6
+ whiten	63.1	64.2	64.5	66.0	68.5	69.3	68.8	67.2
+ whiten^W	72.9	74.6	75.4	76.2	75.8	74.7	75.2	73.1
+ zscore	75.0	76.0	75.9	76.8	75.8	69.3	73.9	72.7
-biases	72.9	73.3	76.2	77.3	75.2	67.3	70.8	74.3
+ normalize	72.9	73.3	76.2	77.3	75.2	67.3	70.8	74.3
+ quantile-u	74.1	76.1	76.3	77.5	76.2	66.3	70.1	69.6
+ quantile-u^W	73.7	75.3	76.8	**78.4**	**77.3**	66.9	70.9	71.7
+ whiten^W	72.6	73.7	75.4	77.0	77.3	72.6	72.6	**74.8**
+ whiten	64.2	64.8	65.2	66.5	69.1	67.4	66.4	66.8
+ zscore	74.4	75.1	75.7	76.9	75.7	67.0	70.7	73.6
`[MASK]`	67.0	74.1						
+ normalize	67.0	74.1						
+ quantile-u	68.0	74.7						
+ quantile-u^W	68.3	75.7						
+ whiten	61.0	65.1						
+ whiten^W	67.7	74.5						
+ zscore	68.0	72.9						

Table 10: Spearman correlation dependence on token aggregation and post-processing techniques for semantic textual similarity STS16 task. The best result for each model is bolded, while underlined result is the best across all the models.

Model	**T0**	**T4**	**BERT**	**BERT + Avg.**	**Avg.**	**B2S**	**B2S-100**	**RE**
Layer	**Last**			**First + Last**			–	
avg.	69.1	69.7	67.9	65.2	56.4	55.2	58.5	55.5
+ normalize	69.1	69.7	67.9	65.2	56.4	55.1	58.5	55.5
+ quantile-u	70.1	72.4	70.3	71.0	67.0	55.9	59.5	54.8
+ quantile-u^W	69.7	71.5	69.8	70.0	65.3	54.6	57.9	55.1
+ whiten	65.4	65.9	67.1	68.9	69.9	65.0	63.0	67.1
+ whiten^W	64.7	67.2	69.9	71.5	71.1	63.1	62.8	68.3
+ zscore	69.3	71.0	72.0	73.2	69.8	60.2	62.3	60.4
idf$_t^W$	70.1	69.1	73.9	73.0	69.5	68.8	68.0	71.9
+ normalize	70.1	69.1	73.9	73.0	69.5	68.8	68.0	71.9
+ quantile-u	70.5	69.9	73.7	74.2	72.9	68.1	65.8	68.9
+ quantile-u^W	70.5	69.8	74.0	74.4	72.7	67.9	66.2	69.6
+ whiten	64.6	64.0	68.0	68.9	69.1	65.6	63.4	67.3
+ whiten^W	70.0	69.1	75.6	76.0	**74.8**	69.6	68.7	**72.4**
+ zscore	72.2	71.7	**75.6**	$\underline{\textbf{76.4}}$	74.8	**70.3**	**69.5**	72.2
idf$_t^T$	72.9	73.1	72.8	72.0	69.7	55.2	58.5	69.1
+ normalize	72.9	73.1	72.8	72.0	69.7	55.2	58.5	69.1
+ quantile-u	**73.9**	**75.0**	72.8	73.0	71.8	55.9	59.5	67.3
+ quantile-u^W	73.7	74.6	72.7	72.7	71.2	54.6	57.9	67.5
+ whiten	67.0	68.6	68.0	68.7	69.5	65.0	63.0	67.8
+ whiten^W	69.0	71.0	73.2	72.8	71.3	63.1	62.8	69.3
+ zscore	73.1	73.1	74.4	74.4	73.1	60.2	62.3	70.1
-biases	70.1	71.9	72.2	72.1	68.4	52.8	56.3	65.4
+ normalize	70.1	71.9	72.2	72.1	68.4	52.8	56.3	65.4
+ quantile-u	71.1	73.7	72.3	73.3	71.6	53.5	57.5	64.7
+ quantile-u^W	70.6	73.2	72.1	72.7	70.4	52.3	55.8	64.4
+ whiten^W	65.2	68.0	71.0	72.1	71.5	61.1	60.9	68.5
+ whiten	64.9	66.2	67.3	68.5	69.3	63.4	61.4	67.4
+ zscore	71.1	72.5	73.3	74.1	72.0	58.2	60.4	66.7
`[MASK]`	67.2	71.0						
+ normalize	67.2	71.0						
+ quantile-u	68.2	71.8						
+ quantile-u^W	67.7	71.7						
+ whiten	65.5	70.0						
+ whiten^W	64.1	69.4						
+ zscore	66.6	68.9						

Table 11: Spearman correlation dependence on token aggregation and post-processing techniques for semantic textual similarity STS-B task. The best result for each model is bolded, while underlined result is the best across all the models.

