Word Mover's Embedding: From Word2Vec to Document Embedding

Lingfei Wu
IBM Research
wuli@us.ibm.com

Ian En-Hsu Yen
Carnegie Mellon University
eyan@cs.cmu.edu

Kun Xu
IBM Research
kun.xu1@ibm.com

Fangli Xu
College of William and Mary
fxu02@email.wm.edu

Avinash Balakrishnan
IBM Research
avinash.bala@us.ibm.com

Pin-Yu Chen
IBM Research
Pin-yu.chen@ibm.com

Pradeep Ravikumar
Carnegie Mellon University
pradeepr@cs.cmu.edu

Michael J. Witbrock
IBM Research
witbrock@us.ibm.com

Abstract

While the celebrated Word2Vec technique yields semantically rich representations for individual words, there has been relatively less success in extending to generate unsupervised sentences or documents embeddings. Recent work has demonstrated that a distance measure between documents called *Word Mover's Distance* (WMD) that aligns semantically similar words, yields unprecedented KNN classification accuracy. However, WMD is expensive to compute, and it is hard to extend its use beyond a KNN classifier. In this paper, we propose the *Word Mover's Embedding* (WME), a novel approach to building an unsupervised document (sentence) embedding from pre-trained word embeddings. In our experiments on 9 benchmark text classification datasets and 22 textual similarity tasks, the proposed technique consistently matches or outperforms state-of-the-art techniques, with significantly higher accuracy on problems of short length.

1 Introduction

Text representation plays an important role in many NLP-based tasks such as document classification and clustering (Zhang et al., 2018; Gui et al., 2016, 2014), sense disambiguation (Gong et al., 2017, 2018a), machine translation (Mikolov et al., 2013b), document matching (Pham et al., 2015), and sequential alignment (Peng et al., 2016, 2015). Since there are no explicit features in text, much work has aimed to develop effective text representations. Among them, the simplest bag of words (BOW) approach (Salton and Buckley, 1988) and its term frequency variants (e.g. TF-IDF) (Robertson and Walker, 1994) are most widely used due to simplicity, efficiency and often surprisingly high accuracy (Wang and Manning, 2012). However, simply treating words and phrases as discrete symbols fails to take into account word order and the semantics of the words, and suffers from frequent near-orthogonality due to its high dimensional sparse representation. To overcome these limitations, Latent Semantic Indexing (Deerwester et al., 1990) and Latent Dirichlet Allocation (Blei et al., 2003) were developed to extract more meaningful representations through singular value decomposition (Wu and Stathopoulos, 2015) and learning a probabilistic BOW representation.

A recent empirically successful body of research makes use of distributional or contextual information together with simple neural-network models to obtain vector-space representations of words and phrases (Bengio et al., 2003; Mikolov et al., 2013a,c; Pennington et al., 2014). A number of researchers have proposed extensions of these towards learning semantic vector-space representations of sentences or documents. A simple but often effective approach is to use a weighted average over some or all of the embeddings of words in the document. While this is simple, important information could easily be lost in such a document representation, in part since it does not consider word order. A more sophisticated approach (Le and Mikolov, 2014; Chen, 2017) has focused on jointly learning embeddings for both words and paragraphs using models similar to Word2Vec. However, these only use word order within a small context window; moreover, the quality of word embeddings learned in such a model may be limited by the size of the training corpus, which cannot scale to the large sizes used in the simpler word embedding models, and which may consequently weaken the quality of the document embeddings.

Recently, Kusner et al. (Kusner et al., 2015) presented a novel document distance metric, Word Mover's Distance (WMD), that measures the dissimilarity between two text documents in the Word2Vec embedding space. Despite its state-of-the-art KNN-based classification accuracy over other methods, combining KNN and WMD incurs very high computational cost. More importantly, WMD is simply a distance that can be only combined with KNN or K-means, whereas many machine learning algorithms require a fixed-length feature representation as input.

A recent work in building kernels from distance measures, D2KE (distances to kernels and embeddings) (Wu et al., 2018a) proposes a general methodology of the derivation of a positive-definite kernel from a given distance function, which enjoys better theoretical guarantees than other distance-based methods, such as k-nearest neighbor and distance substitution kernel (Haasdonk and Bahlmann, 2004), and has also been demonstrated to have strong empirical performance in the time-series domain (Wu et al., 2018b).

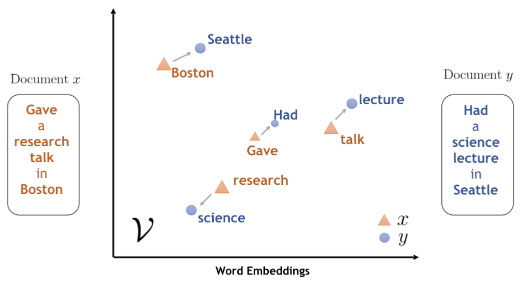
In this paper, we build on this recent innovation D2KE (Wu et al., 2018a), and present the *Word Mover's Embedding* (WME), an unsupervised generic framework that learns continuous vector representations for text of variable lengths such as a sentence, paragraph, or document. In particular, we propose a new approach to first construct a positive-definite *Word Mover's Kernel* via an infinite-dimensional feature map given by the Word Mover's distance (WMD) to random documents from a given distribution. Due to its use of the WMD, the feature map takes into account alignments of individual words between the documents in the semantic space given by the pre-trained word embeddings. Based on this kernel, we can then derive a document embedding via a Random Features approximation of the kernel, whose inner products approximate exact kernel computations. Our technique extends the theory of *Random Features* to show convergence of the inner product between WMEs to a positive-definite kernel that can be interpreted as a soft version of (inverse) WMD.

