Supervised Word Mover's Distance

Anonymous Author(s)
Affiliation
Address
email

Abstract

Accurately measuring the similarity between text documents lies at the core of many real world applications of machine learning. These include web-search ranking, document recommendation, multi-lingual document matching, and article categorization. Recently, a new document metric, the word mover's distance (WMD), has been proposed with unprecedented results on kNN-based document classification. The WMD elevates high quality word embeddings to document metrics by formulating the distance between two documents as an optimal transport problem between the embedded words. However, the document distances are entirely unsupervised and lack a mechanism to incorporate supervision when available. In this paper we propose an efficient technique to learn a supervised metric, which we call the *Supervised WMD (S-WMD)* metric. Our algorithm learns document distances that measure the underlying semantic differences between documents by leveraging semantic differences between individual words discovered during supervised training. This is achieved with an linear transformation of the underlying word embedding space and tailored word-specific weights, learned to minimize the stochastic leave-one-out nearest neighbor classification error on a per-document level. We evaluate our metric on eight real-world text classification tasks on which S-WMD consistently outperforms almost all of our *26* competitive baselines.

1 Introduction

Document distances are a key component of many text retrieval tasks such as web-search ranking [24], book recommendation [16], and news categorization [25]. Because of the variety of potential applications, there has been a wealth of work towards developing accurate document distances [2, 4, 11, 27]. In large part, prior work has focused on extracting meaningful document representations, starting with the classical bag of words (BOW) and term frequency-inverse document frequency (TF-IDF) representations [30]. These sparse, high-dimensional representations are frequently nearly orthogonal [17] and a pair of similar documents may therefore have the nearly the same distance as a pair that are very different. It is possible to design more meaningful representations through eigendecomposing the BOW space with Latent Semantic Indexing (LSI) [11], or learning a probabilistic clustering of BOW vectors with Latent Dirichlet Allocation (LDA) [2]. Other work generalizes LDA [27] or uses denoising autoencoders [4] to learn a suitable document representation.

Recently, Kusner et al. [19] proposed the Word Mover's Distance (WMD), a new distance for text documents that leverages word2vec term embeddings [22]. Word2vec constitutes a breakthrough in learning word embeddings and can be trained from billions of words. The WMD uses such high-quality word representations to define document distances. It defines the distances between two documents as the optimal transport cost of moving all words from one document to another within the word embedding space. This approach was shown to lead to state-of-the-art error rates in k-nearest neighbor (kNN) document classification. Importantly, however, these prior works are

Submitted to 29th Conference on Neural Information Processing Systems (NIPS 2016). Do not distribute.

entirely *unsupervised* and not learned explicitly for any particular task. For example, a set of text documents could be classified by *topic* or by *author*, which would lead to very different definitions of dissimilarity. Lately, there has been a vast amount of work on metric learning [10, 15, 37, 38], most of which focuses on learning a generalized linear Euclidean metric. Most of these methods scale quadratically with the input dimensionality, and can only be applied to high-dimensional text documents after dimensionality reduction techniques such as PCA [37].

In this paper we propose an algorithm for learning a metric to improve the word mover's distance. WMD stands out from prior work in that it computes distances between documents without ever learning a new document representation. Instead, it leverages low-dimensional word representations, for example word2vec, to compute distances. This allows us to transform the word embedding instead of the documents, and remain in a low-dimensional space throughout. At the same time we propose to learn word-specific weights, to emphasize the importance of certain words for distinguishing the document class.

At first glance, incorporating supervision into the WMD appears computationally prohibitive, as each individual WMD computation scales cubically in the size of the documents. However, we devise an efficient technique that exploits a relaxed version of the underlying optimal transport problem, called the Sinkhorn distance [6]. This, combined with a probabilistic filtering of the training set, reduces the computation time significantly.

Our metric learning algorithm, *Supervised Word Mover's Distance (S-WMD)*, directly minimizes a stochastic version of the leave-one-out classification error under the WMD metric. Different from classic metric learning, we learn a linear transformation of the *word representations* while also learning re-weighted word frequencies. These transformations are learned to make the WMD distances match the semantic meaning of similarity encoded in the labels. We show across 8 datasets and 26 baseline methods the superiority of our method.

