P-SIF: Document Embeddings Using Partition Averaging

Vivek Gupta\textsuperscript{1,3}, Ankit Saw\textsuperscript{2}, Pegah Nokhiz\textsuperscript{1}, Praneeth Netrapalli\textsuperscript{3}, Piyush Rai\textsuperscript{4}, Partha Talukdar\textsuperscript{5}

\textsuperscript{1}School of Computing, University of Utah, \textsuperscript{2}Info Edge (India) Limited, \textsuperscript{3}Microsoft Research Lab, Bangalore, \textsuperscript{4}Computer Science Department, IIT Kanpur, \textsuperscript{5}Indian Institute of Science, Bangalore

vgupta@cs.utah.edu, ankit.kgpian@gmail.com, pnokhiz@cs.utah.edu, praneeth@microsoft.com, piyush@cse.iitk.ac.in, ppt@iisc.ac.in

Abstract

Simple weighted averaging of word vectors often yields effective representations for sentences which outperform sophisticated seq2seq neural models in many tasks. While it is desirable to use the same method to represent documents as well, unfortunately, the effectiveness is lost when representing long documents involving multiple sentences. One of the key reasons is that a longer document is likely to contain words from many different topics; hence, creating a single vector while ignoring all the topical structure is unlikely to yield an effective document representation. This problem is less acute in single sentences and other short text fragments where the presence of a single topic is most likely. To alleviate this problem, we present P-SIF, a partitioned word averaging model to represent long documents. P-SIF retains the simplicity of simple weighted word averaging while taking a document’s topical structure into account. In particular, P-SIF learns topic-specific vectors from a document and finally concatenates them all to represent the overall document. We provide theoretical justifications on the correctness of P-SIF. Through a comprehensive set of experiments, we demonstrate P-SIF’s effectiveness compared to simple weighted averaging and many other baselines.

Introduction

Many approaches such as (Socher et al. 2013; Liu, Qiu, and Huang 2015; Le and Mikolov 2014; Ling et al. 2015) are proposed which go beyond words to capture the semantic meaning of sentences. These techniques either use the simple composition of the word-vectors or sophisticated neural network architectures for sentence representation. Recently, (Arora, Liang, and Ma 2017) proposed a smooth inverse frequency (SIF) based word vector averaging model to embed a sentence. They further improved their embedding by removing the first principal component of the weighted average vectors. However, all these approaches are limited to capturing the meaning of a single sentence and representing the sentence in the same space as words, thus reducing their expressive power. Generally, longer texts contain words from multiple topics, so creating a single vector from simple averaging of word-vectors will disregard all the topical structure. Hence, these techniques are largely unable to capture the semantic meanings of larger pieces of text, e.g., multi-sentence documents.

To address these limitations, we present a novel document embedding method called \textit{partition SIF} weighted averaging (P-SIF) to embed documents which usually contain multiple sentences efficiently. P-SIF learns topic-specific vectors from a document and finally concatenates them all to represent the overall document. Thus, P-SIF retains the simplicity of simple weighted word averaging while taking a document’s topical structure into account. We also provide theoretical justifications for the proposed approach and demonstrate its efficacy via a comprehensive set of experiments. P-SIF achieves significant improvements over several embedding techniques on several tasks despite being simple. We have released the source code for P-SIF embeddings.\textsuperscript{2}

The novel characteristics of P-SIF are described below:

- P-SIF can embed larger multi-sentence documents, as it pays attention to the topical structure of the document.
- P-SIF is based on simple weighted word vectors averaging rather than considerably more sophisticated tensor factorization or neural network-based methods.
- P-SIF is unsupervised since it only uses pre-trained word embeddings without using any label information.
- P-SIF outperforms many existing methods on text similarity, text classification, and other supervised tasks.

Related Work

Most of the prior work has computed sentence embeddings by coordinate wise vector and matrix-based compositional operations over word vectors, e.g., (Levy and Goldberg 2014) use unweighted averaging of word vectors (Le and Mikolov 2014) for representing short phrases, (Singh and Mukerjee 2015) propose tfidf-weighted averaging of word vectors to form document vectors, (Socher et al. 2013) propose a recursive neural network defined over a parse tree, and trained with supervision.

\textsuperscript{1}Topical structure denotes word distribution across topics.
\textsuperscript{2}https://github.com/vgupta123/P-SIF
Next, (Le and Mikolov 2014) propose PV-DM and PV-DBOW models which treat each sentence as a shared global latent vector. Other approaches use seq2seq models such as Recurrent Neural Networks (Mikolov et al. 2010) and Long Short Term Memory (Gers, Schraudolph, and Schmidhuber 2002) which can handle long term dependency, hence capturing the syntax structure. Other neural network models include the use of hierarchy and convolutional neural networks such as (Kim 2014). (Wieting et al. 2015) learns paraphrastic sentence embeddings by modifying word embeddings via supervision from the Paraphrase pairs dataset (PPDB) (Gantkevitch, Van Durme, and Callison-Burch 2013).

Recently, a lot of work is harnessing topic modeling (Blei et al. 2003) along with word vectors to learn better word and sentence representations, e.g., LDA (Chen and Liu 2014), weight-BoC (Kim, Kim, and Cho 2017), TWE (Liu et al. 2015), NTSG (Liu, Qiu, and Huang 2015), WM (Fu et al. 2016), w2v-LDA (Nguyen et al. 2015), TV+MeanWV (Li et al. 2016a), LTSG (Law et al. 2017), Gaussian-LDA (Das, Zaheer, and Dyer 2015), Topic2Vec (Niu et al. 2015), TM (Ding, Ruiz, and Blei 2019b), LDA2vec (Moody 2016), D-ETM (Ding, Ruiz, and Blei 2019a) and MtV (Li et al. 2016b). (Kiros et al. 2015) propose skip-thought document embedding vectors which transformed the idea of abstracting the distributional hypothesis from word to sentence level. (Wieting et al. 2016) propose a neural network model which optimizes the word embeddings based on the cosine similarity of the sentence embeddings. Moreover, several recent deep contextual word embeddings such as ELMo (Peters et al. 2018), USE (Cer et al. 2018) and BERT (Devlin et al. 2019) are proposed. These contextual embeddings are state-of-the-art on multiple tasks as they effectively capture the surrounding contexts.

(Gupta et al. 2016) propose methods which employ a clustering-based technique and tf-idf values to form a composite document vector extending the Bag-of-Words (BoW) model (Harris 1954). They represent documents in higher dimensions by using hard clustering over word embeddings. (Mekala et al. 2017) extend this by proposing SCDV using an overlapping clustering technique and direct idf weighting of word vectors. The learned representations try to capture a global context of a sentence, similar to an n-gram model. Our method is the same in essence, but is based on topic-based partitioning; moreover, unlike (Mekala et al. 2017)’s approach, our method is supported by theoretical guarantees.

### Averaging vs Partition Averaging

Figure 1, represents the word-vector space, where similar meaning words occur closer to each other. We can apply sparse coding to partition the word-vector space to a five topic vector space. These five topic vector spaces represent the five topics present in corpus. Some words are multisense and belong to multiple topics with some proportion. In Figure 1 we represent words’ topic number in subscript and proportion in brackets. Let’s consider a document $d_n$:

"Data journalists deliver data science news to general public. They often take part in interpreting the data models. In addition, they create graphical designs and interview the directors and CEOs."

If we directly average words to represent document ($\bar{v}_{dn}$), as is done in SIF (Arora, Liang, and Ma 2017), then different semantic meaning words, e.g., words in partition 1 such as ‘graphical’, ‘design’, and ‘data’ are averaged with words of different semantic meaning of partition 2 such as ‘data science’, ‘model’, and ‘data’. In addition, the document is represented in the same $d$ dimensional space as word vectors. Overall, averaging represents the documents as a single point in the vector space and does not consider the 5 different semantic topics. However, we can weight (topic proportion) average of words within a partition and concatenate ($\oplus$) the average word vectors across partitions to represent document ($\bar{v}_{dn}$), as is done in our proposed method P-SIF. By doing this, words belonging to different semantic topics are separated by concatenation ($\oplus$) as they represent separate meanings, whereas words in similar topics are simply averaged since they represent the same meaning. For example, average of words belonging to partition 1 such as ‘graphical’, ‘design’, and ‘data’ are concatenated to average of words in partition 2 such as ‘data science’, ‘model’, and ‘data’. The final document vector $\bar{v}_{dn}$ is represented in a higher $5 \times d$ dimension vector space, thus having more representational power ($d$ is the dimension of word vectors). Overall, the 5 different semantic topics are taken into account for representation. Additionally, this representation also takes the weight according to which each word belongs to various topics into account, meaning it handles words’ multi-sense natures. For example, ‘data’ belongs to partition 1 with probability 0.3 and partition 2 with probability 0.7. Hence, partitioned averaging with topic weighting is essential for representing longer text documents.
The Proposed Algorithm: P-SIF

In this section, we present the new proposed document embedding learning method in algorithm 1. The feature formation algorithm can be divided into three major steps:

Sparse Dictionary Learning for Word Vectors (Algo 1: Lines 1 - 3): Given word vectors \( \mathbf{w} \in \mathbb{R}^d \), a sparsity parameter \( k \), and an upper bound \( K \), we find a set of unit norm vectors \( \mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_K \), such that \( \tilde{\mathbf{w}} = \sum_{j=1}^{K} \alpha_{(w,j)} \mathbf{A}_j + \eta_w \) where at most \( k \) out of \( K \) of the coefficients \( \alpha_{(w,1)}, \ldots, \alpha_{(w,K)} \) are nonzero (so-called sparsity constraint), and \( \eta_w \) is a noise vector. Sparse coding is usually solved for a given \( K \) and \( k \) by using alternating minimization such as k-svd (Aharon et al. 2006) to find the \( \mathbf{A}_j \)'s that minimize the following \( L_2 \)-reconstruction error: \( \| \tilde{\mathbf{w}} - \sum_{j=1}^{K} \alpha_{(w,j)} \mathbf{A}_j \| \). (Arora et al. 2016b) show that multiple senses of a word reside as a linear superposition within the word embedding and can be recovered by simple sparse coding. Therefore, one can use the sparse coding of word vectors to detect multiple senses of words. Additionally, the atoms of sparse coding (\( \mathbf{A}_1, \ldots, \mathbf{A}_K \)) over word-vectors (\( \tilde{v}_w \)) represent all prominent topics in the corpus. For a given word \( w \), the \( k \) non-zero coefficient of \( \alpha_w \) essentially represents the distribution of words over topics. Furthermore, restricting \( K \) to be much smaller than the number of the words ensures that the same topic needs to be used for multiple words. The learned \( \mathbf{A}_j \) is a significant topic because the sparse coding ensures that each basis element is softly chosen by many words.