The proposed embedding is more efficient and flexible than WMD in many situations. As an example, WME with a simple linear classifier reduces the computational cost of WMD-based KNN *from cubic to linear* in document length and *from quadratic to linear* in number of samples, while simultaneously improving accuracy. WME is extremely easy to implement, fully parallelizable, and highly extensible, since its two building blocks, Word2Vec and WMD, can be replaced by other techniques such as GloVe (Pennington et al., 2014; Wieting et al., 2015b) or S-WMD (Huang et al., 2016). We evaluate WME on 9 real-world text classification tasks and 22 textual similarity tasks, and demonstrate that it consistently matches or outperforms other state-of-the-art techniques. Moreover, WME often achieves orders of magnitude speed-up compared to KNN-WMD while obtaining the same testing accuracy. Our code and data is available at `https://github.com/IBM/WordMoversEmbeddings`.

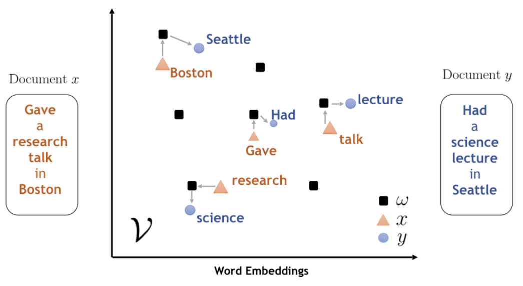
2 Word2Vec and Word Mover's Distance

We briefly introduce Word2Vec and WMD, which are the key building blocks of our proposed method. Here are some notations we will use throughout the paper. Given a total number of documents N with a vocabulary $\mathcal{W}$ of size $|\mathcal{W}| = n$, the Word2vec embedding gives us a d-dimensional vector space $\mathcal{V} \subseteq \mathbb{R}^d$ such that any word in the vocabulary set $w \in \mathcal{W}$ is associated with a semantically rich vector representation $\boldsymbol{v}_w \in \mathbb{R}^d$. Then in this work, we consider each document as a collection of word vectors $x := (\boldsymbol{v}_j)_{j=1}^L$ and denote $\mathcal{X} := \bigcup_{L=1}^{L_{\max}} \mathcal{V}^L$ as the space of documents.

Word2Vec. In the celebrated Word2Vec approach (Mikolov et al., 2013a,c), two shallow yet effective models are used to learn vector-space representations of words (and phrases), by mapping those that co-occur frequently, and consequently with plausibly similar meaning, to nearby vectors in the embedding vector space. Due to the model's simplicity and scalability, high-quality word embeddings can be generated to capture a large number of precise syntactic and semantic word relationships by training over hundreds of billions of words and millions of named entities. The advantage of document representations building on top of word-level embeddings is that one can make full use of high-quality pre-trained word embeddings. Throughout this paper we use Word2Vec as our first building block but other (unsupervised or supervised) word



(a) WMD



(b) WME

Figure 1: An illustration of the WMD and WME. All non-stop words are marked as bold face. WMD measures the distance between two documents. WME approximates a kernel derived from WMD with a set of random documents.

embeddings (Pennington et al., 2014; Wieting et al., 2015b) could also be utilized.

Word Mover's Distance. Word Mover's Distance was introduced by (Kusner et al., 2015) as a special case of the Earth Mover's Distance (Rubner et al., 2000), which can be computed as a solution of the well-known transportation problem (Hitchcock, 1941; Altschuler et al., 2017). WMD is a distance between two text documents $x, y \in \mathcal{X}$ that takes into account the alignments between words. Let $|x|$, $|y|$ be the number of distinct words in x and y. Let $\boldsymbol{f}_x \in \mathbb{R}^{|x|}$, $\boldsymbol{f}_y \in \mathbb{R}^{|y|}$ denote the normalized frequency vectors of each word in the documents x and y respectively (so that $\boldsymbol{f}_x^\top \mathbf{1} = \boldsymbol{f}_y^\top \mathbf{1} = 1$). Then the WMD distance between documents x and y is defined as:

$$\begin{aligned} \mathrm{WMD}(x, y) &:= \min_{F \in \mathbb{R}_+^{|x| \times |y|}} \langle C, F \rangle, \\ s.t., F\mathbf{1} &= \boldsymbol{f}_x, \quad F^\top \mathbf{1} = \boldsymbol{f}_y. \end{aligned} \tag{1}$$

where F is the transportation flow matrix with F_{ij} denoting the amount of flow traveling from i-th word x_i in x to j-th word y_j in y, and C is the transportation cost with $C_{ij} := \mathrm{dist}(\boldsymbol{v}_{x_i}, \boldsymbol{v}_{y_j})$ being the distance between two words measured in the Word2Vec embedding space. A popular choice is the Euclidean distance $\mathrm{dist}(\boldsymbol{v}_{x_i}, \boldsymbol{v}_{y_j}) = \|\boldsymbol{v}_{x_i} - \boldsymbol{v}_{y_j}\|_2$. When $\mathrm{dist}(\boldsymbol{v}_{x_i}, \boldsymbol{v}_{y_j})$ is a *metric*, the WMD distance in Eq. (1) also qualifies as a metric, and in particular, satisfies the triangle inequality (Rubner et al., 2000). Building on top of Word2Vec, WMD is a particularly useful and accurate for measure of the distance between documents with semantically close but syntactically different words as illustrated in Figure 1(a).

The WMD distance when coupled with KNN has been observed to have strong performance in classification tasks (Kusner et al., 2015). However, WMD is expensive to compute with computational complexity of $O(L^3 \log(L))$, especially for long documents where L is large. Additionally, since WMD is just a document distance, rather than a document representation, using it within KNN incurs even higher computational costs $O(N^2 L^3 \log(L))$.

3 Document Embedding via Word Mover's Kernel

In this section, we extend the framework in (Wu et al., 2018a), to derive a positive-definite kernel from an alignment-aware document distance metric, which then gives us an unsupervised semantic embeddings of texts of variable length as a by-product through the theory of *Random Feature Approximation* (Rahimi and Recht, 2007).

