WORD MOVER'S EMBEDDING: FROM WORD2VEC TO DOCUMENT EMBEDDING

Anonymous authors
Paper under double-blind review

ABSTRACT

Learning effective text representations is a key foundation for numerous machine learning and NLP applications. While the celebrated Word2Vec technique yields semantically rich word representations, it is less clear whether sentence or document representations should be built upon word representations or from scratch. 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 very expensive to compute, and is harder to apply beyond simple KNN than feature embeddings. 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. 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. In 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 tasks such as document classification and clustering (Chen, 2017), document retrieval (Le & Mikolov, 2014), machine translation (Mikolov et al., 2013b), and multi-lingual document matching (Pham et al., 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 & Buckley, 1988) and its term frequency variants (e.g. TF-IDF) (Robertson & Walker, 1994) are most widely used due to simplicity, efficiency and often surprisingly high accuracy (Wang & Manning, 2012). However, simply treating words and phrases as discrete symbols fails to take into account word order and 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 & Stathopoulos, 2015) and learning a probabilistic BOW clustering. Other work learns a suitable representation using Denoising Autoencoders (Chen et al., 2012) for text documents.

Another family of research is use of neural-network language models for learning distributed representations of words, phrases, and documents (Bengio et al., 2003; Mikolov et al., 2010; 2013a;c; Le & Mikolov, 2014; Dai & Le, 2015; Wieting et al., 2015a; Kim et al., 2016; Chen, 2017; Arora et al., 2017). The celebrated Word2Vec technique (Mikolov et al., 2013a;c) presented two shallow yet effective models to learn vector representations of words (and phrases) by mapping those of similar meaning nearby 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 (Mikolov et al., 2013a).

Inspired by these successful techniques, a number of researchers recently proposed various methods for learning a sentence or document representation beyond the lexical level. A simple but often effective approach is to use a weighted average over some or all of the words in the document. The advantage of the document representation building on top of word-level embeddings is that one can make full use of high-quality pre-trained word embeddings obtained from an extremely large corpus, such as 100-billion words of Google News (Mikolov et al., 2013a). A more sophisticated approach (Le & Mikolov, 2014; Chen, 2017) is to jointly learn embeddings for both words and paragraphs simultaneously using models similar to Word2Vec, incorporating each document as an input. However, word order may not be fully captured by a small context window, and the quality of word embeddings learned in such a model may be limited by the size of the training corpus. These effects may 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. WMD computes the minimum amount of distance covered in transforming each word from one document into the other, where this ground distance is computed as the Euclidean distance of a pair of words 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 costs. More importantly, WMD is simply a distance that can be combined with KNN or K-means, whereas many machine learning algorithms require fixed-length feature representation as input.

In this paper, we 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. We develop a new approach to constructing a *positive-definite* (p.d.) kernel called the *Word Mover's Kernel*, and its corresponding feature embedding for documents. The *Word Mover's Kernel* defines an explicit feature map given by a distribution of random documents, which exploits WMD to find alignments between a set of words represented in a Word2Vec embedding space between text documents and random documents. In addition, to scale up the proposed kernel we further design the random features in such a way that the exact kernel value can be approximated by the inner products of the transformed feature representation. WME is fully parallelizable, and is highly extensible where its two building blocks, Word2Vec and WMD, can be replaced by other techniques such as GloVe (Pennington et al., 2014) or S-WMD (Huang et al., 2016).

We summarize our main contributions as follows: (i) We propose a novel alignment-aware text kernel for unstructured text data where word alignment is important in learning an effective feature representation. The heart of *Word Mover's Kernel* is a new methodology to transform a distance metric to a p.d. kernel. (ii) We present WME, a random features method for *Word Mover's Kernel* to learn an unsupervised semantic-preserving document embedding based on high-quality word embeddings. Compared to KNN-based WMD, WME admits an efficient computation which substantially reduces the total complexity from $O(N^2L^3\log(L))$ to $O(NRL\log(L))$ in the number N and length L of documents. (iii) We give an analysis showing a number $R = \Omega(1/\epsilon^2)$ of WME suffices for the convergence to within ϵ of the precision of the exact kernel. The proposed analysis extends the Monte Carlo analysis in (Rahimi & Recht, 2007) from a shift-invariant kernel of fixed-dimensional vectors to a text kernel of documents including a set of word vectors of variable length without requiring shift-invariant property. (iv) 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. In particular, WME often achieves orders of magnitude speed-up compared to KNN-WMD when obtaining the same testing accuracy.

2 WORD2VEC AND WORD MOVER'S DISTANCE

We briefly introduce Word2Vec and WMD, two important building blocks to 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. The current most popular word embedding technique known as Word2Vec was presented by Mikolov et al. in their seminal papers (Mikolov et al., 2013a;c). There are two well-known model

architectures: Continuous Bag-of-Words (CBOW) and Skip-gram models. Both models consist of an input layer, a projection layer, and an output layer. CBOW model defines the probability of predicting the current word w in a document x from a window of surrounding context words, and thus the order of words in the context window does not influence the prediction. In contrast, Skip-gram model defines the probability of using the current word to predict the words within a window of before and after the current word. Typically, distant words are less relevant to the current word and thus get lower weights. To train both models, the word vectors $\boldsymbol{v}_w$ are then learned to maximize the log-likelihood of the conditional probability of predicting the target word at each position. Various techniques such as hierarchical soft-max, negative sampling, sub-sampling of frequent words were presented to effectively train an embedding over hundreds of billions of words, which empowers Word2Vec to capture surprisingly accurate word relationship (Mikolov et al., 2013c).

A number of methods have since been proposed to generate document representations from word embeddings (Socher et al., 2012; Tai et al., 2015; Kiros et al., 2015; Le & Mikolov, 2014; Kusner et al., 2015; Huang et al., 2016; Chen, 2017; Arora et al., 2017), among which the weighted average of word vectors is the simplest approach. Rationale behind this simple scheme is that syntactic and semantic regularities of phrases and sentences are reasonably well preserved by adding or subtracting word embedding vectors. However, despite its simplicity, 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. Throughout this paper we use Word2Vec as our first building block but other (supervised or unsupervised) word embeddings (Pennington et al., 2014; Wieting et al., 2015b) could also be utilized.