Model	**T0**	**T4**	**BERT**	**BERT + Avg.**	**Avg.**	**B2S**	**B2S-100**	**RE**
Layer	**Last**		**First + Last**			–		
avg.	60.4	61.0	59.0	59.9	54.2	56.1	60.7	46.5
+ normalize	60.4	61.0	59.0	59.9	54.2	56.1	60.7	46.5
+ quantile-u	67.7	70.9	63.6	65.2	61.7	56.9	61.1	52.4
+ quantile-u^W	62.8	65.0	61.3	63.5	59.7	54.7	60.4	50.5
+ whiten	68.2	70.0	68.7	70.0	69.6	66.7	64.6	68.1
+ whiten^W	62.1	65.8	62.6	67.3	69.4	65.1	64.1	67.5
+ zscore	68.7	71.1	65.0	66.7	63.0	58.8	62.5	54.6
idf$_t^W$	69.0	68.2	70.3	69.6	66.4	68.0	66.9	69.8
+ normalize	69.0	68.2	70.3	69.6	66.4	68.0	66.9	69.8
+ quantile-u	69.4	68.6	69.3	69.6	67.8	66.7	64.7	64.4
+ quantile-u^W	69.4	68.8	70.2	70.3	68.1	67.0	65.3	65.7
+ whiten	65.6	65.0	67.9	68.0	67.2	65.8	64.1	66.5
+ whiten^W	67.0	66.6	**71.2**	**72.1**	**71.5**	**69.8**	**68.2**	**70.4**
+ zscore	70.7	69.9	71.0	71.7	70.1	69.0	67.8	70.0
idf$_t^T$	68.8	70.7	69.6	69.3	67.3	56.1	60.7	67.0
+ normalize	68.8	70.7	69.6	69.3	67.3	56.1	60.7	67.0
+ quantile-u	**72.4**	**75.1**	69.2	69.3	67.7	56.9	61.1	64.2
+ quantile-u^W	70.4	73.2	69.7	69.6	67.7	54.7	60.4	65.0
+ whiten	69.6	72.2	68.4	68.4	67.7	66.7	64.6	67.0
+ whiten^W	67.8	71.0	69.6	69.9	68.7	65.1	64.1	67.1
+ zscore	72.4	74.7	70.1	70.3	68.6	58.8	62.5	67.4
-biases	64.7	65.4	70.2	71.0	67.8	52.9	57.3	66.6
+ normalize	64.7	65.4	70.2	71.0	67.8	52.9	57.3	66.6
+ quantile-u	70.1	73.4	69.9	71.0	68.9	54.2	58.1	62.8
+ quantile-u^W	66.1	69.2	70.3	71.5	69.0	51.7	57.1	63.9
+ whiten^W	66.1	68.5	69.6	71.3	70.9	61.8	60.8	68.8
+ whiten	68.1	69.9	69.5	69.9	69.3	64.6	62.4	66.5
+ zscore	71.1	73.2	70.2	71.1	68.9	56.1	59.4	67.4
`[MASK]`	68.5	73.3						
+ normalize	68.5	73.3						
+ quantile-u	70.8	74.8						
+ quantile-u^W	69.3	74.4						
+ whiten	70.5	74.3						
+ whiten^W	65.3	70.9						
+ zscore	70.5	73.7						

Table 12: Spearman correlation dependence on token aggregation and post-processing techniques for semantic textual similarity SICK-R task. The best result for each model is bolded, while underlined result is the best across all the models.

Model	**T0**	**T4**	**BERT**	**BERT + Avg.**	**Avg.**	**B2S**	**B2S-100**	**RE**
Layer	**Last**		**First + Last**			–		
avg.	64.4	64.2	63.8	63.1	60.3	60.2	61.2	53.1
+ normalize	64.4	64.2	63.8	63.1	60.3	60.2	61.2	53.1
+ quantile-u	66.1	66.3	64.9	64.7	62.5	61.0	61.8	54.8
+ quantile-u^W	64.9	64.7	64.3	64.1	61.6	59.7	61.0	53.1
+ whiten	61.0	60.8	60.4	59.1	55.1	55.3	55.4	53.3
+ whiten^W	63.1	63.2	63.7	63.9	61.5	61.0	59.9	58.4
+ zscore	66.3	66.7	65.0	64.9	62.8	62.1	63.1	56.3
idf$_t^W$	62.5	62.0	61.7	59.7	57.5	61.9	60.1	57.4
+ normalize	62.5	62.0	61.7	59.7	57.5	61.9	60.2	57.4
+ quantile-u	63.0	62.6	61.8	60.4	58.5	60.8	59.4	52.9
+ quantile-u^W	62.9	62.5	61.8	60.5	59.0	61.1	59.5	54.7
+ whiten	57.7	57.5	57.1	55.9	53.8	54.5	54.7	52.0
+ whiten^W	60.9	61.1	61.9	60.8	59.0	61.2	60.1	56.7
+ zscore	63.7	63.4	63.2	62.3	60.9	**62.9**	61.3	57.3
idf$_t^T$	64.4	65.0	62.8	61.1	59.2	60.2	61.3	56.8
+ normalize	64.4	65.0	62.8	61.1	59.2	60.2	61.2	56.8
+ quantile-u	66.0	66.3	63.2	61.6	59.7	61.0	61.8	54.3
+ quantile-u^W	65.0	65.6	63.0	61.3	59.5	59.7	61.0	55.2
+ whiten	59.9	60.3	58.1	56.8	54.4	55.3	55.4	52.5
+ whiten^W	63.5	64.2	62.4	61.0	58.9	61.0	59.9	56.3
+ zscore	66.2	66.5	63.7	62.4	60.8	62.1	**63.1**	57.0
-biases	65.1	65.9	65.6	65.1	63.8	59.0	59.7	59.7
+ normalize	65.1	65.9	65.6	65.1	63.8	59.0	59.7	59.7
+ quantile-u	66.9	67.7	66.0	65.7	64.7	59.9	60.2	56.5
+ quantile-u^W	65.6	66.5	65.8	65.5	64.4	58.6	59.3	57.3
+ whiten^W	64.8	65.4	64.7	63.9	61.3	59.8	58.6	58.0
+ whiten	61.3	61.1	60.3	58.9	55.4	54.2	54.1	53.1
+ zscore	67.1	**67.8**	**66.3**	**66.3**	**65.6**	61.1	62.0	**60.4**
`[MASK]`	64.8	65.0						
+ normalize	64.8	65.0						
+ quantile-u	67.1	66.0						
+ quantile-u^W	65.6	65.6						
+ whiten	61.7	61.8						
+ whiten^W	63.3	64.2						
+ zscore	**67.1**	66.2						

Table 13: Spearman correlation dependence on token aggregation and post-processing techniques for semantic textual similarity STR task. The best result for each model is bolded, while underlined result is the best across all the models.