3.1 Word Mover's Kernel

We start by defining the *Word Mover's Kernel*:

$$\begin{aligned} k(x, y) &:= \int p(\omega) \phi_\omega(x) \phi_\omega(y) d\omega, \\ \text{where} \ \ \phi_\omega(x) &:= \exp(-\gamma \mathrm{WMD}(x, \omega)). \end{aligned} \tag{2}$$

where ω can be interpreted as a random document $\{\boldsymbol{v}_j\}_{j=1}^{D}$ that contains a collection of random word vectors in $\mathcal{V}$, and $p(\omega)$ is a distribution over the space of all possible random documents $\Omega := \bigcup_{D=1}^{D_{max}} \mathcal{V}^D$. $\phi_\omega(x)$ is an possibly infinite-dimensional feature map derived from the WMD between x and all possible documents $\omega \in \Omega$.

An insightful interpretation of this kernel (2):

$$k(x, y) := \exp(-\gamma \, softmin_{p(\omega)} f(\omega))$$

where

$$softmin_{p(\omega)}\, f(\omega) := -\frac{1}{\gamma}\log\int p(\omega)e^{-\gamma f(\omega)}d\omega,$$

and $f(\omega) = \{\text{WMD}(x,\omega) + \text{WMD}(\omega,y)\}$, is a version of soft minimum function parameterized by $p(\omega)$ and γ. Comparing this with the usual definition of soft minimum $softmin_i f_i := -softmax(-f_i) = -\log\sum_i e^{-f_i}$, it can be seen that the soft-min-variant in the above Equations uses a weighting of the objects ω via the probability density $p(\omega)$, and moreover has the additional parameter γ to control the degree of smoothness. When γ is large and $f(\omega)$ is Lipschitz-continuous, the value of the soft-min-variant is mostly determined by the minimum of $f(\omega)$.

Note that since WMD is a metric, by the triangular inequality we have

$$\text{WMD}(x,y) \leq \min_{\omega\in\Omega}\, (\text{WMD}(x,\omega) + \text{WMD}(\omega,y))$$

and the equality holds if we allow the length of random document $D_{\max}$ to be not smaller than L. Therefore, the kernel (2) serves as a good approximation to the WMD between any pair of documents x, y as illustrated in Figure 1(b), while it is *positive-definite* by the definition.

3.2 Word Mover's Embedding

Given the Word-Mover's Kernel in Eq. (2), we can then use the Monte-Carlo approximation:

$$k(x,y) \approx \langle Z(x), Z(y)\rangle = \frac{1}{R}\sum_{i=1}^{R}\phi_{\omega_i}(x)\phi_{\omega_i}(y) \tag{3}$$

where $\{\omega_i\}_{i=1}^{R}$ are i.i.d. random documents drawn from $p(\omega)$ and $Z(x) := (\frac{1}{\sqrt{R}}\phi_{\omega_i}(x))_{i=1}^{R}$ gives a vector representation of document x. We call this random approximation *Word Mover's Embedding*. Later, we show that this Random Features approximation in Eq. (3) converges to the exact kernel (2) uniformly over all pairs of documents (x, y) .

Distribution $p(\omega)$**.** A key ingredient in the Word Mover's Kernel and Embedding is the distribution $p(\omega)$ over random documents. Note that $\omega \in \mathcal{X}$ consists of sets of words, each of which lies in the Word2Vec embedding space; the characteristics of which need to be captured by $p(\omega)$ in order to generate (sets of) "meaningful" random words. Several studies have found that the word vectors $\boldsymbol{v}$ are roughly uniformly dispersed in the word embedding space (Arora et al., 2016, 2017). This is also consistent with our empirical findings, that the uniform distribution centered by the mean of all word vectors in the documents is generally applicable for various text corpora. Thus, if d is the dimensionality of the pre-trained word embedding space, we can draw a random word $u \in \mathbb{R}^d$ as $u_j \sim \text{Uniform}[v_{\min}, v_{\max}]$, for $j = 1, \ldots, d$, and where $v_{\min}$ and $v_{\max}$ are some constants.

Given a distribution over random words, the remaining ingredient is the length D of random documents. It is desirable to set these to a small number, in part because this length is indicative of the number of hidden global topics, and we expect the number of such global topics to be small. In particular, these global topics will allow short random documents to align with the documents to obtain "topic-based" discriminatory features. Since there is no prior information for global topics, we choose to uniformly sample the length of random documents as $D \sim \text{Uniform}[1, D_{\max}]$, for some constant $D_{\max}$. Stitching the distributions over words, and over the number of words, we then get a distribution over random documents. We note that our WME embedding allows potentially other random distributions, and other types of word embeddings, making it a flexible and powerful feature learning framework to utilize state-of-the-art techniques.

Algorithm 1 Word Mover's Embedding: An Unsupervised Feature Representation for Documents

```
Input: Texts $\{x_i\}_{i=1}^N$, $D_{\max}$, $R$.
Output: Matrix $Z_{N\times R}$, with rows corresponding to text embeddings.
1: Compute $v_{\max}$ and $v_{\min}$ as the maximum and minimum values, over all coordinates of the word vectors $\boldsymbol{v}$ of $\{x_i\}_{i=1}^N$, from any pre-trained word embeddings (e.g. Word2Vec, GloVe or PSL999).
2: for $j = 1, \ldots, R$ do
3:     Draw $D_j \sim \text{Uniform}[1, D_{\max}]$.
4:     Generate a random document $\omega_j$ consisting of $D_j$ number of random words drawn as $\omega_{j\ell} \sim \text{Uniform}[v_{\min}, v_{\max}]^d$, $\ell = 1, \ldots, D_j$.
5:     Compute $\boldsymbol{f}_{x_i}$ and $\boldsymbol{f}_{\omega_j}$ using a popular weighting scheme (e.g. NBOW or TF-IDF).
6:     Compute the WME feature vector $Z_j = \phi_{\omega_j}(\{x_i\}_{i=1}^N)$ using WMD in Equation (2).
7: end for
8: Return $Z(\{x_i\}_{i=1}^N) = \frac{1}{\sqrt{R}}[Z_1\ Z_2\ \ldots\ Z_R]$
```

Algorithm 1 summarizes the overall procedure to generate feature vectors for text of any length such as sentences, paragraphs, and documents.