Sparse Dictionary Learning vs. Overlapping Clustering: Sparse coding can also be treated as a linear algebraic analogue of overlapping clustering, where the \( \mathbf{A}_j \)'s act as cluster centers and each \( \tilde{v}_w \) is assigned to each cluster in a soft way (using the coefficients \( \alpha_{(w,j)} \), of which only \( k \) out of \( K \) are non-zero) to a linear combination of at most \( k \) clusters. In practice, sparse coding optimization produces coefficients \( \alpha_{(w,j)} \) which are almost all positive, even though unconstrained. One can use overlapping clustering where each word belongs to every cluster with some probability \( P(c_k|w) \) which can be thought of as a substitute for \( \alpha_{(w,k)} \), similar to the approach in SCDV (Mekala et al. 2017). Instead of GMM, we use a dictionary learning-based approach which imposes a sparsity constraint implicitly during optimization through regularization. Additionally, such high dimensional data structure regularizers, e.g., sparse encodings, help in overcoming the curse of high dimensionality. For single-sentence documents with a small number of topics, it is better to use overlapping clustering because of an easier unconstrained optimization. However, in case of multi-sentence documents where the number of topics is large, dictionary learning performs better than overlapping clustering due to 1) Sparse constraint optimization forces non-redundant clusters (minimally sufficient #clusters) and 2) The sparse constraint diminishes the noise from the long tail of word-cluster assignments \( P(c_k|w) \) (Olshausen and Field 1997; Gao et al. 2010; Yang et al. 2009).

Word Topics Vector Formation (Algo 1: Lines 4 - 9): For single sentence documents all words of a document belong to a single topic. However, for multi-sentence documents, words of a document generally originate from multiple topics. To capture this, topic modeling algorithms such as LDA (Blei et al. 2003) are used to represent the documents. These representations essentially represent the global contexts of the documents as a distribution over topics. However, these representations do not take the local context initiating from the distributional semantics such as word vectors into account. Since our multi-sentence documents have words from multiple topics, a simple averaging technique will not work. Hence, we concatenate the word embeddings over words’ topic distributions. This helps to represent semantically similar words in the same topic, while words which are semantically different are represented in different topics. Concatenation of word embeddings over topics also helps in the expression of words’ multi-sense nature. For each word \( \tilde{w} \), we create \( K \) different word-cluster vectors \( \tilde{c}_{v wk} \) by weighting the word embedding with its learned dictionary coefficient \( \alpha_{w,k} \) of the \( k^{th} \) context.\(^\text{3}\) We then concatenate all the \( K \) word-cluster vectors \( \tilde{c}_{v wk} \) into a \( K \times d \) dimensional embedding to form a word-topic vector \( \tilde{tv}_w \in \mathbb{R}^{K \times d} \). We weigh word-vectors by coefficients of the learned dictionary to capture the cross correlation (\( \alpha_i \alpha_j \)) between topics. The word-topic-vector \( \tilde{tv}_w \), which we average to represent documents, captures both local and global semantics.

SIF Weight Averaging and Common Component Removal (Algo 1: Lines 10 - 16): Finally, for all words appearing in document \( D_n \), we weight the word-vectors \( \tilde{v}_w \) by smooth inverse frequency \( \frac{a}{a+p(w)} \). Next, we remove the common contexts from the weighted average document vectors by removing the first principal component from the weighted average vectors.\(^\text{4}\) Common component removal reduces the noise and redundancy from the document vectors which makes the representations more discriminating. (Arora, Liang, and Ma 2017) empirically shows that SIF weighting outperforms the tf-idf weighting. However, they only use simple averaging to represent a sentence. Detailed code architecture of P-SIF is in the supplementary material.\(^\text{5}\)

Derivation of P-SIF Embeddings: We provide theoretical justifications by showing connections of P-SIF with random-walk based latent variable models (Arora, Liang, and Ma 2017; Arora et al. 2016a; 2016b). Full derivations are provided in the supplementary material.\(^\text{5}\)

Kernels meet Embeddings

In this section, we present one of the novelties of this work where we show that many common sentence embeddings can be represented as similarity kernels over word and topic vectors. Let \( D_A \) and \( D_B \) represent two documents containing \( n \) and \( m \) words respectively, \( w_1^A, w_2^A, \ldots, w_n^A \) denotes \( D_A \)'s words and \( w_1^B, w_2^B, \ldots, w_m^B \) denotes \( D_B \)'s words.

\(^\text{3}\) Empirically, we observed that this weighting generally improves the performance.\(^\text{4}\) We did not remove the common component from final vectors when we used Doc2VecC-initialized (Chen 2017) word vectors with P-SIF. Because frequent words’ word-vectors become close to \( \mathbf{0} \).\(^\text{5}\) https://vgupta123.github.io/docs/appendix_aaa2020.pdf
4. Relaxed Word Mover Distance (Kusner et al. 2015)

2. TWE: Topical Word Embeddings (Liu et al. 2015)

1. Simple Word Vector Averaging:

\[ \vec{v}_w = \frac{1}{|\text{doc}|} \sum_{d \in \text{doc}} \frac{1}{n} \sum_{w \in d} \alpha_{w,j} \vec{A}_j + \vec{v}_w; \]

for each word \( w \) in \( V \) do

1. Form a matrix \( \vec{v}_w \) for each word \( w \) in \( V \).
2. \( \vec{v}_w = \sum_{j=1}^{K} \alpha_{w,j} \vec{A}_j + \vec{v}_w; \)
3. end

4. for each word \( w \) in \( V \) do
5. \( \vec{v}_w = \sum_{j=1}^{K} \alpha_{w,j} \vec{A}_j + \vec{v}_w; \)
6. end

7. Each word \( w \) has \( K \) contributions from each \( \vec{A}_j \) and a vector \( \vec{v}_w \) for each word \( w \).
8. \( \vec{v}_w = \sum_{j=1}^{K} \alpha_{w,j} \vec{A}_j + \vec{v}_w; \)
9. end

10. Each \( \vec{v}_w \) is a vector formed by \( \vec{v}_w \) and \( \sum_{j=1}^{K} \alpha_{w,j} \vec{A}_j \).
11. \( \vec{v}_w \) is a vector formed by \( \vec{v}_w \) and \( \sum_{j=1}^{K} \alpha_{w,j} \vec{A}_j \).
12. end

13. Form a matrix \( X \) whose columns are \( \{\vec{v}_{d,n}, d \in \text{doc}\} \),
14. and \( \vec{u} \) is the first singular vector;
15. for each document \( d \) in \( \text{do} \) do
16. \( \vec{v}_{d,n} \leftarrow \frac{1}{|\text{doc}|} \sum_{w \in d} \alpha_{w,j} \vec{A}_j \vec{v}_w; \)
17. end

18. Experimental Results

We perform a comprehensive set of experiments on several text similarity and multiclass or multilabel text classification tasks. Due to limited space, some details on experiments are in the Supplementary material 5.

Experimental Settings:
We use the Pearson’s coefficient between the predicted and the ground-truth scores for the

Textual Similarity Task

Datasets and Baselines: We perform our experiments on the SemEval dataset (2012 - 2017). These experiments involve 27 semantic textual similarity (STS) tasks (2012 - 2016) (Agirre et al. 2012; 2016) the SemEval 2015 Twitter task (Xu, Callison-Burch, and Dolan 2015), and the SemEval 2014 Semantic relatedness task (Marelli et al. 2014). The objectives of these tasks are to predict the similarity between two sentences. We compare our approach with several unsupervised, semi-supervised and supervised embedding baselines mostly taken from (Arora, Liang, and Ma 2017; Wu et al. 2018; Ethayarajh 2018). Details on the baselines are listed below:

Unsupervised: We used ST, avg-GloVe, tfidf-GloVe, and GloVe + WR as a baseline. ST denotes the skip-thought vectors by (Kiros et al. 2015), avg-GloVe denotes the unweighted average of the GloVe Vectors by (Pennington, Socher, and Manning 2014) 6, and tfidf-GloVe denotes the tf-idf weighted average of GloVe vectors. We also compared our method with the SIF weighting (W) common component removal (R) GloVe vectors (GloVe + WR) by (Arora, Liang, and Ma 2017). For STS 16, we also compared our embedding with Skip-THoughts (Kiros et al. 2015), BERT pretrained embedding average (Devlin et al. 2019), Universal Sentence Encoder (Cer et al. 2018) and SenT2Vec (Pagliardini, Gupta, and Jaggi 2018) embeddings.

Supervised: We used avg-PSL, PSL + WR, and the avg-PSL used the unweighted average of the PARAGRAM-SL999 (PSL) word vectors by (Wieting et al. 2015) as a baseline, obtained by training on PPDB dataset (Ganitkevitch, Van Durme, and Callison-Burch 2013). The word vectors are trained using unlabeled data. Furthermore, sentence embeddings are obtained from unweighted word vectors averaging. We also compared our method with the SIF weighting (W) common component removal (R) PSL word vectors (PSL + WR) by (Arora, Liang, and Ma 2017).