Model	**T0**	**T4**	**BERT**	**BERT + Avg.**	**Avg.**	**B2S**	**B2S-100**	**RE**
Layer	**Last**		**First + Last**			–		
avg.	60.9	59.8	65.4	65.6	59.4	48.2	48.9	53.3
+ normalize	60.9	59.8	65.4	65.6	59.4	48.2	48.9	53.3
+ quantile-u	65.7	67.8	68.1	69.2	65.4	47.3	47.9	53.6
+ quantile-u^W	63.6	64.1	67.3	68.8	65.3	48.0	48.6	53.7
+ whiten	68.9	69.6	70.1	69.7	67.0	51.4	48.8	62.1
+ whiten^W	65.2	67.7	66.7	68.7	68.6	51.3	49.2	63.2
+ zscore	66.6	68.9	66.7	67.5	64.5	50.1	50.1	56.0
idf$_t^W$	68.2	67.8	71.9	70.9	66.3	53.8	51.5	62.6
+ normalize	68.2	67.8	71.9	70.9	66.3	53.8	51.5	62.6
+ quantile-u	69.7	69.5	**72.5**	72.7	70.0	52.4	49.7	59.7
+ quantile-u^W	69.2	69.0	72.5	**72.8**	**70.4**	53.3	50.8	61.2
+ whiten	69.5	68.8	70.1	68.9	66.2	51.9	49.5	61.1
+ whiten^W	68.2	67.8	70.8	70.6	67.7	52.4	49.9	61.7
+ zscore	69.5	68.9	71.4	71.2	68.7	**54.0**	**52.2**	62.4
idf$_t^T$	67.1	67.5	71.8	70.8	66.9	48.2	48.9	**63.3**
+ normalize	67.1	67.5	71.8	70.8	66.9	48.2	48.9	63.3
+ quantile-u	**71.0**	73.2	72.1	71.9	69.3	47.3	47.9	59.3
+ quantile-u^W	69.8	71.6	72.0	72.0	69.7	48.0	48.6	60.6
+ whiten	70.4	71.1	70.4	69.0	66.3	51.4	48.8	61.4
+ whiten^W	69.0	71.3	70.4	70.0	67.6	51.3	49.2	62.1
+ zscore	70.8	73.2	71.4	71.3	69.3	50.1	50.1	63.1
-biases	62.8	62.5	70.7	70.1	64.9	46.0	45.8	60.3
+ normalize	62.8	62.5	70.7	70.1	64.9	46.0	45.8	60.3
+ quantile-u	66.7	69.6	72.2	72.1	68.6	45.7	45.0	58.0
+ quantile-u^W	65.2	67.1	71.9	71.9	68.4	46.2	45.5	58.9
+ whiten^W	67.8	69.3	69.2	69.2	66.4	49.4	47.0	60.1
+ whiten	69.1	69.5	69.6	68.5	65.0	49.6	46.9	59.6
+ zscore	68.3	70.8	69.7	69.1	65.9	48.5	47.6	60.0
`[MASK]`	67.0	70.1						
+ normalize	67.0	70.1						
+ quantile-u	70.7	**73.5**						
+ quantile-u^W	69.1	73.2						
+ whiten	70.5	72.3						
+ whiten^W	68.1	72.6						
+ zscore	70.7	73.1						

Table 14: Clustering accuracy dependence on token aggregation and post-processing techniques for the agnews dataset. The best result for each model is bolded, while underlined result is the best across all the models.

Model	**T0**	**T4**	**BERT**	**BERT + Avg.**	**Avg.**	**B2S**	**B2S-100**	**RE**
Layer	**Last**		**First + Last**			–		
avg.	85.4	80.7	81.2	84.4	79.0	85.0	85.0	28.0
+ normalize	85.5	80.6	85.6	86.1	79.6	**85.7**	85.9	27.2
+ quantile-u	85.2	86.9	84.0	85.1	79.7	85.2	85.2	27.5
+ quantile-u^W	85.4	86.7	85.3	85.7	80.6	85.4	85.4	27.2
+ whiten	34.3	31.3	31.3	31.5	30.5	29.9	29.9	28.8
+ whiten^W	75.2	75.0	72.7	72.8	58.6	70.6	69.0	28.4
+ zscore	86.0	81.2	81.3	83.8	79.8	84.8	85.2	27.7
idf$_t^W$	79.5	81.4	80.6	84.0	78.0	83.6	83.7	39.9
+ normalize	84.9	85.6	84.7	85.1	78.3	85.0	84.9	41.4
+ quantile-u	85.8	85.7	81.7	84.4	78.6	84.2	84.2	41.2
+ quantile-u^W	85.6	85.8	81.8	84.7	79.1	84.6	84.7	**43.4**
+ whiten	30.9	32.0	31.3	31.8	30.7	30.7	30.1	27.6
+ whiten^W	74.5	78.1	73.8	74.4	70.9	72.5	72.1	38.4
+ zscore	85.9	81.8	80.4	84.0	78.9	83.7	83.7	41.0
idf$_t^T$	80.1	82.1	80.9	82.5	79.3	85.0	85.0	28.7
+ normalize	85.2	82.2	85.1	84.8	81.7	85.7	**85.9**	28.5
+ quantile-u	85.9	87.0	82.4	81.9	80.9	85.2	85.2	28.6
+ quantile-u^W	85.9	87.0	82.0	84.4	82.1	85.3	85.5	28.4
+ whiten	30.2	32.6	30.9	31.3	29.9	29.9	29.8	28.0
+ whiten^W	72.1	68.4	73.0	71.9	45.5	70.6	69.0	28.9
+ zscore	85.9	82.6	81.1	82.1	79.7	84.8	85.2	28.8
-biases	85.7	86.4	81.4	85.6	83.4	84.0	83.8	28.5
+ normalize	85.7	86.6	**86.8**	**86.8**	**83.8**	85.3	85.5	28.4
+ quantile-u	85.4	86.7	85.7	85.4	83.4	84.7	84.7	28.5
+ quantile-u^W	85.6	87.1	86.1	86.0	83.6	84.6	84.9	29.4
+ whiten^W	73.5	73.4	72.4	72.1	55.5	68.8	67.6	28.2
+ whiten	31.1	31.5	33.0	30.4	30.3	29.5	29.8	28.3
+ zscore	**86.0**	**87.1**	81.7	84.8	83.5	84.4	84.5	28.1
`[MASK]`	74.8	81.9						
+ normalize	75.1	82.2						
+ quantile-u	63.1	82.1						
+ quantile-u^W	76.1	84.2						
+ whiten	30.9	31.8						
+ whiten^W	52.9	48.3						
+ zscore	76.1	81.7						