KNN-WMD, which uses the WMD distance together with KNN based classification, requires $O(N^2)$ evaluations of the WMD distance, which in turn has $O(L^3 \log(L))$ complexity, assuming that documents have lengths bounded by L, leading to an overall complexity of $O(N^2 L^3 \log(L)$. In contrast, our WME approximation only requires super-linear complexity of $O(NRLlog(L))$ when D is constant. This is because in our case each evaluation of WMD only requires $O(D^2 L \log(L))$ (Bourgeois and Lassalle, 1971), due to the short length D of our random documents. This dramatic reduction in computation significantly accelerates training and testing when combined with empirical risk minimization classifiers such as SVMs. A simple yet useful trick is to pre-compute the word distances to avoid redundant computations since a pair of words may appear multiple times in different pairs of documents. Note that the computation of the ground distance between each pair of word vectors in documents has a $O(L^2 d)$ complexity, which could be close to one WMD evaluation if document length L is short and word vector dimension d is large. This simple scheme leads to additional improvement in runtime performance of our WME method that we show in our experiments.

3.3 Convergence of WME

In this section, we study the convergence of our embedding (3) to the exact kernel (2) under the framework of Random Features (RF) approximation (Rahimi and Recht, 2007). Note that the standard RF convergence theory applies only to the shift-invariant kernel operated on two vectors, while our kernel (2) operates on two documents $x, y \in \mathcal{X}$ that are sets of word vectors. In (Wu et al., 2018a), a general RF convergence theory is provided for any *distance-based kernel* as long as a finite covering number is given w.r.t. the given distance. In the following lemma, we provide the covering number for all documents of bounded length under the *Word Mover's Distance*. Without loss of generality, we will assume that the word embeddings $\{\boldsymbol{v}\}$ are normalized s.t. $\|\boldsymbol{v}\| \leq 1$.

Lemma 1. *There exists an ϵ-covering $\mathcal{E}$ of $\mathcal{X}$ under the WMD metric with Euclidean ground distance, so that:*

$$\forall x \in \mathcal{X}, \exists x_i \in \mathcal{E}, \; \textit{WMD}(x, x_i) \leq \epsilon,$$

that has size bounded as $|\mathcal{E}| \leq (\frac{2}{\epsilon})^{dL}$, where L is a bound on the length of document $x \in \mathcal{X}$.

Equipped with Lemma 1, we can derive the following convergence result as a simple corollary of the theoretical results in (Wu et al., 2018a). We defer the proof to the appendix A.

Theorem 1. *Let $\Delta_R(x, y)$ be the difference between the exact kernel (2) and the random approximation (3) with R samples, we have uniform convergence*

$$P\left\{\max_{x,y\in\mathcal{X}} |\Delta_R(x,y)| > 2t\right\} \leq 2\left(\frac{12\gamma}{t}\right)^{2dL} e^{-Rt^2/2}.$$

where d is the dimension of word embedding and L is a bound on the document length. In other words, to guarantee $|\Delta_R(x, y)| \leq \epsilon$ with probability at least $1 - \delta$, it suffices to have

$$R = \Omega\left(\frac{dL}{\epsilon^2}\log(\frac{\gamma}{\epsilon}) + \frac{1}{\epsilon^2}\log(\frac{1}{\delta})\right).$$

4 Experiments

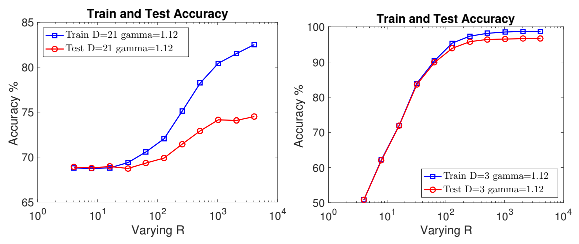
We conduct an extensive set of experiments to demonstrate the effectiveness and efficiency of the proposed method. We first compare its performance against 7 unsupervised document embedding approaches over a wide range of text classification tasks, including sentiment analysis, news categorization, amazon review, and recipe identification. We use 9 different document corpora, with 8 of these drawn from (Kusner et al., 2015; Huang et al., 2016); Table 1 provides statistics of the different datasets. We further compare our method against 10 unsupervised, semi-supervised, and supervised document embedding approaches on the 22 datasets from SemEval semantic textual similarity tasks. Our code is implemented in Matlab, and we use C Mex for the computationally intensive components of WMD (Rubner et al., 2000).

4.1 Effects of R and D on WME

Setup. We first perform experiments to investigate the behavior of the WME method by varying the number of Random Features R and the length D of random documents. The hyper-parameter γ is set via cross validation on training set over the range $[0.01, \; 10]$. We simply fix the $D_{min} = 1$, and vary D_{max} over the range $[3, \; 21]$. Due to limited space, we only show selected subsets of our results, with the rest listed in the Appendix B.2.

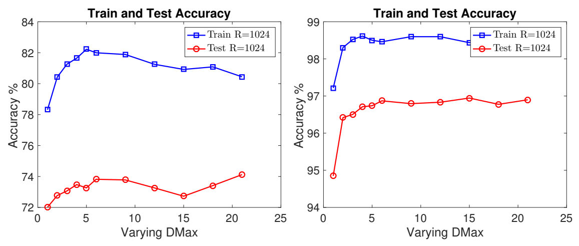
Table 1: Properties of the datasets

Dataset	C:Classes	N:Train	M:Test	BOW Dim	L:Length	Application
BBCSPORT	5	517	220	13243	117	BBC sports article labeled by sport
TWITTER	3	2176	932	6344	9.9	tweets categorized by sentiment
RECIPE	15	3059	1311	5708	48.5	recipe procedures labeled by origin
OHSUMED	10	3999	5153	31789	59.2	medical abstracts (class subsampled)
CLASSIC	4	4965	2128	24277	38.6	academic papers labeled by publisher
REUTERS	8	5485	2189	22425	37.1	news dataset (train/test split)
AMAZON	4	5600	2400	42063	45.0	amazon reviews labeled by product
20NEWS	20	11293	7528	29671	72	canonical user-written posts dataset
RECIPE_L	20	27841	11933	3590	18.5	recipe procedures labeled by origin



(a) TWITTER (b) CLASSIC

Figure 2: Train (Blue) and Test (Red) accuracy when varying R with fixed D.