Word Mover's Distance. Word Mover's Distance is introduced by Kusner et al. (2015) as a special case of Earth Mover's Distance (Rubner et al., 2000), which can be computed as a solution of the well-known transportation problem (Hitchcock, 1941). 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 $w_1, \ldots, w_L$ ($L = max(|x|, |y|)$) in x and y, and one choice of $\boldsymbol{f}_x \in \mathbb{R}^{|x|}$, $\boldsymbol{f}_y \in \mathbb{R}^{|y|}$ could be the normalized frequency vectors of each word in x and y respectively (so $\boldsymbol{f}_x^\mathsf{T}\mathbf{1} = \boldsymbol{f}_y^\mathsf{T}\mathbf{1} = 1$). Then the WMD is defined as

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

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

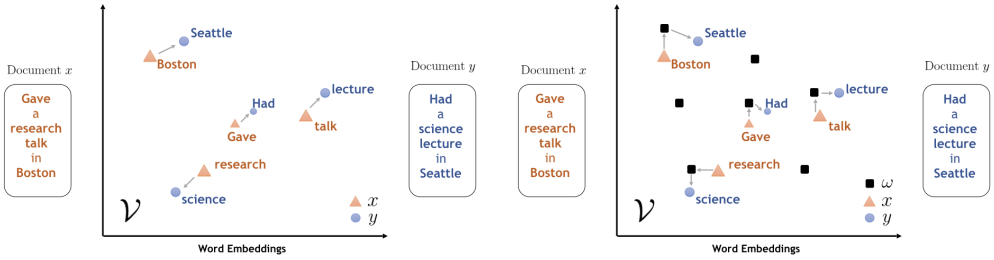
The WMD distance has been observed to perform much better on KNN-based classification tasks (Kusner et al., 2015). However, WMD is expensive to compute with computational complexity of $O(L^3 \log(L))$, especially for long documents (L is large). Additionally, since WMD is just a document distance rather than a document representation, it incurs even higher computational costs $O(N^2 L^3 \log(L))$ when using KNN. To overcome these drawbacks, we present WME, an unsupervised document embedding technique for efficiently learning a semantic-preserving vector representation of texts of variable lengths.

3 Document Embedding via Word Mover's Kernel

3.1 Word Mover's Kernel

In this section, we introduce a methodology to derive a p.d. kernel from an alignment-aware distance metric, which then gives us an effective vector representation of document as a by-product through the generalized theory of *Random Feature Approximation* (Rahimi & Recht, 2007). The *Word Mover's Kernel* is defined as

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



(a) WMD (b) WME

Figure 1: An illustration of the WMD (1a) and WME (1b). All non-stop words are marked as bold face. WMD measures the distance between two documents. WME approximates WMD with a set of random documents through triangular inequality in a low dimensional embedding space.

Here 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 infinite-dimensional feature map derived from the WMD between x and all possible documents $\omega \in \Omega$. An insightful interpretation of the kernel (2) expresses it as

$$k(x, y) := \exp\left(-\gamma \mathit{softmin}_{p(\omega)}\{\mathrm{WMD}(x, \omega) + \mathrm{WMD}(\omega, y)\}\right) \tag{3}$$

$$\textit{where} \quad \mathit{softmin}_{p(\omega)}\, f(\omega) := -\frac{1}{\gamma}\log \int p(\omega) e^{-\gamma f(\omega)} d\omega \tag{4}$$

is a version of soft minimum function parameterized by $p(\omega)$ and γ. Compared with the usual definition of soft minimum $\mathit{softmin}_i f_i := -\mathit{softmax}_i(-f_i) = -\log \sum_i e^{-f_i}$, (4) is re-weighted by a probability density $p(\omega)$ and has one more parameter γ to control the degree of smoothness. When γ is large and $f(\omega)$ is Lipschitz-continuous, the value of (4) is mostly determined by the minimum of $f(\omega)$. Note that since WMD is a metric, by the triangular inequality we have

$$\mathrm{WMD}(x, y) \;\leq\; \min_{\omega \in \Omega} \left(\mathrm{WMD}(x, \omega) + \mathrm{WMD}(\omega, y)\right) \tag{5}$$

and the equality holds if we allow the length of random document $D_{\max}$ to be not smaller than L. Therefore, the kernel (3) serves as a good approximation to the WMD between any pair of documents x, y as illustrated in Figure 1b, while it is *positive-definite* by the definition.

3.2 Word Mover's Embedding

It is not straightforward to derive a simple analytic form of the kernel (2), but we can utilize a Monte-Carlo method to simply yield a random approximation of the form,

$$k(x, y) \approx \langle Z(x), Z(y) \rangle = \frac{1}{R}\sum_{i=1}^{R} \phi_{\omega_i}(x)\phi_{\omega_i}(y) \tag{6}$$

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*, which we will show in the next section this random approximation (6) converges to the exact kernel (2) uniformly over all pairs of documents (x, y) .

Algorithm 1 summarizes the procedure to generate feature vectors for text of any length such as sentences, paragraphs, and documents. We highlight several important features here. First of all, the distribution $p(\omega)$ needs to capture the characteristics of the Word2Vec embedding space in order to generate a meaningful random word. Several studies have found that the word vectors $\boldsymbol{v}$ roughly dispersed uniformly in the word embedding space (Arora et al., 2016; 2017), which is consistent with our experimental finding that the uniform distribution centered by the mean of all word vectors in the documents is generally applicable for various text corpus. Second, the length of random documents

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

Input: Text documents $\{x_i\}_{i=1}^N, 1 < |x_i| < L$, D_{max}, R, uniform distribution $p(\omega) = rand()$.
Output: Feature matrix $Z_{N\times R}$ for texts of any length

1: Compute v_{max} and v_{min} in word vectors $\boldsymbol{v}$ of $\{x_i\}_{i=1}^N$ from any pre-trained word embeddings
2: **for** $j = 1, \ldots, R$ **do**
3: Draw D_j uniformly from $[1, D_{max}]$.
4: Generate a random document ω_j consisting of D_j number of random words drawn from $(v_{min} + (v_{max} - v_{min}) \times (rand(d, D_j)))$.
5: Compute $\boldsymbol{f}_{x_i}$ and $\boldsymbol{f}_{\omega_j}$ using a popular weighting scheme (e.g. NBOW or TF-IDF).
6: Compute a feature vector $Z_j = \phi_{\omega_j}(x_{i=1}^N))$ using WMD in Equation (2).
7: **end for**
8: Return matrix $Z(\{x_i\}_{i=1}^N) = \frac{1}{\sqrt{R}}[Z_1^{'}\ Z_2^{'}\ \ldots\ Z_R^{'}]$

D is typically a quite small number. It suggests that there are some hidden global topics that allow short random documents to align with text documents to obtain discriminatory features. Since there is no prior information for hidden global topics, we choose to uniformly sample the length of random documents from a range $[1, D_{max}]$ to give an unbiased estimate of D. Finally, WME allows any types of word embeddings and weighting schemes, making it a flexible and powerful feature learning framework to take full advantage of state-of-the-art techniques.