2 Background

Here we describe the initial word embedding technique we use (word2vec) and the recently introduced word mover's distance. We then detail the general setting of linear metric learning and give specific details on NCA that we will make use of in the model.

word2vec is a new technique for learning a word embedding over billions of words and was introduced by Mikolov et al. [22]. Each word in the training corpus is associated with an initial word vector, which is then optimized so that if two words w_1 and w_2 frequently occur together they have high conditional probability $p(w_2|w_1)$. This probability is the hierarchical softmax of the word vectors $\mathbf{v}_{w_1}$ and $\mathbf{v}_{w_2}$ [22], an easily-computed quantity which allows a simplified neural language model (the word2vec model) to be trained efficiently on desktop computers. Training an embedding over billions of words allows word2vec to capture surprisingly accurate word relationships [23]. Word embeddings can learn hundreds of millions of parameters and are typically by design unsupervised, allowing them to be trained on large unlabeled text corpora ahead of time. In this paper we will use word2vec, although in principle any initial word embedding can be used [21, 23, 5].

Word Mover's Distance. Leveraging the compelling word vector relationships of the word2vec embedding, Kusner et al. [19] introduced the *word mover's distance* (WMD) as a distance between text documents. At a high level, the WMD is the minimum distance required to move the words from one document to another. We assume that we are given a word2vec embedding matrix $\mathbf{X} \in \mathbb{R}^{d\times n}$ for a vocabulary of n words. Let $\mathbf{x}_i \in \mathcal{R}^d$ be the representation of the i^{th} word, as defined by this embedding. Additionally, let $\mathbf{d}^a, \mathbf{d}^b$ be the n-dimensional normalized bag-of-words (BOW) vectors for two documents, where d_i^a is the number of times word i occurs in $\mathbf{d}^a$ (normalized over all words in $\mathbf{d}^a$). The WMD introduces an auxiliary 'transport' matrix $\mathbf{T} \in \mathcal{R}^{n\times n}$, such that $\mathbf{T}_{ij}$ describes how much of d_i^a should be transported to d_j^b. Formally, the WMD learns $\mathbf{T}$ to minimize the objective function

$$D(\mathbf{x}_i, \mathbf{x}_j) = \min_{\mathbf{T}\geq 0} \sum_{i,j=1}^{n} \mathbf{T}_{ij}\|\mathbf{x}_i - \mathbf{x}_j\|_2, \quad \text{subject to,} \quad \sum_{j=1}^{n} \mathbf{T}_{ij} \ = d_i^a, \ \sum_{i=1}^{n} \mathbf{T}_{ij} \ = d_j^b \ \ \forall i, j. \quad (1)$$

In this way, documents that share many words (or even related ones) should have smaller distances than documents with very dissimilar words. It was noted in Kusner et al. [19] that the WMD is a special case of the Earth Mover's Distance (EMD) [29], also known more generally as the 1-Wasserstein distance [20]. The authors also introduce the *word centroid distance* (WCD), which uses a fast approximation first described by Rubner et al. [29]: $\|\mathbf{X}\mathbf{d} - \mathbf{X}\mathbf{d}'\|_2$. It can be shown that the WCD always lower bounds the WMD. Intuitively the WCD represents each document by the weighted average word vector, where the weights are the normalized BOW counts. The time complexity of solving the WMD optimization problem is $O(p^3 \log p)$ [26], where p is the maximum number of unique words in either $\mathbf{d}$ or $\mathbf{d}'$. The WCD scales asymptotically by $O(dp)$.

Regularized Transport Problem. To alleviate the cubic time complexity of the WMD, Cuturi & Doucet [8] formulated a smoothed version of the underlying transport problem by adding an entropy regularizer to the transport objective. This makes the objective function strictly convex, and efficient algorithms can be adopted to solve it. In particular, given a transport matrix $\mathbf{T}$, let $h(\mathbf{T}) = -\sum_{i,j=1}^{n} \mathbf{T}_{ij} \log(\mathbf{T}_{ij})$ be the entropy of $\mathbf{T}$. For any $\lambda > 0$, the regularized (primal) transport problem is defined as

$$\min_{\mathbf{T} \geq 0} \sum_{i,j=1}^{n} \mathbf{T}_{ij} \|\mathbf{x}_i - \mathbf{x}_j\|_2 - \frac{1}{\lambda} h(\mathbf{T}) \quad \text{subject to,} \quad \sum_{j=1}^{n} \mathbf{T}_{ij} = d_i^a, \ \sum_{i=1}^{n} \mathbf{T}_{ij} = d_j^b \ \ \forall i, j. \tag{2}$$