(a) TWITTER (b) CLASSIC

Figure 3: Train (Blue) and Test (Red) accuracy when varying D with fixed R.

Effects of R. We investigate how the performance changes when varying the number of Random Features R from 4 to 4096 with fixed D. Fig. 2 shows that both training and testing accuracies generally converge very fast when increasing R from a small number ($R = 4$) to a relatively large number ($R = 1024$), and then gradually reach to the optimal performance. This confirms our analysis in Theory 1 that the proposed WME can guarantee the fast convergence to the exact kernel.

Effects of D. We further evaluate the training and testing accuracies when varying the length of random document D with fixed R. As shown in Fig. 3, we can see that near-peak performance can usually be achieved when D is small (typically $D \leq 6$). This behavior illustrates two important aspects: (1) using very few random words (e.g. $D = 1$) is not enough to generate useful Random Features when R becomes large; (2) using too many random words (e.g. $D \geq 10$) tends to generate similar and redundant Random Features when increasing R. Conceptually, the number of random words in a random document can be thought of as the number of the global topics in documents, which is generally small. This is an important desired feature that confers both a performance boost as well as computational efficiency to the WME method.

4.2 Comparison with KNN-WMD

Baselines. We now compare two WMD-based methods in terms of testing accuracy and total training and testing runtime. We consider two variants of WME with different sizes of R. WME(LR) stands for WME with large rank that achieves the best accuracy (using R up to 4096) with more computational time, while WME(SR) stands for WME with small rank that obtains comparable accuracy in less time. We also consider two variants of both methods where +P denotes that we precompute the ground distance between each pair of words to avoid redundant computations.

Setup. Following (Kusner et al., 2015; Huang et al., 2016), for datasets that do not have a predefined train/test split, we report average and standard deviation of the testing accuracy and average run-time of the methods over five 70/30 train/test splits. For WMD, we provide the results (with respect to accuracy) from (Kusner et al., 2015); we also reran the experiments of KNN-WMD and found them to be consistent with the reported results. For all methods, we perform 10-fold cross validation to search for the best parameters on the training documents. We employ a linear SVM implemented using LIBLINEAR (Fan et al., 2008) on WME since it can isolate the effectiveness of the feature representation from the power of the nonlinear learning solvers. For additional results on all KNN-based methods, please refer to Appendix B.3.

Results. Table 2 corroborates the significant advan-

Table 2: Test accuracy, and total training and testing time (in seconds) of WME against KNN-WMD. Speedups are computed between the best numbers of KNN-WMD+P and these of WME(SR)+P when achieving similar testing accuracy. Bold face highlights the best number for each dataset.

Classifier	KNN-WMD		KNN-WMD+P	WME(SR)		WME(SR)+P	WME(LR)		WME(LR)+P	
Dataset	Accu	Time	Time	Accu	Time	Time	Accu	Time	Time	Speedup
BBCSPORT	95.4 ± 1.2	147	122	95.5 ± 0.7	3	1	**98.2 ± 0.6**	92	34	**122**
TWITTER	71.3 ± 0.6	25	4	72.5 ± 0.5	10	2	**74.5 ± 0.5**	162	34	**2**
RECIPE	57.4 ± 0.3	448	326	57.4 ± 0.5	18	4	**61.8 ± 0.8**	277	61	**82**
OHSUMED	55.5	3530	2807	55.8	24	7	**64.5**	757	240	**401**
CLASSIC	**97.2 ± 0.1**	777	520	96.6 ± 0.2	49	10	97.1 ± 0.4	388	70	**52**
REUTERS	96.5	814	557	96.0	50	24	**97.2**	823	396	**23**
AMAZON	92.6 ± 0.3	2190	1319	92.7 ± 0.3	31	8	**94.3 ± 0.4**	495	123	**165**
20NEWS	73.2	37988	32610	72.9	205	69	**78.3**	1620	547	**472**
RECIPE_L	71.4 ± 0.5	5942	2060	72.5 ± 0.4	113	20	**79.2 ± 0.3**	1838	330	**103**

Table 3: Testing accuracy of WME against Word2Vec and Doc2Vec-based methods.

Dataset	SIF(GloVe)	Word2Vec+nbow	Word2Vec+tf-idf	PV-DBOW	PV-DM	Doc2VecC	WME
BBCSPORT	97.3 ± 1.2	97.3 ± 0.9	96.9 ± 1.1	97.2 ± 0.7	97.9 ± 1.3	90.5 ± 1.7	**98.2 ± 0.6**
TWITTER	57.8 ± 2.5	72.0 ± 1.5	71.9 ± 0.7	67.8 ± 0.4	67.3 ± 0.3	71.0 ± 0.4	**74.5 ± 0.5**
OHSUMED	**67.1**	63.0	60.6	55.9	59.8	63.4	64.5
CLASSIC	92.7 ± 0.9	95.2 ± 0.4	93.9± 0.4	97.0 ± 0.3	96.5 ± 0.7	96.6 ± 0.4	**97.1 ± 0.4**
REUTERS	87.6	96.9	95.9	96.3	94.9	96.5	**97.2**
AMAZON	94.1 ± 0.2	94.0 ± 0.5	92.2 ± 0.4	89.2 ± 0.3	88.6 ± 0.4	91.2 ± 0.5	**94.3 ± 0.4**
20NEWS	72.3	71.7	70.2	71.0	74.0	78.2	**78.3**
RECIPE_L	71.1 ± 0.5	74.9 ± 0.5	73.1 ± 0.6	73.1 ± 0.5	71.1 ± 0.4	76.1 ± 0.4	**79.2 ± 0.3**

tages of WME compared to KNN-WMD in terms of both accuracy and runtime. First, WME(SR) can consistently achieve better or similar accuracy compared to KNN-WMD while requiring order-of-magnitude less computational time on all datasets. Second, both methods can benefit from precomputation of the ground distance between a pair of words but WME gains much more from prefetch (typically 3-5x speedup). This is because the typical length D of random documents is very short where computing ground distance between word vectors may be even more expensive than the corresponding WMD distance. Finally, WME(LR) can achieve much higher accuracy compared to KNN-WMD while still often requiring less computational time, especially on large datasets like 20NEWS and relatively long documents like OHSUMED.