Experimental Settings:
We use the Pearson’s coefficient between the predicted and the ground-truth scores for the

Algorithm 1: P-SIF Embedding

Data: \( d \) dimensional Word embeddings \( \{\vec{v}_w : w \in V\} \)
where word \( w \) is in vocabulary \( V \). Documents \( \{d_n : d_n \in D\} \), a set of sentences \( D \) in corpus \( C \), parameter \( a \) and estimated unigram probability \( \{p(w) : w \in V\} \) of word \( w \) in \( V \), a sparsity parameter \( k \), and an upper bound \( K \).

Result: Document vectors \( \{\vec{v}_{d,n} : d_n \in D\} \)

/* Dictionary learning on word-vectors */
1. for each word \( w \) in \( V \) do
2. \( \vec{v}_w = \sum_{j=1}^{K} \alpha_{w,j} \vec{A}_j + \vec{v}_w; \)
3. end

4. for each word \( w \) in \( V \) do
5. \( \vec{v}_w = \sum_{j=1}^{K} \alpha_{w,j} \vec{A}_j + \vec{v}_w; \)
6. end

7. Form a matrix \( \vec{v}_w \) for each word \( w \) in \( V \).
8. \( \vec{v}_w = \sum_{j=1}^{K} \alpha_{w,j} \vec{A}_j + \vec{v}_w; \)
9. end

7. Each \( \vec{v}_w \) is a vector formed by \( \vec{v}_w \) and \( \sum_{j=1}^{K} \alpha_{w,j} \vec{A}_j \).
8. \( \vec{v}_w = \sum_{j=1}^{K} \alpha_{w,j} \vec{A}_j + \vec{v}_w; \)
9. end

10. Each \( \vec{v}_w \) is a vector formed by \( \vec{v}_w \) and \( \sum_{j=1}^{K} \alpha_{w,j} \vec{A}_j \).
11. \( \vec{v}_w = \sum_{j=1}^{K} \alpha_{w,j} \vec{A}_j + \vec{v}_w; \)
12. end

13. Form a matrix \( X \) whose columns are \( \{\vec{v}_{d,n} : d_n \in D\} \),
14. and \( \vec{u} \) is the first singular vector;
15. for each document \( d \) in \( \text{do} \) do
16. \( \vec{v}_{d,n} \leftarrow \frac{1}{|\text{doc}|} \sum_{w \in d} \alpha_{w,j} \vec{A}_j \vec{v}_w; \)
17. end

18. Experimental Results

We perform a comprehensive set of experiments on several text similarity and multiclass or multilabel text classification tasks. Due to limited space, some details on experiments are in the Supplementary material 5.

1. Simple Word Vector Averaging:

\[ K^1(D_A, D_B) = \frac{1}{nm} \sum_{i=1}^{n} \sum_{j=1}^{m} (\vec{v}_{iA} \cdot \vec{v}_{iB}) \]

2. TWE: Topical Word Embeddings (Liu et al. 2015):

\[ K^2(D_A, D_B) = \frac{1}{nm} \sum_{i=1}^{n} \sum_{j=1}^{m} (\vec{v}_{iA} \cdot \vec{v}_{iB}) + (\vec{t}_{iA} \cdot \vec{t}_{iB}) \]

3. P-SIF: Partition Word Vector Averaging (Our approach):

\[ K^3(D_A, D_B) = \frac{1}{nm} \sum_{i=1}^{n} \sum_{j=1}^{m} (\vec{v}_{iA} \cdot \vec{v}_{iB}) - (\vec{t}_{iA} \cdot \vec{t}_{iB}) \]

4. Relaxed Word Mover Distance (Kusner et al. 2015):

\[ K^4(D_A, D_B) = \frac{1}{nm} \sum_{i=1}^{n} \max_j (\vec{v}_{iA} \cdot \vec{v}_{iB}) \]

Here, \( \vec{v}_w \) represents the word vector of word \( w \) and \( \vec{t}_w = \alpha_{w} \in R^K \) represents the topic vector of word \( w \), where \( K \) is the number of topics. \((\vec{a}, \vec{b}) \) represents the dot product of two vectors \( \vec{a} \) and \( \vec{b} \). \( c \times d \) represents the scalar product of \( c \) and \( d \). \( \bigoplus \) represents the row-wise concatenation of the vectors.

Refer to the Supplementary material 5 for the detailed proof.

6 We used the 300-dimensional word vectors that are publicly available at http://nlp.stanford.edu/projects/glove/.
evaluation. We use the PARAGRAM-SL999 (PSL) from (Wieting et al, 2015) as word embeddings, obtained by training on the PPDDB (Ganitkevitch, Van Durme, and Callison-Burch 2013) dataset 7. We use the fixed weighting parameter \( \alpha \) value of \( 10^{-3} \), and the word frequencies \( p(w) \) are estimated from the common-crawl dataset. We tune the number of contexts \( (K) \) to minimize the reconstruction loss over the word-vectors. We fix the non-zero coefficient \( k = K/2 \), for the SIF experiments. For the GMM-based partitioning of the vocabulary, we tune the number of clusters’ parameter \( K \) through a 5-fold cross validation.

**Results and Analysis:** The average results for each year are reported in Tables 1 and 2. We denote our embeddings by P-SIF + PSL (+ PSL denotes using the PSL word vectors). We report the average results for the STS tasks. The detailed results on each sub-dataset are in the Supplementary material 5. We observe that P-SIF + PSL outperforms PSL + WR on all datasets, thus supporting the usefulness of our partitioned averaging. Despite being simple, our method outperforms many complicated methods such as seq2seq, Tree-LSTM(Tai, Socher, and Manning 2015), and SkipThoughts(Kiros et al, 2015). We observe that partitioning through overlapping clustering algorithms such as GMM generates a better performance compared to partitioning through sparse dictionary algorithms such as k-svd for some Semantic Textual Similarity (STS) task datasets. The main reason for this peculiar observation is related to the fact that some STS datasets contain documents which are single sentences of a maximum length of 40 words. As discussed earlier (sparse dictionary learning vs. overlapping clustering), for single sentence documents with a small number of topics, overlapping clustering optimizes better than sparse dictionary learning. Therefore, we use GMM for partitioning words into suitable clusters for some STS tasks. But both k-svd and GMM outperform simple averaging (SIF) by significant margins on most STS tasks. 8 We also report qualitative results with real examples in the Supplementary 5.

**Text Classification Task**

The document embeddings obtained by our method can be used as direct features for many classification tasks.

**Datasets and Baselines:** We run multi-class experiments on 20NewsGroup dataset, 9 and multi-label classification experiments on Reuters-21578 dataset. 10 We use script for preprocessing the dataset. 11 We consider several embedding baselines mostly taken from (Mekala et al, 2017; Wu et al, 2018; Arora et al, 2016b). More details on experimental settings and hyper-parameters’ values are described in the Supplementary material 5. We considered the following baselines: The Bag-of-Words (BoW) model (Harris 1954), the Bag of Word Vector (BoWV) (Gupta et al, 2016) model, Sparse Composite Document Vector (SCDV) (Mekala et al, 2017) 12 paragraph vector models (Le and Mikolov 2014), Topical word embeddings (TWE-1) (Liu et al, 2015), Neural Tensor Skip-Gram Model (NTSG-1 to NTSG-3) (Liu, Qiu, and Huang 2015), tf-idf weighted average word-vector model(Singh and Mukerjee 2015) and weighted Bag of Concepts (weight-BoC) (Kim, Kim, and Cho 2017) where we built document-topic vectors by counting the member words in each topic, and Doc2VecC (Chen 2017) where averaging and training of word vectors are done jointly. Moreover, we used SIF (Arora, Liang, and Ma 2017) smooth inverse frequency weight with common component removal from weighted average vectors as a baseline. We also compared our results with other topic modeling based document embedding methods such as WTM (Fu et al, 2016), w2v-LDA (Nguyen et al, 2015), LDA (Chen and Liu 2014), TV+MeanWV (Li et al, 2016a), LTSG (Law et al, 2017), Gaussian-LDA (Das, Zaheer, and Dyer 2015), Topic2Vec (Niu et al, 2015), Lda2Vec (Moody, 2016), MtVM (Li et al, 2016b) and BERT (Devlin et al, 2019). For BERT, we reported the results on the unsupervised pre-trained (pr) model because of a fair comparison to our approach which is also unsupervised.