Table 15: Clustering accuracy dependence on token aggregation and post-processing techniques for the biomedical dataset. The best result for each model is bolded, while underlined result is the best across all the models.

Model	**T0**	**T4**	**BERT**	**BERT + Avg.**	**Avg.**	**B2S**	**B2S-100**	**RE**
Layer	**Last**		**First + Last**			–		
avg.	32.7	32.4	34.8	36.1	30.7	33.1	37.0	19.4
+ normalize	32.8	32.0	35.1	36.3	30.3	33.8	**37.6**	20.7
+ quantile-u	33.6	32.5	37.0	39.1	33.1	32.9	36.7	21.0
+ quantile-u^W	33.4	32.5	35.4	37.9	34.5	33.0	37.0	21.1
+ whiten	29.4	30.2	27.2	23.2	18.2	12.7	13.2	15.9
+ whiten^W	34.8	34.7	40.2	**42.7**	36.5	35.0	34.4	29.4
+ zscore	33.5	33.0	36.1	37.4	32.1	32.4	36.1	20.0
idf$_t^W$	31.0	30.0	32.3	32.3	28.1	34.5	34.5	32.1
+ normalize	31.2	30.2	32.5	32.6	28.7	34.7	34.8	**35.2**
+ quantile-u	32.5	31.3	33.6	33.6	29.8	33.9	35.0	32.4
+ quantile-u^W	31.7	30.7	31.9	33.6	30.7	33.7	34.4	33.4
+ whiten	27.3	27.6	26.4	20.6	17.1	11.8	12.4	16.7
+ whiten^W	32.8	32.1	37.0	38.7	35.4	**37.2**	37.2	30.8
+ zscore	32.0	30.8	32.6	32.5	28.2	34.3	34.3	31.8
idf$_t^T$	33.4	33.6	36.0	36.2	32.6	33.1	37.0	30.0
+ normalize	33.6	34.1	36.0	37.1	33.9	33.8	37.6	32.2
+ quantile-u	35.4	35.3	36.9	37.2	34.1	32.9	36.7	31.3
+ quantile-u^W	35.0	35.7	36.6	37.5	34.7	33.0	37.0	32.2
+ whiten	28.3	30.8	20.9	19.4	16.3	12.7	13.2	15.2
+ whiten^W	36.8	37.4	40.4	39.9	33.1	35.0	34.4	29.2
+ zscore	34.7	35.4	36.9	36.8	32.4	32.4	36.1	29.5
-biases	35.1	32.2	39.0	39.2	37.2	31.6	35.9	29.2
+ normalize	35.2	32.5	39.4	39.4	**37.7**	32.9	36.9	30.8
+ quantile-u	36.0	33.0	39.8	39.6	37.2	32.3	36.2	29.9
+ quantile-u^W	36.6	34.6	39.6	39.9	37.3	32.4	36.3	30.9
+ whiten^W	**38.6**	**38.7**	**41.7**	40.3	34.4	34.4	33.4	28.7
+ whiten	29.8	31.6	22.0	17.1	13.5	12.5	12.0	13.2
+ zscore	35.6	32.7	39.4	39.5	37.0	30.9	35.6	28.7
`[MASK]`	24.0	34.8						
+ normalize	25.1	35.3						
+ quantile-u	27.4	35.9						
+ quantile-u^W	25.1	34.7						
+ whiten	21.9	28.4						
+ whiten^W	23.7	32.8						
+ zscore	26.4	35.0						

Table 16: Clustering accuracy dependence on token aggregation and post-processing techniques for the googleTS dataset. The best result for each model is bolded, while underlined result is the best across all the models.