4.3 Comparisons with Word2Vec & Doc2Vec

Baselines. We compare against 6 document representations methods: 1) *Smooth Inverse Frequency* (SIF) (Arora et al., 2017): a recently proposed simple but tough to beat baseline for sentence embeddings, combining a new weighted scheme of word embeddings with dominant component removal; 2) *Word2Vec+nbow*: a weighted average of word vectors using NBOW weights; 3) *Word2Vec+tf-idf*: a weighted average of word vectors using TF-IDF weights; 4) *PV-DBOW* (Le and Mikolov, 2014): distributed bag of words model of Paragraph Vectors; 5) *PV-DM* (Le and Mikolov, 2014): distributed memory model of Paragraph Vectors; 6) *Doc2VecC* (Chen, 2017): a recently proposed document-embedding via corruptions, achieving state-of-the-art performance in text classification.

Setup. *Word2Vec+nbow*, *Word2Vec+tf-idf* and WME use pre-trained Word2Vec embeddings while SIF uses its default pre-trained GloVe embeddings. Following (Chen, 2017), to enhance the performance of *PV-DBOW*, *PV-DM*, and *Doc2VecC* these methods are trained transductively on both train and test, which is indeed beneficial for generating a better document representation (see Appendix B.4). In contrast, the hyperparameters of WME are obtained through a 10-fold cross validation only on training set. For a fair comparison, we run a linear SVM using LIBLINEAR on all methods.

Results. Table 3 shows that WME consistently outperforms or matches existing state-of-the-art document representation methods in terms of testing accuracy on all datasets except one (OHSUMED). The first highlight is that simple average of word embeddings often achieves better performance than *SIF(Glove)*, indicating that removing the first principle component could hurt the expressive power of the resulting representation for some of classification tasks. Surprisingly, these two methods often achieve similar or better performance than *PV-DBOW* and *PV-DM*, which may be because of the high-quality pre-trained word embeddings. On

Table 4: Pearson's scores of WME against other unsupervised, semi-supervised, and supervised methods on 22 textual similarity tasks. Results are collected from (Arora et al., 2017) except our approach.

Approaches	Supervised						Unsupervised					Semi-supervised	
WordEmbeddings	PSL						GloVe					PSL	
Tasks	PP	Dan	RNN	iRNN	LSTM(no)	LSTM(o.g.)	ST	nbow	tf-idf	SIF	WME	SIF	WME
STS'12	58.7	56.0	48.1	58.4	51.0	46.4	30.8	52.5	58.7	56.2	60.6	59.5	**62.8**
STS'13	55.8	54.2	44.7	56.7	45.2	41.5	24.8	42.3	52.1	56.6	54.5	**61.8**	56.3
STS'14	70.9	69.5	57.7	70.9	59.8	51.5	31.4	54.2	63.8	68.5	65.5	**73.5**	68.0
STS'15	75.8	72.7	57.2	75.6	63.9	56.0	31.0	52.7	60.6	71.7	61.8	**76.3**	64.2
SICK'14	71.6	70.7	61.2	71.2	63.9	59.0	49.8	65.9	69.4	72.2	68.0	**72.9**	68.1
Twitter'15	52.9	**53.7**	45.1	52.9	47.6	36.1	24.7	30.3	33.8	48.0	41.6	49.0	47.4

the other hand, *Doc2VecC* achieves much better testing accuracy than these previous methods on two datasets (20NEWS, and RECIPE_L). This is mainly because that it benefits significantly from transductive training (See Appendix B.4). Finally, the better performance of WME over these strong baselines stems from fact that WME is empowered by two important building blocks, WMD and Word2Vec, to yield a more informative representation of the documents by considering both the word alignments and the semantics of words.

We refer the readers to additional results on the Imdb dataset in Appendix B.4, which also demonstrate the clear advantage of WME even compared to the supervised RNN method as well as the aforementioned baselines.

4.4 Comparisons on textual similarity tasks

Baselines. We compare WME against 10 supervised, simi-sepervised, and unsupervised methods for performing textual similarity tasks. Six supervised methods are initialized with Paragram-SL999(PSL) word vectors (Wieting et al., 2015b) and then trained on the PPDB dataset, including: 1) *PARAGRAM-PHRASE* (PP) (Wieting et al., 2015a): simple average of refined PSL word vectors; 2) *Deep Averaging Network* (DAN) (Iyyer et al., 2015); 3) *RNN*: classical recurrent neural network; 4) *iRNN*: a variant of RNN with the activation being the identify; 5) *LSTM(no)* (Gers et al., 2002): LSTM with no output gates; 6) *LSTM(o.g.)* (Gers et al., 2002): LSTM with output gates. Four unsupervised methods are: 1) *Skip-Thought Vectors* (ST) (Kiros et al., 2015): an encoder-decoder RNN model for generalizing Skip-gram to the sentence level; 2) *nbow*: simple averaging of pre-trained GloVe word vectors; 3) *tf-idf*: a weighted average of GloVe word vecors using TF-IDF weights; 4) *SIF* (Arora et al., 2017): a simple yet strong method on textual similarity tasks using GloVe word vecors. Two semi-supervised methods use PSL word vectors, which are trained using labeled data (Wieting et al., 2015b).