**Experimental Settings:** We fix the document embeddings and only learn the classifier. We learn word vector embeddings using Skip-Gram with a window size of 10, Negative Sampling (SGNS) of 10, and minimum word frequency of 20. We use 5-fold cross-validation on the F1 score to tune hyperparameters. We use LinearSVM for multi-class classification and Logistic regression with the OneVsRest setting for multi-label classification. We fix the number of dictionary elements to either 40 or 20 (with Doc2vecC initialize word vectors) and non-zero coefficient to \( k = K/2 \) during dictionary learning for all experiments. We use the best parameter settings, as reported in all our baselines to generate their results. We use 200 dimensions for tf-idf weighted word-vector model, 400 for paragraph vector model, 80 topics and 400 dimensional vectors for TWE, NTSG, LTSG and 60 topics and 200 dimensional word vectors for SCDV (Mekala et al, 2017). We evaluate the classifiers’ performance using standard metrics such as accuracy, macro-averaging precision, recall and F-score for multiclass classification tasks. We evaluate multi-label classifications’ performance using Precision@K, nDCG@k, Coverage error, Label ranking average precision(LRAPS) and F1 score. 13

**Results and Analysis:** We observe that P-SIF outperforms all other methods by a significant margin on both 20NewsGroup (Table 4) and Reuters (Table 5). We observe that the dictionary learns more diverse and non-redundant topics compared to overlapping clustering (SCDV) since we require only 40 partitions rather than 60 partitions in SCDV to obtain the best performance. Simple tf-idf weighted averaging-based document representations do not show significant improvement in performance by increasing word vector dimensions. We achieve a \( < 0.4\% \) improvement in the accuracy when the word-vector dimensions increase from 200 to 500 on 20NewsGroup. We observe that increasing the word-vectors’ dimensions beyond 500 does not improve SIF and P-SIF’s performances. We further improve the performance on both datasets using

---

7 For a fair comparison with SIF we use PSL vectors instead of unsupervised GloVe and Word2Vec vectors. 8 k-svd always outperforms GMM on both classification datasets since the documents are multi-sentence with \#words >> 40. 9 https://bit.ly/2pqLCaN 10 https://goo.gl/NrOfu 11 https://gist.github.com/herrfiz/7967778 12 https://github.com/dheeraj7596/SCDV 13 https://goo.gl/4GrR3M
Table 1: Experimental results (Pearson’s r × 100) on textual similarity tasks. Many results are collected from (Wieting et al. 2016), DAN (Iyyer et al. 2015) and (Wieting and Gimpel 2017) (GRAN) except for tfidf-GloVe.

| Tasks | PP | PP proj | DAN | RNN | iRNN | LSTM (no) | LSTM (o.g.) | GRAN | ST | Avg Glove | tfidf Glove | Avg PSL | Glove +WR | PSL +WR | PSIF +WR |
|-------|----|---------|-----|-----|-----|-----------|-------------|-------|---|-----------|-------------|--------|--------|--------|--------|
| STS12 | 58.7 | 60.0 | 56.0 | 48.1 | 58.4 | 51.0 | 46.4 | 62.5 | 30.8 | 52.5 | 58.7 | 52.8 | 56.2 | 59.5 | 65.7 |
| STS13 | 55.8 | 56.8 | 54.2 | 44.7 | 56.7 | 45.2 | 41.5 | 63.4 | 24.8 | 42.3 | 52.1 | 46.4 | 56.6 | 61.8 | 64.0 |
| STS14 | 70.9 | 71.3 | 69.5 | 57.7 | 70.9 | 59.8 | 51.5 | 75.9 | 31.4 | 54.2 | 63.8 | 59.5 | 68.5 | 73.5 | 74.8 |
| STS15 | 75.8 | 74.8 | 72.7 | 57.2 | 75.6 | 63.9 | 56.0 | 77.7 | 31.0 | 52.7 | 60.6 | 60.0 | 71.7 | 76.3 | 77.3 |
| Sick14 | 71.6 | 71.6 | 70.7 | 61.2 | 71.2 | 63.9 | 59.0 | 72.9 | 49.8 | 65.9 | 69.4 | 66.4 | 72.2 | 72.9 | 73.4 |
| Twi15 | 52.9 | 52.8 | 53.7 | 45.1 | 52.9 | 47.6 | 36.1 | 50.2 | 24.7 | 30.3 | 33.8 | 36.3 | 48.0 | 49.0 | 54.9 |

Table 2: P-SIF comparison with the recent embedding techniques on various STS tasks. Baselines taken from (Conneau and Kiela 2018), (Perone, Silveira, and Paula 2018), (Cer et al. 2018), (Devlin et al. 2019), (Wu et al. 2018) and (Ethayarajh 2018).

| Task | ELMo orig+all | ELMo orig+top | Bert(pr) Avg. | USE | p-mean | Fast Text | Skip Thoughts | Infer Sent | Char pharse | WME +PSL | PSIF +PSL | u-SIF +PSL |
|------|---------------|---------------|---------------|-----|--------|-----------|---------------|------------|------------|----------|----------|-----------|
| STS 12 | 35 | 54 | 53 | 65 | 54 | 41 | 61 | 66 | 62.8 | 63.9 | 76.8 |
| STS 13 | 51 | 49 | 67 | 68 | 52 | 29 | 56 | 57 | 56.3 | 63.9 | 76.8 |
| STS 14 | 63 | 62 | 62 | 64 | 63 | 40 | 68 | 74.7 | 68.0 | 74.8 | 79.5 |
| STS 15 | 69 | 67 | 73 | 77 | 66 | 46 | 71 | 76.1 | 64.2 | 77.3 | 77.6 |
| STS 16 | 64 | 63 | 67 | 73 | 67 | 64 | 52 | 77 | - | - | 73.7 |
| Average | 60.4 | 59 | 64.4 | 69.4 | 60.4 | 62.6 | 66.6 | 68.5 | 62.9 | 71.1 | 71.4 |

Table 3: Comparison of P-SIF (SGNS) with the recently proposed word mover distance and word mover embedding (Wu et al. 2018) based on accuracy. In ±x, x is the variance across several runs.

| Dataset | Bbcsport | Twitter | Observed | Classic | Reuters | Amazon | 20NG | Recipe-L |
|---------|----------|---------|----------|---------|---------|--------|------|-----------|
| SIF(GloVe) | 97.3 ± 1.2 | 57.8 ± 2.5 | 67.1 | 92.7 ± 0.9 | 87.6 | 94.1 ± 0.2 | 72.3 | 71.1 ± 0.5 |
| Word2Vec Avg | 97.3 ± 0.9 | 72.0 ± 1.5 | 63 | 95.2 ± 0.4 | 96.9 | 94.0 ± 0.5 | 71.7 | 74.9 ± 0.5 |
| PV-DBOW | 97.2 ± 0.7 | 67.8 ± 0.4 | 55.9 | 97.0 ± 0.3 | 96.3 | 89.2 ± 0.3 | 71 | 73.1 ± 0.5 |
| PV-DM | 97.9 ± 1.3 | 67.3 ± 0.3 | 59.8 | 96.6 ± 0.7 | 94.9 | 88.6 ± 0.4 | 74 | 71.1 ± 0.4 |
| Doc2VecC | 90.5 ± 1.7 | 71.0 ± 0.4 | 63.4 | 96.6 ± 0.4 | 96.5 | 91.2 ± 0.5 | 78.2 | 76.1 ± 0.4 |
| KNN-WMD | 95.4 ± 1.2 | 71.3 ± 0.6 | 55.5 | 97.2 ± 0.1 | 96.5 | 92.6 ± 0.3 | 73.2 | 71.4 ± 0.5 |
| SCDV | 98.1 ± 0.6 | 74.2 ± 0.4 | 53.5 | 96.9 ± 0.1 | 97.3 | 93.9 ± 0.4 | 78.8 | 78.5 ± 0.5 |
| JWE | 98.2 ± 0.6 | 74.5 ± 0.5 | 64.5 | 97.1 ± 0.4 | 97.2 | 94.3 ± 0.4 | 78.3 | 79.2 ± 0.3 |

| P-SIF | 99.05 ± 0.9 | 73.39 ± 0.9 | 67.1 | 96.95 ± 0.5 | 97.67 | 94.17 ± 0.3 | 79.15 | 78.24 ± 0.3 |
| P-SIF (Doc2VecC) | 99.68 ± 0.9 | 72.39 ± 0.9 | 67.1 | 97.7 ± 0.5 | 97.62 | 94.83 ± 0.3 | 86.31 | 77.61 ± 0.3 |

Doc2vecC-initialized (Chen 2017) word-vectors which reduce word level noise in the P-SIF representations. We represent this approach by P-SIF (Doc2VecC) in Table 4 and Table 5. On 20NewsGroup, we require only 20 partitions instead of 40 with Doc2vecC-initialized word vectors. This shows that better word vector representations help in learning more diverse and non-redundant partitions. We also report our results (micro-F1) on each of the 20 classes of 20NewsGroup in the Supplementary material. Additionally, we empirically show that our proposed embedding P-SIF outperforms the word mover distance (Kusner et al. 2015) and performs comparable with the word mover embedding (Wu et al. 2018) in Table 3. Overall, P-SIF outperforms most methods on several datasets by a significant margin.

Comparison with Contextual Embeddings: Despite its simplicity, P-SIF is able to outperform unsupervised contextual embeddings such as BERT (pr) and ELMo. We assume the reason behind this is P-SIF’s focused ability to effectively capture both global and local semantics in sparse higher dimension representations. On other hand, BERT tries to capture both syntax and semantics in single lower dimensional continuous representations. In both classification and similarity tasks, understanding syntax is not as prominent as understanding semantics.

Analysis and Discussion

Effect of Document-Length: We conduct a small experiment to show that our model performs better compared to SIF when we have large size documents. We have divided 26 STS datasets by average document length, i.e., the number of words in documents in bins of (10 – 20, 20 – 30, 30 – 40, 40 – 50) words. Next, we average the relative performance improvement by P-SIF and SCDV by accuracy with respect to SIF (Method – SIF %) for datasets in each bin. In Figure 2, we observe that for complex multi-sentence documents with more words, P-SIF performs relatively better than SCDV.
We also note that short texts require fewer number of partitions to achieve their best performance which is quite intuitive since short text documents will map into fewer topics.

**Effect of Sparse Partitioning:** Partitioning and concatenation of word embeddings over topics also helps in the representation of multi-sense words, which would have been left-out by simple averaging of the word embeddings in document representation otherwise. Empirically, on both datasets, we observe that the dictionary learns more diverse document representation as compared to overlapping clusters of embeddings (mostly non-sparse). Thus, we obtain a performance gain (F1-Score) of 1.5% with less than 0.33 of the size of the SCDV embeddings. Lastly, due to fewer dimensions, the feature formation time is less in P-SIF.