Linear Metric Learning. Assume that we have access to a training set $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \subset \mathbb{R}^d$, arranged as columns in matrix $\mathbf{X} \in \mathbb{R}^{d \times n}$, and corresponding labels $\{y_1, \ldots, y_n\} \subseteq \mathcal{Y}^n$, where $\mathcal{Y}$ contains some finite number of classes $C = |\mathcal{Y}|$. Linear metric learning learns a matrix $\mathbf{A} \in \mathbb{R}^{r \times d}$, where $r \leq d$, and defines the generalized Euclidean distance between two documents $\mathbf{x}_i$ and $\mathbf{x}_j$ as $d_{\mathbf{A}}(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{A}(\mathbf{x}_i - \mathbf{x}_j)\|_2$. Popular linear metric learning algorithms are NCA [15], LMNN [37], and ITML [10] amongst others [38]. All of these methods learn a matrix $\mathbf{A}$ to minimize a loss function that is often an approximation of the leave-one-out (LOO) classification error of the kNN classifier.

Neighborhood Components Analysis (NCA) was introduced by Goldberger et al. [15] to learn a generalized Euclidean metric. The authors address the problem that the leave-one-out kNN error is non-continuous by defining a stochastic neighborhood process. An input $\mathbf{x}_i$ is assigned input $\mathbf{x}_j$ as its nearest neighbor with probability

$$p_{ij} = \frac{\exp(-d^2_{\mathbf{A}}(\mathbf{x}_i, \mathbf{x}_j))}{\sum_{k \neq i} \exp\left(-d^2_{\mathbf{A}}(\mathbf{x}_i, \mathbf{x}_k)\right)}, \tag{3}$$

where we define $p_{ii} = 0$. NCA optimizes this metric explicitly for kNN. Under this rule, an input $\mathbf{x}_i$ with label y_i is classified correctly if its nearest neighbor is any $\mathbf{x}_j \neq \mathbf{x}_i$ from the same class $(y_j = y_i)$. The probability of this event can be stated as

$$p_i = \sum\nolimits_{j \neq i : y_j = y_i} p_{ij}. \tag{4}$$

NCA learns $\mathbf{A}$ by maximizing the expected LOO accuracy $\sum_i p_i$, or equivalently by minimizing $-\sum_i \log(p_i)$, the KL-divergence from a perfect classification distribution ($p_i = 1$ for all $\mathbf{x}_i$).

3 Learning a Word Embedding Metric

In this section we propose a method for learning a document distance, by way of learning a generalized Euclidean metric within the word embedding space. We will refer to the learned document distance metric as the *Supervised Word Mover's Distance (S-WMD)*. To learn such a metric we assume we have a training dataset consisting of m documents $\{\mathbf{d}^1, \ldots, \mathbf{d}^m\} \subset \Sigma_n$, where Σ_n is the $(n-1)$-dimensional simplex (thus each document is represented as a histogram over the words in the vocabulary, of size n). For each document we have a label $\{y_1, \ldots, y_m\} \subseteq \mathcal{Y}^m$, out of a possible C classes. Additionally, we are given a word embedding matrix $\mathbf{X} \in \mathbb{R}^{d \times n}$ (e.g., the word2vec embedding) which defines a d-dimensional word vector for each of the words in the vocabulary.

Supervised WMD. As described in the previous section, it is possible to define a distance between any two documents $\mathbf{d}^a$ and $\mathbf{d}^b$ as the minimum cumulative word distance of moving $\mathbf{d}^a$ to $\mathbf{d}^b$ in word embedding space, as is done in the WMD, eq. (1). Given document labels, we would like to learn this distance so that documents with the same labels are close, and otherwise are far apart, via a linear transformation $\mathbf{x}_i \rightarrow \mathbf{A}\mathbf{x}_i$. We also introduce a histogram importance vector $\mathbf{w}$ that re-weights the histogram values to reflect the importance of words for distinguishing the classes:

$$\tilde{\mathbf{d}}^a = (\mathbf{w} \circ \mathbf{d}^a)/(\mathbf{w}^\top \mathbf{d}^a), \tag{5}$$

where "$\circ$" denotes the Hadamard product.