Setup. There are total 22 textual similarity datasets from STS tasks (2012-2015) (Agirre et al., 2012, 2013, 2014, 2015), SemEval 2014 Semantic Relatedness task (Xu et al., 2015), and SemEval 2015 Twitter task (Marelli et al., 2014). The goal of these tasks is to predict the similarity between two input sentences. Each year STS usually has 4 to 6 different tasks and we only report the averaged Pearson's scores for clarity. Detailed results on each dataset are listed in Appendix B.5.

Results. Table 4 shows that WME consistently matches or outperforms other unsupervised and supervised methods except the *SIF* method. Indeed, compared with *ST* and *nbow*, WME improves Pearson's scores substantially by 10% to 33% as a result of the consideration of word alignments and the use of TF-IDF weighting scheme. *tf-idf* also improves over these two methods but is slightly worse than our method, indicating the importance of taking into account the alignments between the words. *SIF* method is a strong baseline for textual similarity tasks but WME still can beat it on STS'12 and achieve close performance in other cases. Interestingly, WME is on a par with three supervised methods *RNN*, *LSTM(no)*, and *LSTM(o.g.)* in most cases. The final remarks stem from the fact that, WME can gain significantly benefit from the supervised word embeddings similar to *SIF*, both showing strong performance on PSL.

5 Related Work

Two broad classes of *unsupervised* and *supervised* methods have been proposed to generate sentence and document representations. The former primarily generate general purpose and domain independent embeddings of word sequences (Socher et al., 2011; Kiros et al., 2015; Arora et al., 2017); many unsupervised training research efforts have focused on either training an auto-encoder to learn the latent structure of a sentence (Socher et al., 2013), a paragraph, or document (Li et al., 2015); or generalizing Word2Vec models to predict words in a paragraph (Le and Mikolov, 2014; Chen, 2017) or in neighboring sentences (Kiros et al., 2015). However, some important information could be lost in the resulting document representation without considering the word order. Our proposed WME overcomes this difficulty by considering the alignments between each pair of words.

The other line of work has focused on developing compositional supervised models to create a vector representation of sentences (Kim et al., 2016; Gong et al., 2018b). Most of this work proposed composition using recursive neural networks based on parse structure (Socher et al., 2012, 2013), deep averaging networks over bag-of-words models (Iyyer et al., 2015; Wieting et al., 2015a), convolutional neural networks (Kim, 2014; Kalchbrenner et al., 2014; Xu et al., 2018), and recurrent neural networks using long short-term memory (Tai et al., 2015; Liu et al., 2015). However, these methods are less well suited for domain adaptation settings.

6 Conclusion

In this paper, we have proposed an alignment-aware text kernel using WMD for texts of variable lengths, which takes into account both word alignments and pre-trained high quality word embeddings in learning an effective semantics-preserving feature representation. The proposed WME is simple, efficient, flexible, and unsupervised. Extensive experiments show that WME consistently matches or outperforms state-of-the-art models on various text classification and textual similarity tasks. WME embeddings can be easily used for a wide range of downstream supervised and unsupervised tasks.

References

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel M Cer, Mona T Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Inigo Lopez-Gazpio, Montse Maritxalar, Rada Mihalcea, et al. 2015. Semeval-2015 task 2: Semantic textual similarity, english, spanish and pilot on interpretability. In *SemEval@ NAACL-HLT*, pages 252–263.

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

Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. 2013. sem 2013 shared task: Semantic textual similarity, including a pilot on typed-similarity. In *In* SEM 2013: The Second Joint Conference on Lexical and Computational Semantics. Association for Computational Linguistics*. Citeseer.

Eneko Agirre, Mona Diab, Daniel Cer, and Aitor Gonzalez-Agirre. 2012. Semeval-2012 task 6: A pilot on semantic textual similarity. In *Proceedings of 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*, pages 385–393. Association for Computational Linguistics.

Jason Altschuler, Jonathan Weed, and Philippe Rigollet. 2017. Near-linear time approximation algorithms for optimal transport via sinkhorn iteration. In *Advances in Neural Information Processing Systems*, pages 1964–1974.

Sanjeev Arora, Yuanzhi Li, Yingyu Liang, Tengyu Ma, and Andrej Risteski. 2016. A latent variable model approach to pmi-based word embeddings. *Transactions of the Association for Computational Linguistics*, 4:385–399.

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A simple but tough-to-beat baseline for sentence embeddings. In *ICLR*.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022.

François Bourgeois and Jean-Claude Lassalle. 1971. An extension of the munkres algorithm for the assignment problem to rectangular matrices. *Communications of the ACM*, 14(12):802–804.

Chris Buckley, Gerard Salton, James Allan, and Amit Singhal. 1995. Automatic query expansion using smart: Trec 3. *NIST special publication sp*, pages 69–69.

Minmin Chen. 2017. Efficient vector representation for documents through corruption. In *ICLR*.

Minmin Chen, Zhixiang Xu, Kilian Weinberger, and Fei Sha. 2012. Marginalized denoising autoencoders for domain adaptation. *Proceedings of the 29th international conference on Machine learning*.

Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. Liblinear: A library for large linear classification. *Journal of machine learning research*, 9(Aug):1871–1874.

Felix A Gers, Nicol N Schraudolph, and Jürgen Schmidhuber. 2002. Learning precise timing with lstm recurrent networks. *Journal of machine learning research*, 3(Aug):115–143.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 513–520.

Hongyu Gong, Suma Bhat, and Pramod Viswanath. 2018a. Embedding syntax and semantics of prepositions via tensor decomposition. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 896–906.

Hongyu Gong, Jiaqi Mu, Suma Bhat, and Pramod Viswanath. 2017. Prepositions in context. *arXiv preprint arXiv:1702.01466*.