To efficiently approximate the Word Mover's kernel, we enjoy the double benefits of improved accuracy and reduced computation due to WME. Compared to high computational costs of KNN-WMD, it requires $O(N^2)$ times evaluation of WMD which takes $O(L^3 log(L))$ complexity assuming that all documents have similar lengths L. In contrast, our WME approximation only requires super-linear complexity of $O(NRLlog(L))$ computation if D is treated as a constant. This is because one evaluation of WMD only requires $O(D^2 Llog(L))$ (Bourgeois & Lassalle, 1971) thanks to the short random documents. This dramatic reduction in computation significantly accelerates training and testing when combining with empirical risk minimization classifiers such as SVM. In practice, the computation of ground distance between each pair of word vectors in documents results in $O(L^2 d)$ complexity, which could be close to one WMD evaluation if document length L is short and word vector dimension d is large. 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. This simple scheme leads to additional improvement of the runtime performance of WME which we will show in the experiments.

3.3 Convergence of Word Mover's Embedding

In this section, we study the convergence of our embedding (6) to the exact kernel (2) under the framework of Random Feature (RF) approximation (Rahimi & Recht, 2007). Note that the standard RF convergence theory applies only to the shift-invariant kernel operated directly on two vectors, while our kernel (2) operates on two documents $x, y \in \mathcal{X}$ that are sets of word vectors without requiring shift-invariant property. The following lemma fills this gap by constructing a ϵ-covering for the space of *set of vectors* from a ϵ-covering of vector space. Without loss of generality, we will assume that the word vectors $\{\boldsymbol{v}\}$ are normalized s.t. $\|\boldsymbol{v}\| \leq 1$.

Lemma 1. *There is an ϵ-covering $\mathcal{E}$ of $\mathcal{X}$ under the metric defined by WMD with Euclidean ground distance and*

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

with $|\mathcal{E}| \leq (\frac{2}{\epsilon})^{dL_{\max}}$, where $L_{\max}$ is a bound on the length of document $x \in \mathcal{X}$.

Equipped with Lemma 1, we give the following convergence theorem, which follows the spirit of (Rahimi & Recht, 2007) but generalizes it to the case without shift-invariance.

Theorem 1. *Let $\Delta_R(x, y)$ be the difference between the exact kernel* (2) *and the random approximation* (6) *with R samples, we have uniform convergence*

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

where d is the dimension of word embedding and $L_{\max}$ 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

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

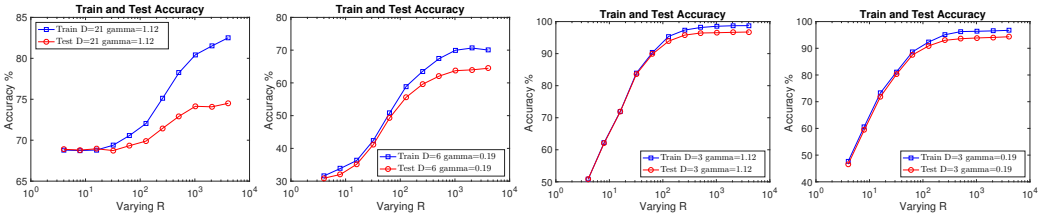
4 EXPERIMENTS

We conduct extensive sets of experiments to demonstrate the effectiveness and efficiency of the proposed method. We first compare its performance against 7 baselines over a wide range of text classification tasks, including sentiment analysis, news categorization, amazon review, recipe identification, and so on. We choose 9 different document corpora where 8 of them are overlapped with datasets in (Kusner et al., 2015; Huang et al., 2016). A complete data statistics is in Table 1. We further compare our method against 10 baselines on the 22 datasets from SemEval semantic textual similarity (STS) tasks. Our code is implemented in Matlab and we use the C Mex function for the computationally expensive components of Word Mover's Distance [1] (Rubner et al., 2000) and the freely available Word2Vec word embedding [2] which has pre-trained embeddings for 3 million words/phrases (from Google News) (Mikolov et al., 2013a) for text classification tasks.

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

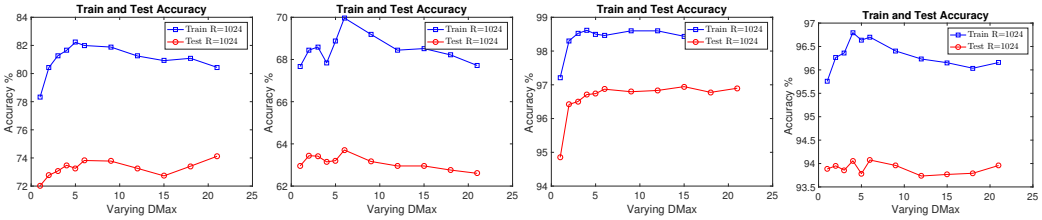
Table 1: Properties of the datasets

4.1 EFFECTS OF R AND D ON RANDOM FEATURES



(a) TWITTER (b) OHSUMED (c) CLASSIC (d) AMAZON

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



(a) TWITTER (b) OHSUMED (c) CLASSIC (d) AMAZON

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

[1] We adopt Rubner's C code from `http://ai.stanford.edu/~rubner/emd/default.htm`.

[2] We use word2vec code from `https://code.google.com/archive/p/word2vec/`.

Table 2: Testing 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.

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	$\mathbf{98.2 \pm 0.6}$	92	34	**122**
TWITTER	71.3 ± 0.6	25	4	72.5 ± 0.5	10	2	$\mathbf{74.5 \pm 0.5}$	162	34	**2**
RECIPE	57.4 ± 0.3	448	326	57.4 ± 0.5	18	4	$\mathbf{61.8 \pm 0.8}$	277	61	**82**
OHSUMED	55.5	3530	2807	55.8	24	7	**64.5**	757	240	**401**
CLASSIC	$\mathbf{97.2 \pm 0.1}$	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	$\mathbf{94.3 \pm 0.4}$	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	$\mathbf{79.2 \pm 0.3}$	1838	330	**103**

Setup. We first perform experiments to investigate the behavior of the WME method by varying the rank R and the length D of random documents. The hyper-parameter γ is obtained through cross validation within typical range $[0.01,\ 10]$. Due to limited space, we only show selected subsets of our results and more results are listed in Appendix 7.2.