### Conclusions and Future Work

We propose a novel unsupervised document feature formation technique based on partitioned word vector averaging. Our embedding retains the simplicity of simple weighted word averaging while taking documents' topical structure into account. Our simple and efficient approach achieves significantly better performance on several textual similarity and textual classification tasks, e.g., we outperform contextual embeddings such as BERT (pr) and ELMo. One limitation of our work is its ignorance of words' order and syntax. In the future, we plan to address this problem and model partitioning, averaging, and learning as a joint process.

### References

Agirre, E.; Diab, M.; Cer, D.; and Gonzalez-Agirre, A. 2012. SemEval-2012 task 6. In SemEval, 385–393. ACL.

Agirre, E.; Banea, C.; Cer, D.; Diab, M.; Gonzalez-Agirre, A.; Mihalcea, R.; Rigau, G.; and Wiebe, J. 2016. SemEval-2016. In SemEval-2016, 497–511.

Aharon, M.; Elad, M.; Bruckstein, A.; et al. 2006. K-svd: An algorithm for designing overcomplete dictionaries for sparse representation. IEEE Transactions on signal processing 54(11):4311.

Arora, S.; Li, Y.; Liang, Y.; Ma, T.; and Risteski, A. 2016a. A latent variable model approach to pmi-based word embeddings. TACL.

Arora, S.; Li, Y.; Liang, Y.; Ma, T.; and Risteski, A. 2016b. Linear algebraic structure of word senses, with applications to polysemy. arXiv preprint arXiv:1601.03764.

Arora, S.; Liang, Y.; and Ma, T. 2017. A simple but tough-to-beat baseline for sentence embeddings. ICLR.
Wieting, J.; Bansal, M.; Gimpel, K.; and Livescu, K. 2016. Towards universal paraphrastic sentence embeddings. *ICLR*.

Wu, L.; Yen, I. E.-H.; Xu, K.; Xu, F.; Balakrishnan, A.; Chen, P.-Y.; Ravikumar, P.; and Witbrock, M. J. 2018. Word mover’s embedding: From word2vec to document embedding. In *EMNLP*.

Xu, W.; Callison-Burch, C.; and Dolan, B. 2015. SemEval-2015 task 1. In *SemEval* 2015, 1–11.

Yang, J.; Yu, K.; Gong, Y.; and Huang, T. 2009. Linear spatial pyramid matching using sparse coding for image classification. In *CVPR*, 1794–1801. IEEE.

### Derivation of P-SIF Embeddings

To derive the P-SIF embedding, we propose a generative model which treats corpus generation as a dynamic process where the $t^{th}$ word is produced at step $t$. This process is driven by random walk over a unit norm sphere with the center at the origin. Let $\vec{v}_w$ be the $d$ dimensional vector from the origin to the current walk point at time $t$. We call this vector the context vector $\vec{v}_w$, as it represents the context in the discussion. Let $Z_t$, represent the partition function for the random context vector $\vec{v}_w$, given by $Z_t = \sum_{\vec{v}_w} \exp(\langle \vec{v}_w, \vec{c}_t \rangle)$. $c_0$ and $\vec{v}_w$ represent a common context and its corresponding $d$ dimensional context vector based on syntax.

Using log linear model of (Mnih and Hinton 2007), we define the probability of observing a word $w$ from the random walk with current context $c_t$ at time $t$ as

$$Pr[w|c_t] \propto \exp(\langle \vec{c}_t, \vec{v}_w \rangle)$$  \hspace{1cm} (1)

It is easy to show that such random walk under some reasonable assumptions (Arora et al. 2016a) can give word-word co-occurrence probabilities similar to empirical works like word2vec (Mikolov et al. 2013) and GloVe (Pennington, Socher, and Manning 2014). To account for frequent stop-words which occur more often regardless of context and the common context related to document syntax, two correction terms need be added: one based on $p(w)$ and the other on the common context vector $\vec{v}_0$ in Equation (1). These terms allow words with a low inner product with $\vec{c}_0$ a chance to appear either from $p(w)$ if they are frequent or by the common context $\vec{c}_0$ if they have a large dot product with $\vec{c}_0$. Given a context vector $c_t$, the probability of a word $w$ in document $d$ being generated by context $c_t$ is given by,

$$Pr[w|c_t] = \lambda p(w) + (1 - \lambda) \frac{\exp(\langle \vec{c}_t, \vec{v}_w \rangle)}{Z_{c_t}}$$  \hspace{1cm} (2)

where $\vec{c}_t = \beta \vec{c}_0 + (1 - \beta) \hat{c}_t$, $(\vec{c}_0, \hat{c}_t) = 0$, $\lambda$ and $\beta$ are scalar hyper-parameters.

For generating a document from the above random walk-based latent variable model, we consider the following assumptions:

1. Context vector $(\vec{v}_w)$ does not change significantly while words are generated from the random walk, as shown by (Arora, Liang, and Ma 2017), except the jumps due to topic change.
2. Total number of topics in the entire corpus is $K$, which can be determined by sparse dictionary learning as shown by (Arora et al. 2016b) over word vectors $\vec{v}_w$.
3. Word vectors $\vec{v}_w$ are uniformly distributed, thus making the partition function $Z_{c_t}$ roughly the same in all directions for a given context $c$, emerging from each of the $K$ topics.

For a document $d$, the likelihood of document is being generated by the $K$ contexts is given by:

$$p(d|\{c_1, c_2 \ldots c_K\}) \propto \prod_{j=1}^{K} \prod_{(w|d)} p(w|c_j)$$  \hspace{1cm} (3)

$$= \prod_{j=1}^{K} \prod_{w \in d} [\lambda p(w) + (1 - \lambda) \frac{\exp(\langle \vec{v}_w, \vec{v}_{c_j} \rangle)}{Z_j}]$$  \hspace{1cm} (4)

Let, $f_w(c_j) = \log [\lambda p(w) + (1 - \lambda) \frac{\exp(\langle \vec{v}_w, \vec{v}_{c_j} \rangle)}{Z_j}]$  \hspace{1cm} (5)

Here, $p(w|c_j)$ is the probability that word $w$ is generated by context $c_j$, the value of which is determined by 1) The overall frequency of word $w$ in the corpus, i.e., prior probability $(p(w))$ and 2) The relative frequency of $w$ appearing with context $j$ with respect to other contexts (determined by $\alpha(w, j)$).

Using simple algebra and treating $p(w)$ as a constant, we can show that $V(f_w(c_j))$ equals,

$$\frac{1}{\lambda p(w) + (1 - \lambda) \frac{\exp(\langle \vec{v}_w, \vec{v}_{c_j} \rangle)}{Z_j}} \cdot \frac{1 - \lambda}{Z_j} \exp(\langle \vec{v}_w, \vec{v}_{c_j} \rangle) \vec{v}_w$$  \hspace{1cm} (6)

Then, by using the Taylor expansion, we can show

$$f_w(c_j) \approx f_w(c_j = 0) + \nabla f_w(c_j = 0)^T \vec{v}_{c_j}$$  \hspace{1cm} (7)

$$f_w(c_j) \approx \text{constant} + \nabla f_w(c_j = 0)^T \vec{v}_{c_j}$$  \hspace{1cm} (8)

Therefore, the maximum likelihood estimator (MLE) for $\vec{v}_{c_j}$ on the unit sphere (ignoring normalization) given $a = \frac{1 - \lambda}{\lambda Z_j}$ is approximately 15

$$\arg\max_{w} \sum_{u \in d} f_w(c_j) \propto \sum_{w \in d} a \frac{p(w) + a \vec{v}_w}{p(w) + a}$$  \hspace{1cm} (9)

Thus, the MLE estimate is approximately a weighted average of the word-vectors generated from context $j$ in document $d$ from the random walk. We can get the overall context representation $\vec{v}_{c_d}$ of the document by simple concatenation over all $K$ topics i.e. $\vec{v}_{c_d} = \bigoplus_{k=1}^{K} \vec{v}_{c_k}$. Here, $\bigoplus$ represents the concatenation operation. For a document if no word is generated from the context $c_j$ then we can substitute the context vector $\vec{v}_{c_j}$ by a $0$ vector to represent $\vec{v}_{c_d}$ in $K \times d$ dimensions. The embedding of a sentence can be obtained by $\vec{v}_{s} = \sum_{(w|d)} \frac{p(w) + a}{p(w) + a} \vec{v}_w$ where $a = \frac{1 - \lambda}{\lambda Z_j}$.

### Relation to SIF model:

(Arora, Liang, and Ma 2017) show sentences can be represented as averaging of word vectors, under the two assumptions:

- uniform distribution of word vectors $\vec{v}_w$ which implies that the partition function $Z_t$ is roughly the same in all directions for a sentence.
- context vector $\vec{v}_{c_d}$ remains constant while the words in the sentence are emitted, implying the replacement of $\vec{v}_{c_d}$ in the sentence by $\vec{v}_{s}$ and partition function $Z_t$ by $Z_s$.

However, the above assumptions do not hold true for a document with multiple sentences where one can expect to have more frequent jumps during a random walk due to topic change. Instead of assuming a single context for the whole document $c_d$, we assume that the total number of contexts over a given corpus is bounded by the number of topics $K$ (as shown by (Arora et al. 2016b)), and the random walk can perform jumps to switch context from one context to the rest of $K - 1$ contexts. The partition function remains the same in all directions only for the words emerging from the same context $c_j$ instead of the words coming from all the $K$ contexts. Thus, our approach is a strict generalization of the sentence embedding approach by (Arora, Liang, and Ma 2017) which is a special case of $K = 1$.