Model	**T0**	**T4**	**BERT**	**BERT + Avg.**	**Avg.**	**B2S**	**B2S-100**	**RE**
Layer	**Last**		**First + Last**			–		
avg.	63.5	64.3	66.1	66.1	65.1	66.2	65.8	61.2
+ normalize	62.7	64.6	66.2	67.1	66.2	**68.3**	**67.7**	67.1
+ quantile-u	64.8	65.1	67.1	67.4	67.3	66.3	66.9	63.5
+ quantile-u^W	63.9	64.7	66.4	68.0	66.7	66.9	67.2	63.6
+ whiten	59.2	60.3	57.5	57.3	54.5	58.1	58.4	53.1
+ whiten^W	62.4	62.0	65.6	66.1	65.2	65.8	65.8	61.7
+ zscore	63.6	65.0	66.7	67.1	65.9	66.0	66.6	61.8
idf$_t^W$	63.5	62.6	65.2	66.2	65.6	64.5	65.8	63.8
+ normalize	64.3	63.1	66.0	66.5	66.0	66.5	66.7	**69.5**
+ quantile-u	64.0	65.4	66.6	66.5	67.0	66.4	66.1	64.1
+ quantile-u^W	64.4	64.9	67.5	67.8	66.6	67.4	65.9	63.4
+ whiten	57.9	59.9	58.3	56.5	55.2	58.9	58.6	51.4
+ whiten^W	61.9	61.4	66.2	67.7	66.7	65.5	66.7	63.3
+ zscore	63.5	63.3	66.1	66.4	65.7	65.9	65.6	63.9
idf$_t^T$	63.2	64.9	66.4	67.4	65.9	66.2	66.0	61.3
+ normalize	64.7	64.7	67.7	**68.6**	65.9	68.3	67.7	67.4
+ quantile-u	**65.2**	65.5	**68.5**	67.9	67.5	66.6	67.0	63.7
+ quantile-u^W	64.8	**66.2**	67.7	67.3	67.3	67.1	67.2	62.9
+ whiten	58.8	59.2	58.7	59.0	55.6	57.7	58.5	55.1
+ whiten^W	60.1	60.4	65.1	66.7	64.8	65.8	65.8	61.2
+ zscore	63.8	65.0	66.4	67.6	66.3	66.0	66.3	62.1
-biases	63.2	64.1	66.0	66.3	65.7	65.7	65.3	62.0
+ normalize	62.6	63.4	66.3	66.8	65.8	67.0	66.8	67.1
+ quantile-u	64.8	65.1	66.7	68.0	**67.7**	65.7	65.4	64.0
+ quantile-u^W	64.4	65.5	67.1	67.4	66.7	66.5	66.7	64.1
+ whiten^W	60.9	61.6	66.0	66.4	65.2	63.9	65.5	63.2
+ whiten	61.3	60.3	59.2	59.2	57.6	57.3	57.1	56.9
+ zscore	63.8	64.9	65.7	66.7	66.1	65.3	65.0	62.7
`[MASK]`	45.5	56.0						
+ normalize	45.9	56.5						
+ quantile-u	47.8	56.2						
+ quantile-u^W	46.9	56.4						
+ whiten	53.4	53.1						
+ whiten^W	43.0	53.3						
+ zscore	47.0	56.0						

Table 17: Clustering accuracy dependence on token aggregation and post-processing techniques for the searchsnippets dataset. The best result for each model is bolded, while underlined result is the best across all the models.

Model	**T0**	**T4**	**BERT**	**BERT + Avg.**	**Avg.**	**B2S**	**B2S-100**	**RE**
Layer	**Last**		**First + Last**			–		
avg.	67.3	73.3	72.2	70.8	63.6	72.0	72.0	23.3
+ normalize	68.2	73.8	81.7	73.1	65.7	76.4	76.3	20.4
+ quantile-u	69.8	72.4	74.1	73.0	66.8	75.2	72.9	22.3
+ quantile-u^W	69.7	73.5	81.6	74.7	68.6	75.7	75.1	21.7
+ whiten	33.8	36.1	32.3	31.7	26.3	22.9	24.8	21.2
+ whiten^W	47.7	49.2	58.3	61.8	56.3	58.8	58.3	24.0
+ zscore	67.9	73.2	72.7	71.3	63.7	72.3	72.1	22.8
idf$_t^W$	67.1	67.0	71.9	70.4	63.0	69.9	69.9	29.2
+ normalize	68.4	69.4	81.8	73.0	65.2	**81.4**	**81.4**	**36.6**
+ quantile-u	70.3	71.6	73.8	72.5	65.9	74.4	72.2	30.7
+ quantile-u^W	70.8	72.5	80.6	74.0	67.1	74.8	80.7	32.1
+ whiten	32.7	34.8	33.0	31.7	23.7	27.3	24.1	20.5
+ whiten^W	47.8	51.9	64.5	66.1	61.9	67.6	68.5	30.0
+ zscore	69.8	70.0	72.2	70.4	62.7	70.1	70.2	29.7
idf$_t^T$	55.1	75.3	72.0	70.5	59.3	72.0	72.0	26.1
+ normalize	63.2	75.6	81.2	76.9	60.9	76.4	76.3	24.8
+ quantile-u	68.8	78.5	74.6	76.1	59.1	75.2	72.9	26.1
+ quantile-u^W	**71.2**	78.7	81.3	**77.0**	61.0	75.7	75.1	25.8
+ whiten	30.6	35.7	30.8	29.8	25.0	22.9	24.8	21.3
+ whiten^W	46.8	46.1	62.7	63.0	52.9	58.8	58.3	26.7
+ zscore	68.6	79.1	73.3	72.4	59.0	72.3	72.1	25.8
-biases	66.5	70.4	82.6	73.0	71.7	70.5	71.9	22.8
+ normalize	67.0	70.5	**82.9**	74.8	73.8	75.2	80.9	22.6
+ quantile-u	68.8	80.3	80.7	73.8	72.3	73.5	73.8	23.8
+ quantile-u^W	69.8	**80.6**	82.6	76.5	**74.2**	75.6	72.7	22.9
+ whiten^W	48.4	46.0	64.5	64.4	54.9	57.1	53.6	23.9
+ whiten	31.6	36.3	33.2	29.3	22.7	23.3	23.9	22.0
+ zscore	67.6	79.8	82.1	72.6	71.4	72.6	72.6	23.3
`[MASK]`	61.1	69.9						
+ normalize	61.0	69.6						
+ quantile-u	60.3	69.6						
+ quantile-u^W	61.4	69.8						
+ whiten	31.1	34.0						
+ whiten^W	39.4	45.3						
+ zscore	61.8	69.8						

Table 18: Clustering accuracy dependence on token aggregation and post-processing techniques for the stackoverflow dataset. The best result for each model is bolded, while underlined result is the best across all the models.