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

Effects of D. We further evaluate the training and testing accuracy when varying the length of random document D with fixed R. As shown in Fig. 3, we can see that the near-peak performance can usually be achieved when D is small (typically $D \leq 6$). This behavior illustrates two important aspects: (i) using very few random words (like $D = 1$) is not enough to generate useful random features when R becomes large; 2) using too many random words (like $D \geq 10$) 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 contributes both performance boost and computational efficiency to the WME method.

4.2 Comparisons against KNN-WMD in both accuracy and runtime

Baselines. We now compare two WMD-based methods in terms of testing accuracy and total training and testing runtime. The most computationally expensive component is WMD for which both methods use same implementation from Rubner (Rubner et al., 2000). 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. We run the experiments on all 5 train/test splits but we only report average runtime since the variation of runtime is quite small.

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 of the methods over five 70/30 train/test splits. For all aforementioned methods, we adopted results from Kusner's original paper (Kusner et al., 2015) though we also rerun the experiments of all methods with KNN and found them consistent. For all methods, we perform 10-fold cross validation to search for the best parameters on 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. Bold face highlights the best number for each dataset. More results on comparisons against KNN-based methods refer to Appendix 7.3.

Results. Table 2 corroborates the significant advantages of WME compared to KNN-WMD in terms of both accuracy and runtime. First, WME(SR) can consistently achieve better or similar testing 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. The substantially improved accuracy of WME suggests that a truly p.d. kernel implicitly admits expressive feature representation of documents learned from the Word2Vec embedding space in which the alignments between words are considered by using WMD.

4.3 COMPARISONS AGAINST WORD2VEC AND DOC2VEC-BASED REPRESENTATIONS

Baselines. We compare against 6 document representations methods: 1) *Smooth Inverse Frequency* (SIF) (Arora et al., 2017): a newly 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 normalized BOW weights for generating document representation; 3) *Word2Vec+tf-idf*: a weighted average of word vectors using TF-IDF weights for generating document representations; 4) *PV-DBOW* (Le & Mikolov, 2014): distributed bag of words model of Paragraph Vector; 5) *PV-DM* (Le & Mikolov, 2014): distributed memory model of Paragraph Vector; 6) *Doc2VecC* (Chen, 2017): recently proposed document vector learning framework through corruption, representing each document as a simple average of sampled word embeddings that achieves state-of-the-art performance in document classification.

Setup. We remove RECIPE and keep RECIPE_L due to its large number of classes and documents for favoring neural network language models. *Word2Vec+nbow* and *Word2Vec+tf-idf* use pre-trained Word2Vec embeddings while SIF uses its default pretrained 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 indeed is beneficial (see Appendix 7.4) for generating a better document representation. For each method, we perform a grid search for some important parameters while using recommended parameters for others. For a fair comparison, we run a linear SVM using LIBLINEAR (Fan et al., 2008) on all methods due to the aforementioned reason.

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 $\pm$ 1.2	97.3 $\pm$ 0.9	96.9 $\pm$ 1.1	97.2 $\pm$ 0.7	97.9 $\pm$ 1.3	90.5 $\pm$ 1.7	**98.2 $\pm$ 0.6**
TWITTER	57.8 $\pm$ 2.5	72.0 $\pm$ 1.5	71.9 $\pm$ 0.7	67.8 $\pm$ 0.4	67.3 $\pm$ 0.3	71.0 $\pm$ 0.4	**74.5 $\pm$ 0.5**
OHSUMED	**67.1**	63.0	60.6	55.9	59.8	63.4	64.5
CLASSIC	92.7 $\pm$ 0.9	95.2 $\pm$ 0.4	93.9$\pm$ 0.4	97.0 $\pm$ 0.3	96.5 $\pm$ 0.7	96.6 $\pm$ 0.4	**97.1 $\pm$ 0.4**
REUTERS	87.6	96.9	95.9	96.3	94.9	96.5	**97.2**
AMAZON	94.1 $\pm$ 0.2	94.0 $\pm$ 0.5	92.2 $\pm$ 0.4	89.2 $\pm$ 0.3	88.6 $\pm$ 0.4	91.2 $\pm$ 0.5	**94.3 $\pm$ 0.4**
20NEWS	72.3	71.7	70.2	71.0	74.0	78.2	**78.3**
RECIPE_L	71.1 $\pm$ 0.5	74.9 $\pm$ 0.5	73.1 $\pm$ 0.6	73.1 $\pm$ 0.5	71.1 $\pm$ 0.4	76.1 $\pm$ 0.4	**79.2 $\pm$ 0.3**

Results. Table 3 shows that WME consistently outperforms or matches currently state-of-the-art document representation methods in terms of testing accuracy on all datasets except one (OHSUMED). The first highlight in this table is that simple average of word embeddings *Word2Vec+nbow* and *Word2Vec+tf-idf* often achieve better performance than *SIF(Glove)* which might indicate that removing first principle component (roughly corresponding to the syntactic information or common words) could hurt the expressive power of the resulting representation for some of classification tasks. Surprisingly, these two methods often achieves similar or better performance than *PV-DBOW* and *PV-DM*, which may be because of the high-quality word embeddings pre-trained from hundreds of billions of words (Mikolov et al., 2013a;c). On 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 where it can directly access to the features of the test set (See Appendix 7.4). Finally, the better performance of WME over three strong baselines *Word2Vec+nbow*, *Word2Vec+tf-idf*, and *SIF(Glove)* stems from fact that WME takes into account pair-wise word alignments that leads to a more informative representation. Compared to other Doc2Vec-based methods, WME is empowered by two important building blocks WMD and Word2Vec to ensure a semantically meaningful representation of the documents by considering both

the word alignments and the semantics of words. We refer the interested readers to more experimental results on Imdb dataset in Appendix 7.4.

4.4 COMPARISONS FOR PERFORMING TEXTUAL SIMILARITY TASKS

Baselines. We compare WME against 10 supervised 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) *GV+ave*: simple averaging of pretrained GloVe word vectors; 3) *GV+tf-idf*: a weighted average of GloVe word vecors using TF-IDF weights; 4) *SIF* (Arora et al., 2017): a state-of-the-art simple method for textual similarity tasks using GloVe.