15 Note that $\arg\max_{c_j|c_j = 1} C + \langle \vec{c}_j, \hat{g} \rangle = \frac{\hat{g}}{|\hat{g}|}$ for any constant $C$

16 It is trivial to assume that these jumps occur more frequently in multi-sentence documents because of more number of topics.
Details of Textual Similarity Task
In this supplementary section, we present the details of the STS tasks for each year. Each year, there are 4 to 6 STS tasks, as shown in Table 6. Note that tasks with the same name in different years are different tasks in reality. We provide detailed results for each task in STS 12 - 15 in Table 7. Our method outperforms all other methods from (Arora, Liang, and Ma 2017) and (Wieting et al. 2016) on all 16 out of 22 tasks. Our method performs significantly better in comparison to all unsupervised embedding methods. In addition, P-SIF is very close to the best performance by supervised methods on the rest of the datasets. Our method was also able to outperform state of the art supervised averaging based Gated Recurrent Averaging Network (GRAN) (Wieting and Gimpel 2017) on 11 datasets shown in Table 7. Our results also outperform state of the art methods on many recent supervised embedding methods on the STS 16 task (See Table 8).

Experimental Details

Textual Similarity Task:
We use the PARAGRAM-SL999 (PSL) from (Wieting et al. 2015) as word embeddings, obtained by training on the PPDB (Ganitkevitch, Van Durme, and Callison-Burch 2013) dataset 17. We use the fixed weighting parameter α value of $10^{-3}$, and the word frequencies $p(w)$ are estimated from the common-crawl dataset. We tune the number of contexts ($k$) to minimize the reconstruction loss over the word-vectors. We fix the non-zero coefficient $k = K/2$, for the SIF experiments. For the GMM-based partitioning of the vocabulary, we tune the number of clusters’ parameter $K$ through a 5-fold cross validation.

1. Unsupervised: We used ST, avg-GloVe, tfidf-GloVe, and GloVe + WR as a baseline. ST denotes the skip-thought vectors by (Kiros et al. 2015), avg-GloVe denotes the unweighted average of the GloVe Vectors by (Pennington, Socher, and Manning 2014) 18, and tfidf-GloVe denotes the tf-idf weighted average of GloVe vectors. We also compared our method with the SIF weighting (W) common component removal (R) GloVe vectors (GloVe + WR) by (Arora, Liang, and Ma 2017). For STS 16, we also compared our embedding with Skip-Thoughts (Kiros et al. 2015), BERT pretrained embedding average (Devlin et al. 2019) , Universal Sentence Encoder (Cer et al. 2018) and Sent2Vec (Pagliardini, Gupta, and Jaggi 2018) embeddings.

2. Semi-Supervised: We used avg-PSL, PSL + WR, and the avg-PSL the unweighted average of the PARAGRAM-SL999 (PSL) word vectors by (Wieting et al. 2015) as a baseline, obtained by training on PPDB dataset(Ganitkevitch, Van Durme, and Callison-Burch 2013). The word vectors are trained using unlabeled data. Furthermore, sentence embeddings are obtained from unweighted word vectors averaging. We also compared our method with the SIF weighting (W) common component removal (R) PSL word vectors (PSL + WR) by (Arora, Liang, and Ma 2017).

3. Supervised: We compared our method with PP, PP-proj., DAN, RNN, iRNN, LSTM (o.g), LSTM(no) and GRAN. All these methods are initialized with PSL word vectors and then trained on the PPDB dataset (Ganitkevitch, Van Durme, and Callison-Burch 2013). PP(Wieting et al. 2016) is the average of word vectors, while PP-proj is the average of word vectors followed by a linear projection. The word vectors are updated during the training. DAN denotes the deep averaging network of (Jiyyer et al. 2015). RNN is a Recurrent neural network, iRNN is the identity activated Recurrent Neural Network based on identity-initialized weight matrices. The LSTM is the version from (Gers, Schraudolph, and Schmidhuber 2002), either with output gates (denoted as LSTM (o.g)) or without (denoted as LSTM (no)). GRAN denotes state of the art supervised averaging based Gated Recurrent Averaging Network from (Wieting and Gimpel 2017). For STS 16 we also compared our embedding with Tree-LSTM (Tai, Socher, and Manning 2015) embedding.

Textual Classification Task:
We fix the document embeddings and only learn the classifier. We learn word vector embeddings using Skip-Gram with a window size of 10, Negative Sampling (SGNS) of 10, and minimum word frequency of 20. We use 5-fold cross-validation on the $F_1$ score to tune hyperparameters. We use LinearSVM for multi-class classification and Logistic regression with the OneVsRest setting for multi-label classification. We fix the number of dictionary elements to either 40 or 20 (with Doc2vec initialize word vectors) and non-zero coefficient to $k = K/2$ during dictionary learning for all experiments. We use the best parameter settings, as reported in all our baselines to generate their results. We use 200 dimensions for tf-idf weighted word-vector model, 400 for paragraph vector model, 80 topics and 400 dimensional vectors for TWE, NTSG, LTSG and 60 topics and 200 dimensional word vectors for SCDV (Mekala et al. 2017).

Baseline Details: We considered the following baselines: The Bag-of-Words (BoW) model (Harris 1954), the Bag of Word Vector (BoVV) (Gupta et al. 2016) model, Sparse Composite Document Vector (SCDV) (Mekala et al. 2017) 19 paragraph vector models (Le and Mikolov 2014), Topical word embeddings (TWE-1) (Liu et al. 2015), Neural Tensor Skip-Gram Model (NTSG-1 to NTSG-3) (Liu, Qiu, and Huang 2015), tf-idf weighted average word-vector model (Singh and Mukerjee 2015) and weighted Bag of Concepts (weight-BoC) (Kim, Kim, and Cho 2017) where we built document-topic vectors by counting the member words in each topic, and Doc2Vec (Chen 2017) where averaging and training of word vectors are done jointly. Moreover, we used SIF (Arora, Liang, and Ma 2017) smooth inverse frequency weight with common component removal from weighted average vectors as a baseline. We also compared our results with other topic modeling based document embedding methods such as WTM (Fu et al. 2016), w2v-LDA (Nguyen et al. 2015), LDA (Chen and Liu 2014), TV+MeanWV (Li et al. 2016a)), LTSG (Law et al. 2017), Gaussian-LDA (Das, Zaheer, and Dyer 2015), Topic2Vec (Niu et al. 2015), Lda2Vec (Moody 2016), MvTM (Li et al. 2016b) and BERT (Devlin et al. 2019). For BERT, we reported the results on the unsupervised pre-trained (pr) model because of a fair comparison to our approach which is also unsupervised.

Class wise Performance on 20NewsGroup
We also reported the precision, recall, and micro-F1 results of separate 20 classes of the 20NewsGroup dataset. We compared our embedding (P-SIF) with Bag of Words, and SCDV embeddings. In Table 9, P-SIF (Doc2VecC) (20 partitions) embeddings outperforms SCDV (60 partitions) on 18 out of the 20 classes.

Other Supervised Tasks
We also considered three out of domain supervised tasks: the SICK similarity task, the SICK entailment task, and the Stanford Sentiment Treebank (SST) binary classification task by (Socher et al. 2013).
Table 6: The STS tasks by year. Tasks with the same name in different years are different tasks

| STS12 | STS13 | STS14 | STS15 | STS16 |
|-------|-------|-------|-------|-------|
| MSRpar | headline | deft forum | answers-forums | headlines |
| MSRvid | OnWN | deft news | answers-students | plagiarism |
| SMT-eur | FNWN | headline | belief | postediting |
| OnWN | SMT | images | headline | answer-answer |
| SMT-news | OnWN | images | question-question | |

Table 7: Experimental results (Pearson’s r × 100) on textual similarity tasks. The highest score in each row is in bold. The methods can be supervised (denoted as Su.), semi-supervised (Se.), or unsupervised (Un.). See the main text for description of the methods. Many results are collected from (Wieting et al. 2016) and (Wieting and Gimpel 2017) (GRAN) except the tfidf-GloVe and our representation.

Table 8: Experimental results (Pearson’s r × 100) on textual similarity tasks on STS 16. The highest score in each row is in bold.
Table 9: Class performance on the 20newsgroup dataset. P-SIF represents our embedding with 40 partitions. P-SIF (Doc2VecC) represents our embeddings initialized with Doc2VecC trained word-vectors with 20 partitions.