Model	**T0**	**T4**	**BERT**	**BERT + Avg.**	**Avg.**	**B2S**	**B2S-100**	**RE**
Layer	**Last**		**First + Last**			–		
avg.	25.0	30.9	35.9	43.5	41.9	14.8	20.2	39.2
+ normalize	25.4	30.7	36.9	42.8	42.3	15.0	20.7	39.0
+ quantile-u	27.8	32.7	38.3	45.9	45.1	14.7	20.5	38.6
+ quantile-u^W	25.7	32.4	38.4	46.9	47.9	15.0	20.4	40.0
+ whiten	39.8	39.7	34.2	17.8	13.0	9.8	12.6	12.2
+ whiten^W	28.9	33.9	49.4	57.3	62.0	**21.7**	**23.5**	56.6
+ zscore	26.7	33.1	37.3	44.2	43.6	14.0	19.8	39.2
idf$_t^W$	28.6	29.8	30.6	37.7	39.1	16.1	17.5	55.4
+ normalize	29.0	30.3	31.1	37.2	38.0	16.8	18.6	62.7
+ quantile-u	29.8	31.1	32.0	38.1	40.7	16.8	17.5	57.7
+ quantile-u^W	30.0	30.9	32.1	39.9	40.8	16.3	18.0	58.8
+ whiten	30.6	30.1	26.1	15.4	12.2	9.7	9.7	12.5
+ whiten^W	29.8	31.4	40.0	50.3	55.8	16.2	17.1	56.8
+ zscore	30.2	30.6	31.1	37.4	38.7	16.1	16.8	54.1
idf$_t^T$	36.2	39.0	54.2	63.4	60.5	14.8	20.2	61.6
+ normalize	36.9	38.6	55.1	**66.6**	62.4	15.0	20.7	**70.6**
+ quantile-u	38.7	40.7	57.1	64.3	**63.1**	14.7	20.5	63.7
+ quantile-u^W	38.6	40.0	57.7	65.6	63.0	15.0	20.4	65.2
+ whiten	**43.5**	**44.6**	32.1	19.7	16.6	9.8	12.6	16.7
+ whiten^W	35.7	36.7	**59.9**	63.1	58.7	21.7	23.5	61.6
+ zscore	38.3	40.6	54.5	62.0	58.9	14.0	19.8	60.7
-biases	28.5	34.5	44.8	56.7	59.7	14.3	18.2	53.9
+ normalize	28.4	35.3	45.2	57.0	58.7	14.6	19.4	63.6
+ quantile-u	31.0	36.8	44.5	56.7	59.3	14.0	19.0	58.0
+ quantile-u^W	30.8	35.5	46.4	59.9	60.6	14.8	19.3	57.4
+ whiten^W	35.1	36.1	55.4	62.8	61.0	21.0	20.9	52.2
+ whiten	42.6	38.7	21.4	15.7	11.3	8.0	11.6	11.3
+ zscore	31.5	36.9	45.2	57.5	57.9	13.4	18.5	53.0
[MASK]	26.4	35.5						
+ normalize	26.5	36.1						
+ quantile-u	26.0	37.7						
+ quantile-u^W	24.9	37.0						
+ whiten	22.7	41.0						
+ whiten^W	18.0	32.2						
+ zscore	27.3	37.4						

Table 19: Clustering accuracy dependence on token aggregation and post-processing techniques for the tweet dataset. The best result for each model is bolded, while underlined result is the best across all the models.

Model	**T0**	**T4**	**BERT**	**BERT + Avg.**	**Avg.**	**B2S**	**B2S-100**	**RE**
Layer	**Last**		**First + Last**			–		
avg.	47.0	48.2	51.8	54.1	50.7	51.7	51.7	46.5
+ normalize	46.2	48.3	53.0	53.2	52.4	**53.7**	**55.2**	56.4
+ quantile-u	46.6	48.5	51.9	53.9	51.7	50.8	52.9	48.2
+ quantile-u^W	46.9	48.0	53.6	53.4	53.0	51.4	51.1	47.8
+ whiten	15.9	16.3	15.9	17.0	18.5	18.1	17.2	17.6
+ whiten^W	44.9	44.1	51.1	51.2	51.7	50.5	49.9	49.0
+ zscore	47.5	49.0	53.0	53.3	51.4	50.0	50.5	46.4
idf$_t^W$	48.6	47.2	50.5	52.1	48.5	49.4	50.8	45.4
+ normalize	48.3	47.8	51.9	52.7	48.5	50.3	50.4	51.5
+ quantile-u	49.0	47.9	52.7	51.9	48.8	49.8	49.9	46.4
+ quantile-u^W	48.3	47.9	52.7	52.8	48.3	49.4	49.6	46.0
+ whiten	15.3	15.1	15.7	16.2	18.3	17.4	17.0	15.9
+ whiten^W	43.0	42.3	50.0	50.8	50.1	49.8	49.3	45.5
+ zscore	49.3	46.7	52.2	51.4	48.0	48.9	49.1	46.3
idf$_t^T$	**50.8**	50.5	55.0	55.2	**54.5**	51.7	51.7	49.0
+ normalize	49.6	50.6	**55.1**	54.7	53.6	53.7	55.2	**58.5**
+ quantile-u	49.7	51.0	53.5	54.3	53.7	50.8	52.9	50.5
+ quantile-u^W	50.0	**52.3**	53.4	54.8	53.3	51.4	51.1	51.7
+ whiten	17.6	18.0	15.5	16.5	18.2	18.1	17.2	17.6
+ whiten^W	44.4	46.6	51.9	53.2	54.0	50.5	49.9	48.8
+ zscore	49.7	50.0	53.5	55.4	53.8	50.0	50.5	47.8
-biases	47.1	49.6	52.4	52.3	51.7	49.6	49.6	48.0
+ normalize	47.3	49.3	54.0	**55.7**	53.4	51.3	51.3	55.1
+ quantile-u	47.9	50.4	53.0	55.6	53.4	48.6	50.7	48.2
+ quantile-u^W	48.8	50.0	53.4	54.4	53.5	50.0	49.0	47.4
+ whiten^W	45.9	46.0	52.2	53.0	53.3	47.3	47.5	47.4
+ whiten	16.3	16.9	16.4	17.4	18.7	18.7	17.7	18.5
+ zscore	47.3	49.4	52.0	53.5	53.2	48.3	49.0	47.4
`[MASK]`	41.1	47.0						
+ normalize	41.0	47.1						
+ quantile-u	42.0	47.7						
+ quantile-u^W	41.9	47.4						
+ whiten	14.9	16.6						
+ whiten^W	38.4	41.7						
+ zscore	41.6	46.7						