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 the SemEval 2015 Twitter task (Marelli et al., 2014). Each year STS typically 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 7.5. Our method can be built on any high-quality word embeddings and combined with a good weighting scheme for WMD computation. To promote a fair comparison with other unsupervised methods, we also apply the same GloVe word embeddings with the tf-idf weighting scheme. Conceptually, the weighting scheme of SIF can also be easily adopted in our method.

Table 4: Pearson's scores of WME against other unsupervised and supervised methods on 22 textual similarity tasks. Results are collected from (Arora et al., 2017) except our approach and only average scores each year are reported. All unsupervised approaches are built on GloVe except ST.

Approaches	Supervised						Unsupervised				
Tasks	PP	Dan	RNN	iRNN	LSTM(no)	LSTM(o.g.)	ST	GV+ave	GV+tf-idf	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**
STS'13	55.8	54.2	44.7	**56.7**	45.2	41.5	24.8	42.3	52.1	56.6	54.5
STS'14	**70.9**	69.5	57.7	**70.9**	59.8	51.5	31.4	54.2	63.8	68.5	65.5
STS'15	**75.8**	72.7	57.2	75.6	63.9	56.0	31.0	52.7	60.6	71.7	61.8
SICK'14	71.6	70.7	61.2	71.2	63.9	59.0	49.8	65.9	69.4	**72.2**	68.0
Twitter'15	52.9	**53.7**	45.1	52.9	47.6	36.1	24.7	30.3	33.8	48.0	41.6

Results. Table 4 shows that WME achieves quite decent performance compared to other unsupervised and supervised methods although WME itself is also an unsupervised method. Indeed, compared with *ST* and *GV+ave*, 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. *GV+tf-idf* also improves over these two methods but 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. Clearly, removing dominant component from learned sentence representation is very helpful for this type of tasks. Interestingly, WME can outperform or match the scores of three supervised methods *RNN*, *LSTM(no)*, and *LSTM(o.g.)* in most of cases. There are no surprises that the supervised methods like *PP* and *iRNN* via fine tuning word embeddings (from PSL) with aid of the large external corpora PPDB would yield better performance, but still not always beat unsupervised methods (like WME and SIF). The final remarks stem from the fact that WME may also gain significantly benefit from the supervised word embeddings that Arora et al. (2017) have shown with *SIF* on PSL.

5 CONCLUSION

In this paper, we have proposed for the first time an alignment-aware text kernel using WMD for unstructured text data, which takes into account both word alignments and pre-trained high quality word embeddings in learning an effective semantic-preserving feature representation. The proposed WME is simple, efficient, flexible, and unsupervised. For instance, it can substantially

improve the classification accuracy while reducing the computational cost of WMD-based KNN from cubic to linear in document length and from quadratic to linear in number of samples. Extensive experiments show that WME consistently matches or outperforms state-of-the-art Word2Vec and Doc2Vec based models on 9 real-word text classification tasks, and some sophisticated supervised methods such as RNN and LSTM models and other unsupervised methods on 22 textual similarity tasks, respectively. The WME embeddings can be easily used for a wide range of downstream supervised and unsupervised tasks.

REFERENCES

Eneko Agirre, Mona Diab, Daniel Cer, and Aitor Gonzalez-Agirre. 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*, pp. 385–393. Association for Computational Linguistics, 2012.

Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. 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, 2013.

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

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. Semeval-2015 task 2: Semantic textual similarity, english, spanish and pilot on interpretability. In *SemEval@ NAACL-HLT*, pp. 252–263, 2015.

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

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

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

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

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

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

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

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

Andrew M Dai and Quoc V Le. Semi-supervised sequence learning. In *Advances in Neural Information Processing Systems*, pp. 3079–3087, 2015.

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

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

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

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

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

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

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

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pp. 1681–1691, 2015.

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

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

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

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

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

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Interspeech*, volume 2, pp. 3, 2010.

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

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

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

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

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

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

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

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

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

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

Richard Socher, Brody Huval, Christopher D Manning, and Andrew Y Ng. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pp. 1201–1211. Association for Computational Linguistics, 2012.

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

Sida Wang and Christopher D Manning. 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*, pp. 90–94. Association for Computational Linguistics, 2012.

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

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

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

Lingfei Wu, Eloy Romero, and Andreas Stathopoulos. Primme_svds: A high-performance preconditioned svd solver for accurate large-scale computations. *arXiv preprint arXiv:1607.01404*, 2016.

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

6 APPENDIX A: PROOF OF LEMMA 1 AND THEOREM 1

6.1 PROOF OF LEMMA 1

Proof. Firstly, we find an ϵ-covering $\mathcal{E}_W$ of size $(\frac{2}{\epsilon})^d$ for the word vector space $\mathcal{V}$. Then define $\mathcal{E}$ as all possible sets of $\boldsymbol{v} \in \mathcal{E}_W$ of size no larger than $L_{\max}$. We have $|\mathcal{E}| \leq (\frac{2}{\epsilon})^{dL_{\max}}$, and for any document $x = (\boldsymbol{v}_j)_{j=1}^L \in \mathcal{X}$, we can find $x_i \in \mathcal{E}$ with also L words $(\boldsymbol{u}_j)_{j=1}^L$ such that $\|\boldsymbol{u}_j - \boldsymbol{v}_j\| \leq \epsilon$. Then by the definition of WMD (1), a solution that assigns each word $\boldsymbol{v}_j$ in x to the word $\boldsymbol{u}_j$ in x_i would have overall cost less than ϵ, and therefore, $\text{WMD}(x, x_i) \leq \epsilon$. □

6.2 PROOF OF THEOREM 1

Proof. Let $s_R(x, y)$ be the random approximation (6). Our goal is to bound the magnitude of $\Delta_R(x, y) = s_R(x, y) - k(x, y)$. Since $E[\Delta_R(x, y)] = 0$ and $|\Delta_R(x, y)| \leq 1$, from Hoefding inequality, we have

$$P\left\{|\Delta_R(x, y)| \geq t\right\} \leq 2\exp(-Rt^2/2)$$

for a given pair of documents (x, y). To get a uniform bound that holds for $\forall(x, y) \in \mathcal{X} \times \mathcal{X}$, we find an ϵ-covering of $\mathcal{X}$ of finite size, given by Lemma 1. Applying union bound over the ϵ-covering $\mathcal{E}$ for x and y, we have

$$P\left\{\max_{x_i \in \mathcal{E}, y_j \in \mathcal{E}} |\Delta_R(x_i, y_j)| > t\right\} \leq 2|\mathcal{E}|^2 \exp(-Rt^2/2). \tag{7}$$