| Class Name                        | BoW   | SCDV  | P-SIF | P-SIF (Doc2VecC) |
|-----------------------------------|-------|-------|-------|------------------|
|                                   | Pre.  | Rec.  | F-mes | Pre.  | Rec.  | F-mes | Pre.  | Rec.  | F-mes | Pre.  | Rec.  | F-mes |
| all.athens                        | 67.8  | 72.1  | 69.8  | 80.2  | 79.5  | 79.8  | 83.3  | 80.2  | 81.7  | 83    | 79.9  | 81.4  |
| comp.graphics                     | 67.1  | 73.5  | 70.1  | 75.3  | 77.4  | 76.3  | 76.6  | 78.1  | 77.3  | 76.8  | 79.2  | 77.9  |
| comp.os.ms-windows.misc           | 71.1  | 66.5  | 71.4  | 78.6  | 77.2  | 77   | 76.3  | 77.7  | 76.9  | 77.2  | 78.2  | 77.7  |
| comp.sys.ibm.pc.hardware          | 62.8  | 72.4  | 67.2  | 75.6  | 73.5  | 74.2  | 73.4  | 74.5  | 73.9  | 71.1  | 74.2  | 72.6  |
| comp.sys.mac.hardware             | 77.4  | 78.2  | 77.8  | 83.4  | 85.5  | 84.4  | 87.1  | 84.4  | 85.7  | 87.5  | 87.5  | 87.5  |
| comp.windows.x                    | 83.2  | 73.2  | 77.8  | 87.6  | 78.6  | 82.8  | 89.3  | 78   | 83.2  | 88.8  | 78.5  | 83.3  |
| misc.forsale                      | 81.3  | 88.2  | 84.6  | 81.4  | 85.9  | 83.5  | 82.7  | 88   | 85.2  | 82.4  | 86.4  | 84.3  |
| rec.autos                         | 80.7  | 82.8  | 81.7  | 91.2  | 90.6  | 90.9  | 93    | 90.1  | 91.5  | 92.8  | 90.7  | 91.7  |
| rec.motorcycles                   | 92.3  | 87.9  | 90.0  | 95.4  | 95.7  | 95.5  | 93.6  | 95.5  | 94.5  | 97    | 96.5  | 96.7  |
| rec.sport.baseball                | 89.8  | 89.2  | 89.5  | 93.2  | 94.7  | 93.9  | 93.3  | 95.2  | 94.2  | 95.2  | 95.7  | 95.4  |
| rec.sport.hockey                  | 93.3  | 93.7  | 93.5  | 96.3  | 99.2  | 97.7  | 95.6  | 98.5  | 97   | 96.8  | 98.8  | 97.7  |
| sci.crypt                         | 92.2  | 86.1  | 89.0  | 92.5  | 94.7  | 93.5  | 89.8  | 93.2  | 91.4  | 93.4  | 96.7  | 95.0  |
| sci.electronics                   | 70.9  | 73.3  | 72.8  | 74.6  | 74.9  | 74.7  | 79.6  | 78.6  | 79.1  | 78  | 79.3  | 78.6  |
| sci.med                           | 79.3  | 81.3  | 80.2  | 91.3  | 88.4  | 89.8  | 91.9  | 88.6  | 90.2  | 92.7  | 89.9  | 91.2  |
| sci.space                         | 90.2  | 88.3  | 89.2  | 88.5  | 93.8  | 91.07 | 89.4  | 94   | 91.6  | 90.7  | 94.4  | 92.5  |
| soc.religion.christian            | 77.3  | 87.9  | 82.2  | 83.3  | 92.3  | 87.5  | 84    | 94.3  | 88.8  | 86  | 92.5  | 89.1  |
| talk.politics.guns                | 71.7  | 85.7  | 78.0  | 72.7  | 90.6  | 80.6  | 73.1  | 91.2  | 81.1  | 77.3  | 89.8  | 83.1  |
| talk.politics.mideast             | 91.7  | 76.9  | 83.6  | 96.2  | 95.4  | 95.8  | 97    | 94.5  | 95.7  | 97.5  | 94.2  | 95.8  |
| talk.politics.misc                | 71.7  | 56.5  | 63.2  | 80.9  | 59.7  | 68.7  | 81    | 59  | 68.2  | 82    | 62  | 70.6  |
| talk.religion.misc                | 63.2  | 55.4  | 59.04 | 73.5  | 57.2  | 64.3  | 72.2  | 59  | 64.9  | 67.4  | 62.4  | 64.8  |

Table 10: Results on similarity, entailment, and sentiment tasks. The row for similarity (SICK) shows Pearson’s r x 100 and the last two rows show accuracy. The highest score in each row is in bold. Results in Column 2 to 6 are collected from (Wieting et al. 2016), and those in Column 7 for skip-thought are from (Kiros et al. 2015), Column 8 for PSL + WR are from (Arora, Liang, and Ma 2017).

| Tasks            | PP      | DAN     | RNN     | LSTM (no) | LSTM (o.g.) | skip thought | PSL +WR | P-SIF +PSL |
|------------------|---------|---------|---------|-----------|-------------|--------------|--------|------------|
| similarity (SICK)| 84.9    | 85.96   | 73.13   | 85.45     | 83.4        | 85.8         | 86.3   | 87.6       |
| entailment (SICK)| 83.1    | 84.5    | 76.4    | 83.2      | 82.0        | -            | 84.6   | 85.5       |
| sentiment (SST)  | 79.4    | 83.4    | 86.5    | 86.6      | 89.2        | -            | 82.2   | 86.4       |

**Results and Analysis.** Our method (P-SIF) obtains a better performance compared to PSL + WR on all the three tasks similarity, entailment, and sentiment. We obtained the best results for two of the supervised tasks, although many of these methods (DAN, RNN, LSTM) are trained with supervision. Furthermore, the skip thought vectors use a higher dimension of 2400 instead of 300 dimensions (which we projected to 2400 for a fair comparison). Our method wasn’t able to outperform the sentiment task compared to supervised tasks because a) due to the antonym problem word-vectors capture the semantic meaning of words and b) in our weighted average scheme, we didn’t assign more weights to sentiment words such as ‘not’, ‘good’, ‘bad’, there may be some important sentiment words which are down-weighted by the SIF weighting scheme. However, we outperform PSL + WR by a significant margin and have a less performance gap with the best supervised approach.

**Proof: Kernels meet Embeddings**

1. $K^1(D_A, D_B)$ represents document similarity between the documents represented by average word vectors $d_x = \sum_i \vec{v}_i$

   **Proof:**

   \[
   d^A = \frac{1}{n} \sum_{i=1}^{n} \vec{v}_{w_i}^A \\
   d^B = \frac{1}{m} \sum_{j=1}^{m} \vec{v}_{w_j}^B \\
   K^1(D_A, D_B) = \langle d^A, d^B \rangle
   \]

   By substituting values from 10 and 11 into 12, we will get

   \[
   K^1(D_A, D_B) = \frac{1}{nm} \sum_{i=1}^{n} \vec{v}_{w_i}^A \cdot \sum_{j=1}^{m} \vec{v}_{w_j}^B
   \]

   \[
   K^1(D_A, D_B) = \frac{1}{nm} \sum_{i=1}^{n} \vec{v}_{w_i}^A \cdot \sum_{j=1}^{m} \vec{v}_{w_j}^B
   \]
Using the distributive property of dot product

\[
\sum_{i=1}^{n} (\sum_{j=1}^{m} \vec{v}_{w_i} \cdot \vec{v}_{w_j}) = \sum_{i=1}^{n} \sum_{j=1}^{m} (\vec{v}_{w_i} \cdot \vec{v}_{w_j})
\]  

(15)

By substituting 15 in 14, we will get

\[
K^2(D_A, D_B) = \frac{1}{nm} \sum_{i=1}^{n} \sum_{j=1}^{m} (\vec{v}_{w_i} \cdot \vec{v}_{w_j})
\]

(16)

2. \(K^2(D_A, D_B)\) represents the document similarity between the documents represented by topical word vectors (Liu, Qiu, and Huang 2015)

**Proof:**

\[
dv^A = \frac{1}{n} \sum_{i=1}^{n} \vec{v}_{w_i} = \sum_{i=1}^{n} \alpha_{w_i,} \vec{v}_{w_i}^A
\]

(17)

\[
dv^B = \frac{1}{m} \sum_{j=1}^{m} \vec{v}_{w_j} = \sum_{j=1}^{m} \alpha_{w_j,} \vec{v}_{w_j}^B
\]

(18)

\[
K^2(D_A, D_B) = \langle dv_A, dv_B \rangle
\]

(19)

By substituting values from 17 and 18 to 19, we will get

\[
K^2(D_A, D_B) = \frac{1}{nm} \sum_{i=1}^{n} \sum_{j=1}^{m} \langle \vec{v}_{w_i}^A, \vec{v}_{w_j}^B \rangle
\]

(20)

Using the distributive property of dot product

\[
K^2(D_A, D_B) = \frac{1}{nm} \sum_{i=1}^{n} \sum_{j=1}^{m} \langle \vec{v}_{w_i}^A, \vec{v}_{w_j}^B \rangle
\]

(21)

\[
K^2(D_A, D_B) = \frac{1}{nm} \sum_{i=1}^{n} \sum_{j=1}^{m} \langle \alpha_{w_i,} \vec{v}_{w_i}^A, \alpha_{w_j,} \vec{v}_{w_j}^B \rangle
\]

(22)

Using the distributive and scalar multiplication property of dot product

\[
\langle \alpha_{w_i,} \vec{v}_{w_i}^A, \alpha_{w_j,} \vec{v}_{w_j}^B \rangle = \langle \alpha_{w_i,} \cdot \alpha_{w_j,} \vec{v}_{w_i}^A, \vec{v}_{w_j}^B \rangle
\]

(23)

Since, \(t_{w_i} = \alpha_{w_i,}\) and \(t_{w_j} = \alpha_{w_j,}\)

\[
\langle \alpha_{w_i,} \vec{v}_{w_i}^A, \alpha_{w_j,} \vec{v}_{w_j}^B \rangle = \langle t_{w_i}, t_{w_j} \rangle \cdot \langle \vec{v}_{w_i}^A, \vec{v}_{w_j}^B \rangle
\]

(24)

By substituting 24 in 22, we will get

\[
K^2(D_A, D_B) = \frac{1}{nm} \sum_{i=1}^{n} \sum_{j=1}^{m} \langle \vec{v}_{w_i}^A, \vec{v}_{w_j}^B \rangle + \langle t_{w_i}, t_{w_j} \rangle \cdot \langle \vec{v}_{w_i}^A, \vec{v}_{w_j}^B \rangle
\]

(25)

3. \(K^3(D_A, D_B)\) represents the document similarity between the documents represented by partition average word vectors (P-SIF)

**Proof:**

\[
dv^A = \frac{1}{n} \sum_{i=1}^{n} \vec{v}_{w_i} = \sum_{i=1}^{n} \sum_{k=1}^{K} \alpha_{w_i,k} \vec{v}_{w_i}^A
\]

(26)

\[
dv^B = \frac{1}{m} \sum_{j=1}^{m} \vec{v}_{w_j} = \sum_{j=1}^{m} \sum_{k=1}^{K} \alpha_{w_j,k} \vec{v}_{w_j}^B
\]