A.2 BERT + Avg. Model in Different Layers

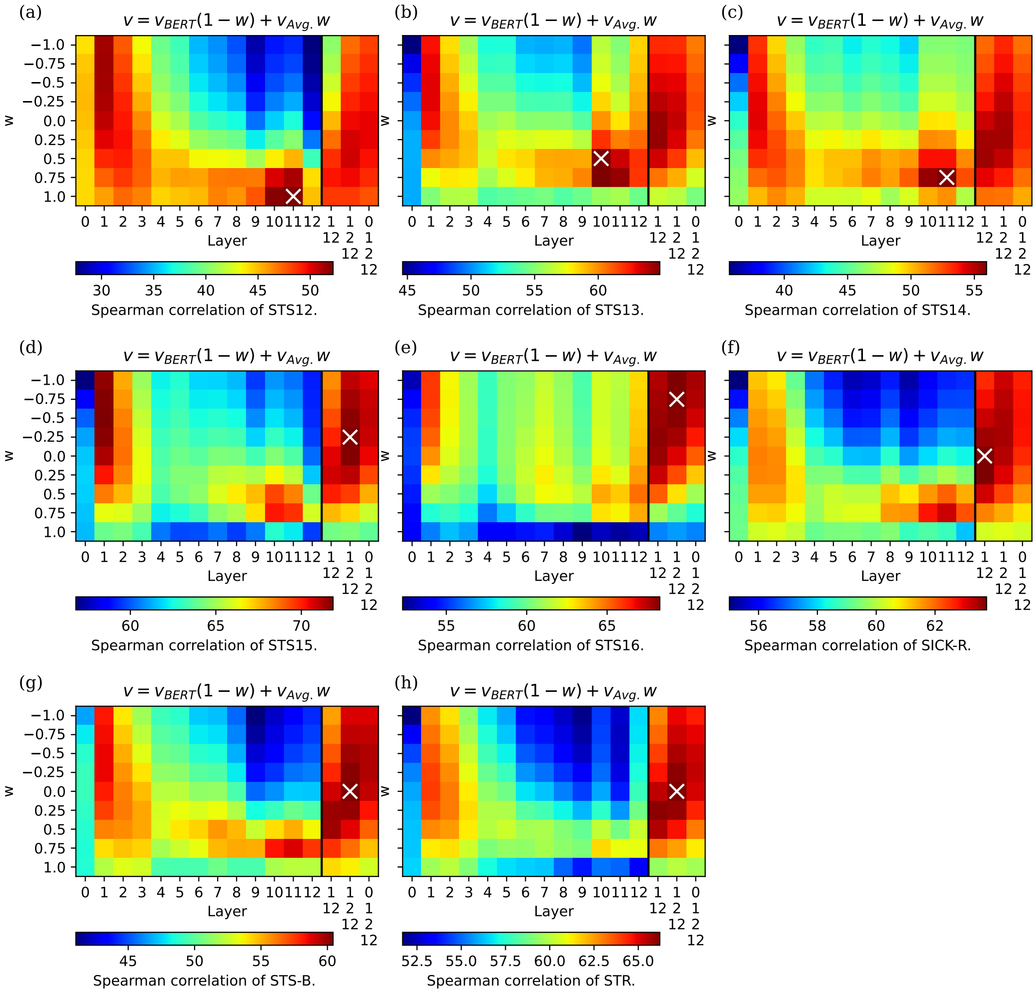

Figure 6: BERT + Avg. model individual task STS performance dependence on the weight w of Avg. model and layer, from which (for both models) representations are used. To the right of the black line on the horizontal axis, average aggregation of multiple layers is also shown. Tokens are simply averaged and no post-processing is used. The horizontal line with $w = 0.0$ corresponds to a regular Bert (B) model, $w = 0.5$ is B + Avg., and $w = 1.0$ is the Avg. model. The white × marks the maximum value.