Then by the definition of $\mathcal{E}$ we have $|\text{WMD}(x, \omega) - \text{WMD}(x_i, \omega)| \leq \text{WMD}(x, x_i) \leq \epsilon$. Together with the fact that $\exp(-\gamma t)$ is Lipschitz-continuous with parameter γ for $t \geq 0$, we have

$$|\phi_\omega(x) - \phi_\omega(x_i)| \leq \gamma\epsilon$$

and thus

$$|s_R(x, y) - s_R(x_i, y_i)| \leq 3\gamma\epsilon, \quad |k(x, y) - k(x_i, y_i)| \leq 3\gamma\epsilon$$

for $\gamma\epsilon$ chosen to be ≤ 1. This gives us

$$|\Delta_R(x, y) - \Delta_R(x_i, y_i)| \leq 6\gamma\epsilon \tag{8}$$

Combining (7) and (8), we have

$$P\left\{\max_{x_i \in \mathcal{E}, y_j \in \mathcal{E}} |\Delta_R(x, y)| > t + 6\gamma\epsilon\right\} \leq 2\left(\frac{2}{\epsilon}\right)^{2dL_{\max}} \exp(-Rt^2/2). \tag{9}$$

Choosing $\epsilon = t/6\gamma$ yields the result. □

7 APPENDIX B: ADDITIONAL EXPERIMENTAL RESULTS AND DETAILS

7.1 EXPERIMENTAL SETTINGS AND PARAMETERS FOR WME

Setup. We choose 9 different document corpora where 8 of them are overlapped with datasets in (Kusner et al., 2015; Huang et al., 2016). A complete data summary is in Table 1. These datasets come from various applications, including news categorization, sentiment analysis, product identification, and have various number of classes, varying number of documents, and a wide range of document lengths. Our code is implemented in Matlab and we use the C Mex function for computationally expensive components of Word Mover's Distance [3] (Rubner et al., 2000) and the freely available Word2Vec word embedding [4] which has pre-trained embeddings for 3 millon words/phrases (from Google News) (Mikolov et al., 2013a). All computations were carried out on a DELL dual socket system with Intel Xeon processors 272 at 2.93GHz for a total of 16 cores and 250 GB of memory, running the SUSE Linux operating system. To accelerate the computation of WMD-based methods, we use multithreading with total 12 threads for WME and KNN-WMD in all experiments. For all experiments, we generate random document from uniform distribution with mean centered in

[3] We adopt Rubner's C code from http://ai.stanford.edu/ rubner/emd/default.htm.

[4] We use word2vec code from https://code.google.com/archive/p/word2vec/.

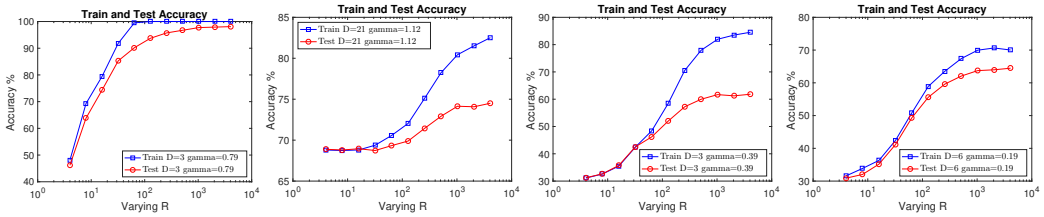
Word2Vec embedding space since we observe the best performance with this setting. We perform 10-fold cross-validation to search for best parameters for γ and D_{max} as well as parameter C for LIBLINEAR on training set for each dataset. We simply fix the $D_{min} = 1$, and vary D_{max} in the range of 3 to 21, γ in the range of [1e-2 3e-2 0.10 0.14 0.19 0.28 0.39 0.56 0.79 1.0 1.12 1.58 2.23 3.16 4.46 6.30 8.91 10], and C in the range of [1e-5 1e-4 1e-3 1e-2 1e-1 1 1e1 1e2 3e2 5e2 8e2 1e3 3e3 5e3 8e3 1e4 3e4 5e4 8e4 1e5 3e5 5e5 8e5 1e6 1e7 1e8] respectively in all experiments.

We collect all document corpora from these public websites:

1. BBCSPORT: `http://mlg.ucd.ie/datasets/bbc.html`
2. TWITTER: `http://www.sananalytics.com/lab/twitter-sentiment/`
3. RECIPE: `https://www.kaggle.com/kaggle/recipe-ingredients-dataset`
4. OHSUMED:`https://www.mat.unical.it/OlexSuite/Datasets/SampleDataSets-download.htm`
5. CLASSIC:`http://www.dataminingresearch.com/index.php/2010/09/classic3-classic4-datasets/`
6. REUTERS and 20NEWS: `http://www.cs.umb.edu/~smimarog/textmining/datasets/`
7. AMAZON: `https://www.cs.jhu.edu/~mdredze/datasets/sentiment/`

7.2 More results about effects of R and D on random documents

Setup and results. To fully study the characteristic of the WME method, we study the effect of the R number of random documents and the D length of random documents on the performance of various datasets in terms of training and testing accuracy. Clearly, the training and testing accuracy can converge rapidly to the exact kernels when varying R from 4 to 4096, which confirms our analysis in Theory 1. When varying D from 1 to 21, we can see that in most of cases $D_{max} = [3,\ 12]$ generally yields a near-peak performance except BBCSPORT.



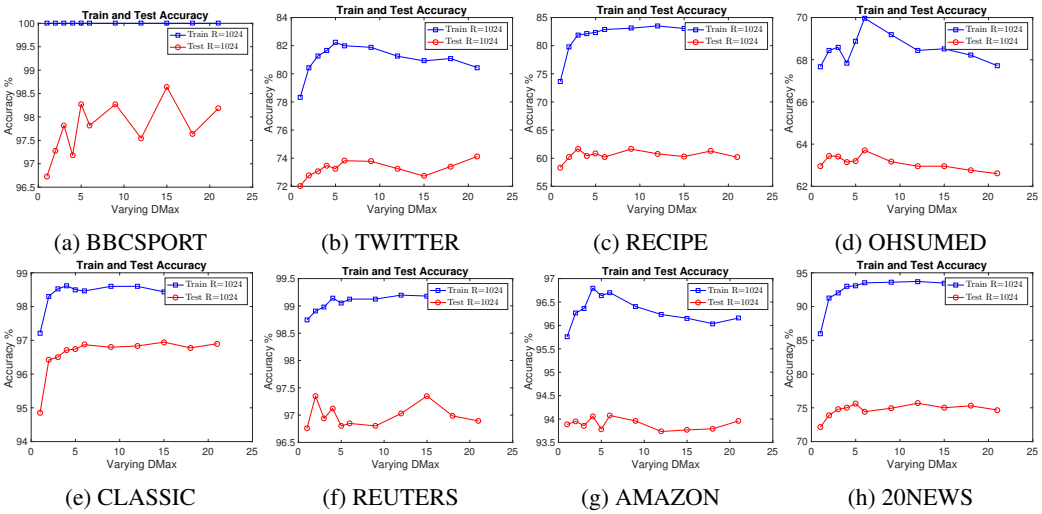
(a) BBCSPORT (b) TWITTER (c) RECIPE (d) OHSUMED