After applying the vector $\mathbf{w}$ and the linear mapping $\mathbf{A}$, the WMD distance between documents $\mathbf{d}^a$ and $\mathbf{d}^b$ becomes

$$D_{\mathbf{A},\mathbf{w}}(\mathbf{d}^a,\mathbf{d}^b) \triangleq \min_{\mathbf{T}\geq 0} \sum_{i,j=1}^{n} \mathbf{T}_{ij}\|\mathbf{A}(\mathbf{x}_i-\mathbf{x}_j)\|_2^2 \;\text{ s.t. } \sum_{j=1}^{n}\mathbf{T}_{ij}=\tilde{d}_i^a \;\text{ and } \sum_{i=1}^{n}\mathbf{T}_{ij}=\tilde{d}_j^b \quad \forall i,j. \tag{6}$$

Loss Function. Our goal is to learn the matrix $\mathbf{A}$ and vector $\mathbf{w}$ to make the distance $D_{\mathbf{A},\mathbf{w}}$ reflect the semantic definition of similarity encoded in the labeled data. Similar to prior work on metric learning [15, 10, 37] we achieve this by minimizing the kNN LOO error with the distance $D_{\mathbf{A},\mathbf{w}}$ in the document space. As the LOO error is non-differentiable, we use the stochastic neighborhood relaxation proposed by Hinton & Roweis [18], which is also used for NCA.

Similar to prior work, we use the squared Euclidean word distance in Eq. (6) as opposed to the non-squared distance in WMD, Eq. (1). We use the KL-divergence loss proposed in NCA with (3) and (4) and obtain

$$\ell(\mathbf{A},\mathbf{w}) = -\sum_{a=1}^{m}\log\left(\sum_{b:y_b=y_a}^{m}\frac{\exp(-D_{\mathbf{A},\mathbf{w}}(\mathbf{d}_a,\mathbf{d}_b))}{\sum_{k\neq a}\exp\left(-D_{\mathbf{A},\mathbf{w}}(\mathbf{d}_a,\mathbf{d}_k)\right)}\right). \tag{7}$$

Gradient. Note that the loss function $\ell(\mathbf{A},\mathbf{w})$ contains the nested linear program defined in (6). We can compute the gradient with respect to $\mathbf{A}$ and $\mathbf{w}$ as follows,

$$\frac{\partial}{\partial(\mathbf{A},\mathbf{w})}\ell(\mathbf{A},\mathbf{w}) = \sum_{a=1}^{m}\sum_{b\neq a}\frac{p_{ab}}{p_a}(\delta_{ab}-p_a)\frac{\partial}{\partial(\mathbf{A},\mathbf{w})}D_{\mathbf{A},\mathbf{w}}(\mathbf{d}^a,\mathbf{d}^b), \tag{8}$$

where $\delta_{ab}=1$ if and only if $y_a=y_b$, and $\delta_{ab}=0$ otherwise. The remaining gradient can be computed based on prior work by Bertsimas & Tsitsiklis [1], Cuturi & Avis [7] and Cuturi & Doucet [8], who consider the differentiability of transportation problems.

Gradient w.r.t. A. The authors show that because the optimization in eq. (6) is a linear program, the gradient of $D_{\mathbf{A},\mathbf{w}}(\mathbf{d}^a,\mathbf{d}^b)$ with respect to $\mathbf{A}$ is

$$\frac{\partial}{\partial\mathbf{A}}D_{\mathbf{A},\mathbf{w}}(\mathbf{d}^a,\mathbf{d}^b) = 2\mathbf{A}\sum_{i,j=1}^{n}\mathbf{T}_{ij}^{ab}(\mathbf{x}_i-\mathbf{x}_j)(\mathbf{x}_i-\mathbf{x}_j)^\top, \tag{9}$$

where $\mathbf{T}^{ab}$ is the optimizer of (6), so long as it is unique. Even if $\mathbf{T}^{ab}$ is not unique, they show that the above expression (9) is in the sub-differential $\partial D_{\mathbf{A}}(\mathbf{d}^a,\mathbf{d}^b)$.