Hongyu Gong, Tarek Sakakini, Suma Bhat, and JinJun Xiong. 2018b. Document similarity for texts of varying lengths via hidden topics. In *ACL*, volume 1, pages 2341–2351.

Tom Griffiths and Mark Steyvers. 2007. Probabilistic topic models. *Latent Semantic Analysis: A Road to Meaning*.

Jie Gui, Tongliang Liu, Dacheng Tao, Zhenan Sun, and Tieniu Tan. 2016. Representative vector machines: a unified framework for classical classifiers. *IEEE transactions on cybernetics*, 46(8):1877–1888.

Jie Gui, Zhenan Sun, Jun Cheng, Shuiwang Ji, and Xindong Wu. 2014. How to estimate the regularization parameter for spectral regression discriminant analysis and its kernel version? *IEEE Transactions on Circuits and Systems for Video Technology*, 24(2):211–223.

Bernard Haasdonk and Claus Bahlmann. 2004. Learning with distance substitution kernels. In *Joint Pattern Recognition Symposium*, pages 220–227. Springer.

Frank L Hitchcock. 1941. The distribution of a product from several sources to numerous localities. *Studies in Applied Mathematics*, 20(1-4):224–230.

Gao Huang, Chuan Guo, Matt J Kusner, Yu Sun, Fei Sha, and Kilian Q Weinberger. 2016. Supervised word mover's distance. In *Advances in Neural Information Processing Systems*, pages 4862–4870.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. 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)*, volume 1, pages 1681–1691.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302.

Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. 2015. From word embeddings to document distances. In *International Conference on Machine Learning*, pages 957–966.

Quoc V Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *ICML*, volume 14, pages 1188–1196.

Jiwei Li, Minh-Thang Luong, and Dan Jurafsky. 2015. A hierarchical neural autoencoder for paragraphs and documents. *arXiv preprint arXiv:1506.01057*.

Pengfei Liu, Xipeng Qiu, Xinchi Chen, Shiyu Wu, and Xuanjing Huang. 2015. Multi-timescale long short-term memory neural network for modelling sentences and documents. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 2326–2335.

Marco Marelli, Luisa Bentivogli, Marco Baroni, Raffaella Bernardi, Stefano Menini, and Roberto Zamparelli. 2014. Semeval-2014 task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. In *SemEval@ COLING*, pages 1–8.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Quoc V Le, and Ilya Sutskever. 2013b. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013c. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Xi Peng, Rogerio S Feris, Xiaoyu Wang, and Dimitris N Metaxas. 2016. A recurrent encoder-decoder network for sequential face alignment. In *European conference on computer vision*, pages 38–56. Springer, Cham.

Xi Peng, Shaoting Zhang, Yu Yang, and Dimitris N Metaxas. 2015. Piefa: Personalized incremental and ensemble face alignment. In *Proceedings of the IEEE international conference on computer vision*, pages 3880–3888.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543.

Hieu Pham, Minh-Thang Luong, and Christopher D Manning. 2015. Learning distributed representations for multilingual text sequences. In *Proceedings of NAACL-HLT*, pages 88–94.

Ali Rahimi and Benjamin Recht. 2007. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems*, page 5.

Stephen E Robertson and Steve Walker. 1994. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *ACM SIGIR conference on Research and development in information retrieval*.

Stephen E Robertson, Steve Walker, Susan Jones, Micheline M Hancock-Beaulieu, Mike Gatford, et al. 1995. Okapi at trec-3. *Nist Special Publication Sp*, 109:109.

Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. 2000. The earth mover's distance as a metric for image retrieval. *International journal of computer vision*, 40(2):99–121.

Gerard Salton and Christopher Buckley. 1988. Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523.

Richard Socher, Eric H Huang, Jeffrey Pennin, Christopher D Manning, and Andrew Y Ng. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Advances in neural information processing systems*, pages 801–809.

Richard Socher, Brody Huval, Christopher D Manning, and Andrew Y Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *EMNLP*, pages 1201–1211. Association for Computational Linguistics.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, Christopher Potts, et al. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*.

Sida Wang and Christopher D Manning. 2012. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 90–94. Association for Computational Linguistics.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015a. Towards universal paraphrastic sentence embeddings. *arXiv preprint arXiv:1511.08198*.

John Wieting, Mohit Bansal, Kevin Gimpel, Karen Livescu, and Dan Roth. 2015b. From paraphrase database to compositional paraphrase model and back. *Transactions of the ACL (TACL)*.

Lingfei Wu, Eloy Romero, and Andreas Stathopoulos. 2017. Primme_svds: A high-performance preconditioned svd solver for accurate large-scale computations. *SIAM Journal on Scientific Computing*, 39(5):S248–S271.

Lingfei Wu and Andreas Stathopoulos. 2015. A preconditioned hybrid svd method for accurately computing singular triplets of large matrices. *SIAM Journal on Scientific Computing*, 37(5):S365–S388.

Lingfei Wu, Ian En-Hsu Yen, Fnagli Xu, Pradeep Ravikumar, and Witbrock Michael. 2018a. D2ke: From distance to kernel and embedding. *https://arxiv.org/abs/1802.04956*.

Lingfei Wu, Ian En-Hsu Yen, Jinfeng Yi, Fangli Xu, Qi Lei, and Michael Witbrock. 2018b. Random warping series: A random features method for time-series embedding. In *International Conference on Artificial Intelligence and Statistics*, pages 793–802.

Kun Xu, Lingfei Wu, Zhiguo Wang, and Vadim Sheinin. 2018. Graph2seq: Graph to sequence learning with attention-based neural networks. *arXiv preprint arXiv:1804.00823*.

Wei Xu, Chris Callison-Burch, and Bill Dolan. 2015. Semeval-2015 task 1: Paraphrase and semantic similarity in twitter (pit). In *SemEval@ NAACL-HLT*, pages 1–11.

Yue Zhang, Qi Liu, and Linfeng Song. 2018. Sentence-state lstm for text representation. *arXiv preprint arXiv:1805.02474*.