(e) CLASSIC (f) REUTERS (g) AMAZON (h) 20NEWS

Figure 4: Train (Blue) and test (Red) accuracy when varying R with fixed D.

7.3 More results on Comparisons against distance-based methods

Setup. We preprocess all datasets by removing all words in the SMART stop word list (Buckley et al., 1995). For 20NEWS, we remove the words appearing less than 5 times. For LDA, we use the Matlab Topic Modeling Toolbox (Griffiths & Steyvers, 2007) and use sample code that first run 100 burn-in iterations and then run the chain for additional 1000 iterations. For mSDA, we use high-dimensional function mSDAhd where the parameter dd is set as 0.2 times BOW Dimension. For all datasets, a



(a) BBCSPORT (b) TWITTER (c) RECIPE (d) OHSUMED

(e) CLASSIC (f) REUTERS (g) AMAZON (h) 20NEWS

Figure 5: Train (Blue) and test (Red) accuracy when varying D with fixed R.

Table 5: Testing accuracy comparing WME against KNN-based methods

Dataset	BOW	TF-IDF	BM25	LSI	LDA	mSDA	KNN-WMD	WME
BBCSPORT	79.4 ± 1.2	78.5 ± 2.8	83.1 ± 1.5	95.7 ± 0.6	93.6 ± 0.7	91.6 ± 0.8	95.4 ± 0.7	**98.2 ± 0.6**
TWITTER	56.4 ± 0.4	66.8 ± 0.9	57.3 ± 7.8	68.3 ± 0.7	66.2 ± 0.7	67.7 ± 0.7	71.3 ± 0.6	**74.5 ± 0.5**
RECIPE	40.7 ± 1.0	46.4 ± 1.0	46.4 ± 1.9	54.6 ± 0.5	48.7 ± 0.6	52 ± 1.4	57.4 ± 0.3	**61.8 ± 0.8**
OHSUMED	38.9	37.3	33.8	55.8	49.0	50.7	55.5	**64.5**
CLASSIC	64.0 ± 0.5	65.0 ± 1.8	59.4 ± 2.7	93.3 ± 0.4	95.0 ± 0.3	93.1 ± 0.4	**97.2 ± 0.1**	97.1 ± 0.4
REUTERS	86.1	70.9	67.2	93.7	93.1	91.9	96.5	**97.2**
AMAZON	71.5 ± 0.5	58.5 ± 1.2	41.2 ± 2.6	90.7 ± 0.4	88.2 ± 0.6	82.9 ± 0.4	92.6 ± 0.3	**94.3 ± 0.4**
20NEWS	42.2	45.6	44.1	71.1	68.5	60.5	73.2	**78.3**

5-fold cross validation on training set is performed to get the optimal K for KNN classifier, where K is searched in the range of [1, 21].

Baselines. We compare against 7 document representation or distance methods: 1) *bag-of-words* (BOW) (Salton & Buckley, 1988); 2) *term frequency-inverse document frequency* (TF-IDF) (Robertson & Walker, 1994); 3) *Okapi BM25* (Robertson et al., 1995): first TF-IDF variant ranking function used in search engines; 4) *Latent Semantic Indexing* (LSI) (Deerwester et al., 1990): factorize BOW into their leading singular components subspace using SVD (Wu & Stathopoulos, 2015; Wu et al., 2016); 5) *Latent Dirichlet Allocation* (LDA) (Blei et al., 2003): a generative probability method to model mixtures of word "topics" in documents. LDA is trained *transductively* on both train and test; 6) *Marginalized Stacked Denoising Autoencoders* (mSDA) (Chen et al., 2012): a fast method for training denoising autoencoder that achieved state-of-the-art performance on sentiment analysis tasks (Glorot et al., 2011); 7) *WMD*: a state-of-the-art document distance discussed in Section 2.

Results. Table 5 clearly demonstrates the superior performance of our method WME compared to other KNN-based methods in terms of testing accuracy. Indeed, BOW and TF-IDF performs poorly compared to other methods which may be the result of frequent near-orthogonality of their high-dimensional sparse feature representation in KNN classifier. KNN-WMD achieves noticeably better testing accuracy than LSI, LDA and mSDA since WMD takes into account the word alignments and leverages the power of Word2Vec. Remarkably, our proposed method WME achieves much higher accuracy compared to other methods including KNN-WMD on all datasets except one (CLASSIC).

7.4 More results on comparisons against Word2Vec and Doc2Vec-based document representations

Setup and results. For *PV-DBOW*, *PV-DM*, and *Doc2VecC*, we set the word and document vector dimension $d = 300$ to match the pre-trained word embeddings we used for WME and other Word2Vec-based methods in order to make a fair comparison. For other parameters, we use recommended

parameters in the papers but we search for the best parameter C in LIBLINEAR for these methods. Additionally, we also train *Doc2VecC* with different corruption rate in the range of [0.1 0.3 0.5 0.7 0.9]. Following (Chen, 2017), these methods are trained transductively on both training and testing set. For *Doc2VecC(Train)*, we train the model only on training set in order to show the effect of the transductive training on the testing accuracy. As shown in Table 6, *Doc2VecC* clearly outperforms *Doc2VecC(Train)*, sometimes having a significant performance boost on some datasets (OHSUMED and 20NEWS).

Table 6: Testing accuracy of WME against Word2Vec and Doc2Vec-based methods.