(27)

\[
K^3(D_A, D_B) = \langle dv^A, dv^B \rangle
\]

(28)

By substituting values from 26 and 27 in 28, we will get

\[
K^3(D_A, D_B) = \frac{1}{nm} \sum_{i=1}^{n} \sum_{j=1}^{m} \langle \vec{v}_{w_i}^A, \vec{v}_{w_j}^B \rangle \cdot \langle \vec{v}_{w_i}^A, \vec{v}_{w_j}^B \rangle
\]

(29)

Using the distributive property of dot product

\[
K^3(D_A, D_B) = \frac{1}{nm} \sum_{i=1}^{n} \sum_{j=1}^{m} \langle \vec{v}_{w_i}^A, \vec{v}_{w_j}^B \rangle \cdot \langle \vec{v}_{w_i}^A, \vec{v}_{w_j}^B \rangle
\]

(30)

Using distributive and scalar multiplication property of dot product

\[
\langle \sum_{k=1}^{K} \alpha_{w_i,k} \vec{v}_{w_i}^A, \sum_{k=1}^{K} \alpha_{w_j,k} \vec{v}_{w_j}^B \rangle = \langle \vec{v}_{w_i}^A, \vec{v}_{w_j}^B \rangle \times \langle \vec{v}_{w_i}^A, \vec{v}_{w_j}^B \rangle
\]

(31)

Since, \(t_{w_i,k} = \alpha_{w_i,k}\) and \(t_{w_j,k} = \alpha_{w_j,k}\)

\[
\langle \sum_{k=1}^{K} \alpha_{w_i,k} \vec{v}_{w_i}^A, \sum_{k=1}^{K} \alpha_{w_j,k} \vec{v}_{w_j}^B \rangle = \langle \sum_{k=1}^{K} t_{w_i,k} \cdot t_{w_j,k} \rangle \times \langle \vec{v}_{w_i}^A, \vec{v}_{w_j}^B \rangle
\]

(32)

From the definition of the dot product of vectors,

\[
\langle \sum_{k=1}^{K} \alpha_{w_i,k} \vec{v}_{w_i}^A, \sum_{k=1}^{K} \alpha_{w_j,k} \vec{v}_{w_j}^B \rangle = \langle \vec{v}_{w_i}^A, \vec{v}_{w_j}^B \rangle \times \langle \vec{v}_{w_i}^A, \vec{v}_{w_j}^B \rangle
\]

(35)

By substituting 35 in 31, we will get

\[
K^3(D_A, D_B) = \frac{1}{nm} \sum_{i=1}^{n} \sum_{j=1}^{m} \langle \vec{v}_{w_i}^A, \vec{v}_{w_j}^B \rangle \times \langle \vec{v}_{w_i}^A, \vec{v}_{w_j}^B \rangle
\]

(36)
4. \(K^4(D_A, D_B)\) represents the document similarity between the documents represented by the relaxed word mover distance (Kusner et al. 2015) when words of \(D_A\) are matched to \(D_B\).

**Proof:** From the definition of the relaxed word mover distance in (Kusner et al. 2015). Relaxed word mover maps each word \(w_i^A\) of document \(D_A\) to the closest word \(w_j^B\) of document \(D_B\).

Since, word \(w_j^B\) of document \(D_B\) is closer to word \(w_i^A\) of document \(D_A\), we will have

\[w_j^B = \arg \max_{w_j^B} (\vec{v}_{w_i^A} \cdot \vec{v}_{w_j^B}) \] (37)

Therefore, similarity contribution \(K_{(i,j)}\) from word \(w_i^A\) of document \(D_A\) is given by:

\[K_{(i,j)} = \frac{1}{n} \max_{w_j^B} (\vec{v}_{w_i^A} \cdot \vec{v}_{w_j^B}) \] (38)

Total similarity contribution from all the \(n\) words of the document \(D_A\):

\[K^4(D_A, D_B) = \frac{1}{n} \sum_{i=1}^{n} K_{(i,j)} = \frac{1}{n} \sum_{i=1}^{n} \max_{w_j^B} (\vec{v}_{w_i^A} \cdot \vec{v}_{w_j^B}) \] (39)

We can write \(\max_{w_j^B}\) as \(\max_j\), thus finally

\[K^4(D_A, D_B) = \frac{1}{n} \sum_{i=1}^{n} \max_j (\vec{v}_{w_i^A} \cdot \vec{v}_{w_j^B}) \] (40)

**Qualitative Example: Document Similarity**

Let’s consider a corpus \((C)\) with \(N\) documents with the corresponding most frequent vocabulary \((V)\). Figure 3 represents the word-vectors \(V\), where similar meaning words are closer. We can apply sparse coding and partition the word-vectors space into five (total topics \(K = 5\)) topic vector spaces. Some words are polysemic and belong to multiple topics with some proportion, as shown in Figure 3. For example, words such as baby, person, dog and kangaroo, belong to multiple topics with a significant proportion. Words and corresponding vectors in these topic vector spaces are represented by topic numbers in the subscript. Table 11 shows an example pair from the STS Task 2012 MSRvid dataset and the corresponding SIF (averaging) and P-SIF (partition averaging) representation vectors. We can see that in the SIF representation, we are averaging words vectors which semantically have different meanings. The document is represented in the same \(d\) dimensional word-vectors space. Overall, SIF represents the document as a single point in the vector space and does not take account of different semantic meanings of the topics. Whereas, in the P-SIF representation, we treat the five different semantic topics distinctly. Words belonging to different semantic topics are separated by concatenation \(\odot\) as they represent different meanings, whereas words coming from the same topic are averaged as they represent the same meaning. The final document vector \(\vec{d}_d\) has more representational power as it is represented in a higher \(5 \times d\) dimensional vector space. Thus, partitioned averaging with topic weighting is important for representing documents. Empirically, P-SIF assigned a lower score of 0.16 (rescaled to a 0-1 scale) for sentences \((d_1, d_2)\) where the ground truth is 0.15 (rescaled to a 0-1 scale), whereas SIF gave similarity score of 0.57 (0-1 scale), farther than the ground score. Thus, we obtain a relative improvement of 98% in the error difference from the ground truth. Here, the simple averaging-based embedding of \(d_1^A\) and \(d_2^B\), brings the document representations closer. But partitioned based averaging, P-SIF, projects the documents farther in a higher-dimensional space.

**Figure 3: Words in different topics are represented by different subscripts and separated by hyperplanes. Bold represents words from example documents.**

**Qualitative Results: Similarity task**

Table 13 represents successful example pair from STS 2012 MSRvid dataset where P-SIF assigns similarity scores closer to ground truth than SIF. Table 12 represents the failed example pairs from STS 2012 MSRvid dataset where SIF assigns a similarity score closer to the ground truth than P-SIF. We now introduce the header notations used in the Table 13 and 12 in details.

- **GT:** represents the given ground truth similarity score in a range of 0-5.
- **NGT:** represents the normalized ground truth similarity score. NGT is obtained by dividing the GT score by 5 so that it is in a range of 0-1.
- **SIF:** represents the SIF embedding similarity score in a range of 0-1.
- **P-SIF:** represents the P-SIF embedding similarity score in a range of 0-5.
- **SIFerr:** represents absolute error \(\|SIF_{ac} - NGT\|\) between normalized ground truth similarity score and the SIF embedding similarity score.
- **P-SIFerr:** represents the absolute error \(\|P-SIF_{ac} - NGT\|\) between the ground truth similarity score and the P-SIF embedding similarity score.
- **Differr:** represents absolute difference between SIFerr and P-SIFerr. Examples where P-SIF performs better Differr = P-SIFerr - SIFerr (used in Table 13). Examples where SIF performs better Differr = SIFerr - P-SIFerr (used in Table 12)
- **Relerr:** represents relative difference between SIFerr and P-SIFerr. Examples where P-SIF performs better Relerr = \(\frac{Differr}{SIFerr}\) (used in Table 13). Examples where SIF performs better Relerr = \(\frac{Differr}{P-SIFerr}\) (used in Table 12)
Table 11: STS Task 2012 MSRVid dataset similarity example pair. Here, P-SIF assigns a score of 0.16 (rescaled to a 0-1 scale) to sentences \((d_1^a, d_2^a)\), where the ground truth of 0.15 (0-1 scale), whereas SIF assigns a similarity score of 0.57 (rescaled to a 0-1 scale). Thus, we obtain a relative improvement of 98% in the error difference. Here, \(\oplus\) represents concatenation. \(\vec{v}_{zero}\) is the zero padding vector.

| Doc | SIF  | P-SIF  | Score |
|-----|------|--------|-------|
| A man is riding a motorcycle | \(\vec{v}_{man1} \oplus \vec{v}_{riding1} \oplus \vec{v}_{motorcycle1} \oplus \vec{v}_{zero}\) | \(\vec{v}_{man1} \oplus \vec{v}_{riding1} \oplus \vec{v}_{horse1}\) | 0.15 |

Table 12: STS 2012 MSRVid examples where the P-SIF score were far away from the ground truth, whereas the SIF scores were closer to the actual ground truth.

| sentence1 | sentence2 | GT | NGT | SIFsc | SIFerr | Differr | Relerr |
|-----------|-----------|----|-----|-------|--------|---------|--------|
| A man is riding a motorcycle | A woman is riding a horse | 0.5 | 0.1 | 0.1971 | 0.3944 | 0.0971 | 0.2944 | 0.1973 | 0.6703 |

Table 13: STS 2012 MSRVid example where the P-SIF scores were closer to the ground truth, whereas SIF scores were more away from the ground truth.

| sentence1 | sentence2 | GT | NGT | SIFsc | SIFerr | P-SIFsc | P-SIFerr | Differr | Relerr |
|-----------|-----------|----|-----|-------|--------|---------|---------|---------|--------|