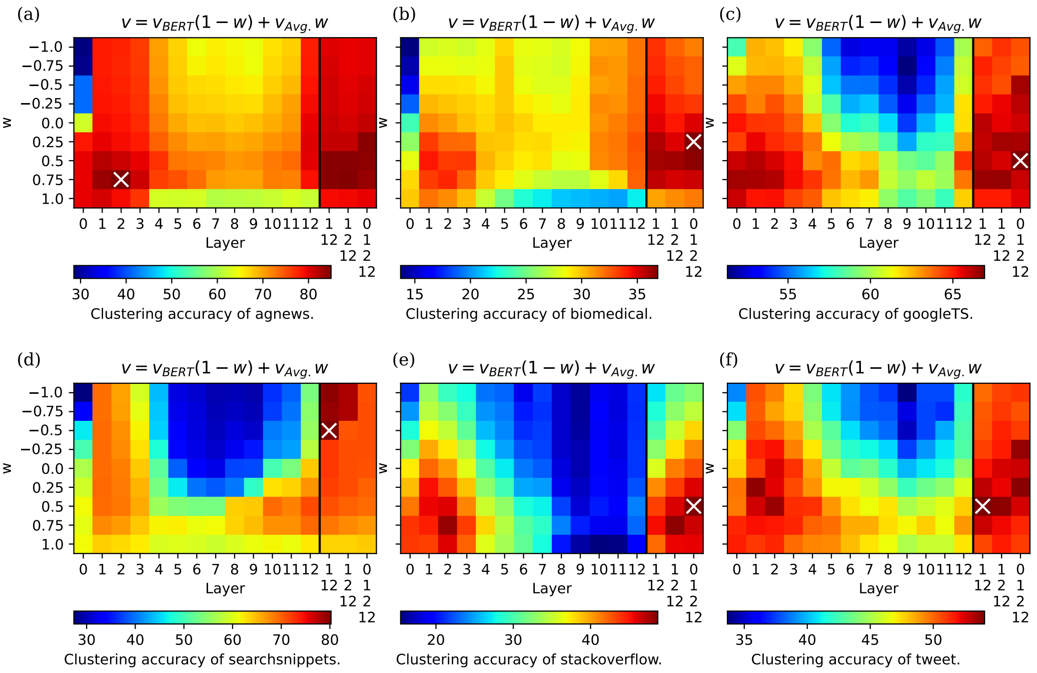

Figure 7: BERT + Avg. model individual task clustering performance dependence on the weight w of Avg. model and layer, from which (for both models) representations are used. To the right of the black line on the horizontal axis, average aggregation of multiple layers is also shown. Tokens are simply averaged and no post-processing is used. The horizontal line with $w = 0.0$ corresponds to a regular Bert (B) model, $w = 0.5$ is B + Avg., and $w = 1.0$ is the Avg. model. The white × marks the maximum value.

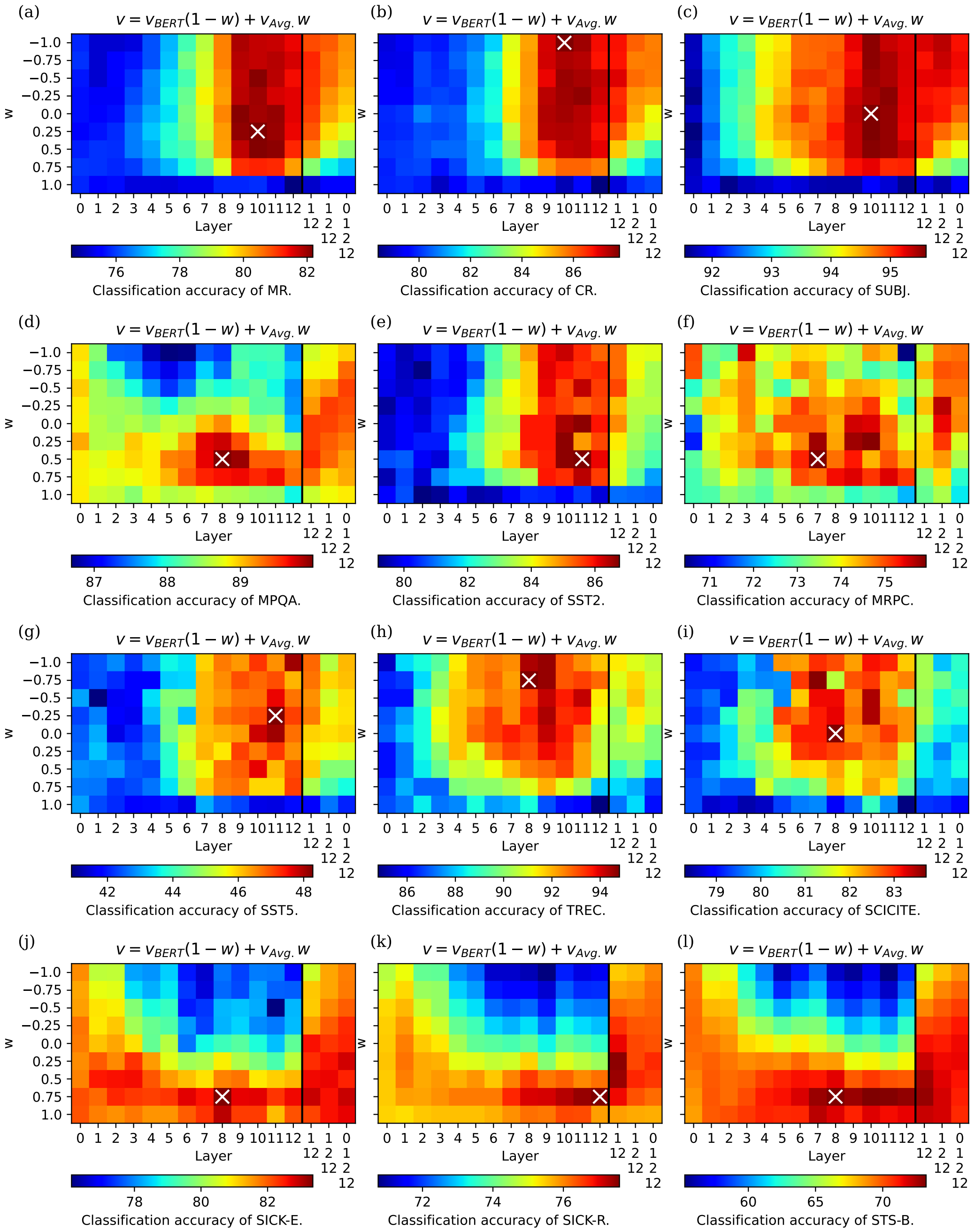

Figure 8: BERT + Avg. model individual task supervised classification performance dependence on the weight w of Avg. model and layer, from which (for both models) representations are used. To the right of the black line on the horizontal axis, average aggregation of multiple layers is also shown. Tokens are simply averaged and no post-processing is used. The horizontal line with $w = 0.0$ corresponds to a regular Bert (B) model, $w = 0.5$ is B + Avg., and $w = 1.0$ is the Avg. model. The white × marks the maximum value.