Dataset	Word2Vec+nbow	Word2Vec+tf-idf	PV-DBOW	PV-DM	Doc2VecC(Train)	Doc2VecC	WME
BBCSPORT	97.3 $\pm$ 0.9	96.9 $\pm$ 1.1	97.2 $\pm$ 0.7	97.9 $\pm$ 1.3	89.2 $\pm$ 1.4	90.5 $\pm$ 1.7	**98.2 $\pm$ 0.6**
TWITTER	72.0 $\pm$ 1.5	71.9 $\pm$ 0.7	67.8 $\pm$ 0.4	67.3 $\pm$ 0.3	69.8 $\pm$ 0.9	71.0 $\pm$ 0.4	**74.5 $\pm$ 0.5**
OHSUMED	63.0	60.6	55.9	59.8	59.6	63.4	**64.5**
CLASSIC	95.2 $\pm$ 0.4	93.9$\pm$ 0.4	97.0 $\pm$ 0.3	96.5 $\pm$ 0.7	96.2 $\pm$ 0.5	96.6 $\pm$ 0.4	**97.1 $\pm$ 0.4**
REUTERS	96.9	95.9	96.3	94.9	96.0	96.5	**97.2**
AMAZON	94.0 $\pm$ 0.5	92.2 $\pm$ 0.4	89.2 $\pm$ 0.3	88.6 $\pm$ 0.4	89.5 $\pm$ 0.4	91.2 $\pm$ 0.5	**94.3 $\pm$ 0.4**
20NEWS	71.7	70.2	71.0	74.0	72.9	78.2	**78.3**
RECIPE_L	74.9 $\pm$ 0.5	73.1 $\pm$ 0.6	73.1 $\pm$ 0.5	71.1 $\pm$ 0.4	75.6 $\pm$ 0.4	76.1 $\pm$ 0.4	**79.2 $\pm$ 0.3**

We further conduct experiments on Imdb dataset using our method. We use only training data to select hyper-parameters. For a more fair comparison, we only report the results of other methods that use all data excluding test. Table 7 shows that WME can achieve slightly better accuracy than other state-of-the-art document representation methods. This collaborates the importance to make full use of both word alignments and high-quality pretrained word embeddings.

Table 7: Testing accuracy of WME against other document representations on Imdb dataset (50K). Results are collected from (Chen, 2017) and (Arora et al., 2017).

Dataset	RNN_LM	SIF(GloVe)	Word2Vec+AVG	Word2Vec+IDF	PV-DBOW	ST	Doc2VecC	WME
Imdb	86.4	85.0	87.3	88.1	87.9	82.6	88.3	**88.5**

7.5 More results on comparisons for textual similarity tasks

Setup and results. To obtain the hyper-parameters in our method, we use the corresponding training data or the similar tasks from previous years. Note that the tasks with same names but in different years are different ones. As we can see in Table 7.5, WME can achieve better performance on tasks of STS'12 and perform fairly well on other tasks. Among the unsupervised methods and some supervised methods except *PP*, *Dan*, and *iRNN*, WME is almost always to be one of the best methods.

Table 8: Pearson's scores of WME against other unsupervised and supervised methods on 22 textual similarity tasks. Results are collected from (Arora et al., 2017) except our approach. All unsupervised approaches are built on GloVe except ST.

Approaches	Supervised						Unsupervised				
Tasks	PP	Dan	RNN	iRNN	LSTM(no)	LSTM(o.g.)	ST	Ave	Tf-idf	SIF	WME
MSRpar	42.6	40.3	18.6	43.4	16.1	9.3	16.8	47.7	**50.3**	35.6	45.3
MSRvid	74.5	70.0	66.5	73.4	71.3	71.3	41.7	63.9	77.9	**83.8**	75.9
SMT-eur	47.3	43.8	40.9	47.1	41.8	44.3	35.2	46.0	54.7	49.9	**57.7**
OnWN	**70.6**	65.9	63.1	70.1	65.2	56.4	29.7	55.1	64.7	66.2	67.8
SMT-news	58.4	**60.0**	51.3	58.1	60.8	51.0	30.8	49.6	45.7	45.6	56.1
STS'12	58.7	56.0	48.1	58.4	51.0	46.4	30.8	52.5	58.7	56.2	**60.6**
headline	72.4	71.2	59.5	**72.8**	57.4	48.5	34.6	63.8	69.2	69.2	70.5
OnWN	67.7	64.1	54.6	69.4	68.5	50.4	10.0	49.0	72.9	**82.8**	80.1
FNWN	43.9	43.1	30.9	**45.3**	24.7	38.4	30.4	34.2	36.6	39.4	33.7
SMT	39.2	38.3	33.8	**39.4**	30.1	28.8	24.3	22.3	29.6	37.9	33.7
STS'13	55.8	54.2	44.7	**56.7**	45.2	41.5	24.8	42.3	52.1	56.6	54.5
deft forum	48.7	**49.0**	41.5	**49.0**	44.2	46.1	12.9	27.1	37.5	41.2	41.2
deft news	**73.1**	71.7	53.7	72.4	52.8	39.1	23.5	68.0	68.7	69.4	66.7
headline	**69.7**	69.2	57.5	70.2	57.5	50.9	37.8	59.5	63.7	64.7	65.6
images	78.5	76.9	67.6	78.2	68.5	62.9	51.2	61.0	72.5	**82.6**	69.2
OnWN	78.8	75.7	67.7	78.8	76.9	61.7	23.3	58.4	75.2	82.8	81.1
tweet news	76.4	74.2	58.0	**76.9**	58.7	48.2	39.9	51.2	65.1	70.1	68.9
STS'14	**70.9**	69.5	57.7	**70.9**	59.8	51.5	31.4	54.2	63.8	68.5	65.5
answers-forum	**68.3**	62.6	32.8	67.4	51.9	50.7	36.1	30.5	45.6	63.9	56.4
answers-student	**78.2**	78.1	64.7	**78.2**	71.5	55.7	33.0	63.0	63.9	70.4	63.1
belief	**76.2**	72.0	51.9	75.9	61.7	52.6	24.6	40.5	49.5	71.8	50.6
headline	74.8	73.5	65.3	75.1	64.0	56.6	43.6	61.8	70.9	70.7	70.8
images	81.4	77.5	71.4	81.1	70.4	64.2	17.7	67.5	72.9	**81.5**	67.9
STS'15	**75.8**	72.7	57.2	75.6	63.9	56.0	31.0	52.7	60.6	71.7	61.8
SICK'14	71.6	70.7	61.2	71.2	63.9	59.0	49.8	65.9	69.4	**72.2**	68.0
Twitter'15	52.9	**53.7**	45.1	52.9	47.6	36.1	24.7	30.3	33.8	48.0	41.6