Gradient w.r.t. w. To obtain the gradient of the WMD distance with respect to $\mathbf{w}$, we need the optimal solution to the dual transport problem:

$$D^*_{\mathbf{A},\mathbf{w}}(\mathbf{d}^a,\mathbf{d}^b) \triangleq \max_{(\boldsymbol{\alpha},\boldsymbol{\beta})} \boldsymbol{\alpha}^\top\tilde{\mathbf{d}}^a+\boldsymbol{\beta}^\top\tilde{\mathbf{d}}^b;\; \text{s.t. } \alpha_i+\beta_j \leq \|\mathbf{A}(\mathbf{x}_i-\mathbf{x}_j)\|_2^2 \quad \forall i,j. \tag{10}$$

Cuturi & Doucet [8] points out that any optimal dual solution $\boldsymbol{\alpha}^*$ and $\boldsymbol{\beta}^*$ to (10) are subgradients of the primal WMD with respect to $\tilde{\mathbf{d}}^a$ and $\tilde{\mathbf{d}}^b$ respectively. Given that both $\tilde{\mathbf{d}}^a$ and $\tilde{\mathbf{d}}^b$ are functions of $\mathbf{w}$, we have that

$$\frac{\partial}{\partial\mathbf{w}}D_{\mathbf{A},\mathbf{w}}(\mathbf{d}^a,\mathbf{d}^b) = \frac{\partial D_{\mathbf{A},\mathbf{w}}}{\partial\tilde{\mathbf{d}}^a}\frac{\partial\tilde{\mathbf{d}}^a}{\partial\mathbf{w}}+\frac{\partial D_{\mathbf{A},\mathbf{w}}}{\partial\tilde{\mathbf{d}}^b}\frac{\partial\tilde{\mathbf{d}}^b}{\partial\mathbf{w}} = \frac{\boldsymbol{\alpha}\circ\mathbf{d}^a-(\boldsymbol{\alpha}^\top\tilde{\mathbf{d}}^a)\mathbf{d}^a}{\mathbf{w}^\top\mathbf{d}^a}+\frac{\boldsymbol{\beta}\circ\mathbf{d}^b-(\boldsymbol{\beta}^\top\tilde{\mathbf{d}}^b)\mathbf{d}^b}{\mathbf{w}^\top\mathbf{d}^b}. \tag{11}$$

3.1 Fast gradient computation

The above subgradient descent procedure is prohibitively slow in all but the most simple cases. Indeed, at each iteration we have to solve the dual transport problem for each pair of documents, which has a time complexity of $O(p^3\log p)$. Motivated by the recent works on fast Wasserstein distance computation [6, 8, 12], we propose to relax the modified linear program in eq. (6) by subtracting an entropy regularization term, as proposed in eq. (2).

This relaxed optimization problem can be shown to be strongly convex, thus admitting a unique solution $\mathbf{T}^*_\lambda$. More importantly, [6] gives an efficient algorithm to solve for both the primal variable

$\mathbf{T}^*_\lambda$ and the dual variables $(\boldsymbol{\alpha}^*_\lambda, \boldsymbol{\beta}^*_\lambda)$ using a clever matrix-scaling algorithm. Using this technique, we define the matrix $\mathbf{K}_{ij} = \exp(-\lambda\|\mathbf{x}_i - \mathbf{x}_j\|_2)$ and alternately solve for the scaling vectors $\mathbf{u}, \mathbf{v}$ to a fixed-point via the mapping $(\mathbf{u}, \mathbf{v}) \mapsto (\mathbf{d}^a/(\mathbf{K}\mathbf{v}), \mathbf{d}^b/(\mathbf{K}^\top\mathbf{u}))$. This yields the relaxed transport

$$\mathbf{T}^*_\lambda = \mathrm{diag}(\mathbf{u})\mathbf{K}\,\mathrm{diag}(\mathbf{v})$$

This algorithm can be shown to have empirical time complexity $O(p^2)$ [6], which is significantly faster than solving the WMD problem exactly. Once we have solved $\mathbf{u}$ and $\mathbf{v}$, the optimal dual variables may also be obtained by $\boldsymbol{\alpha}^*_\lambda = \frac{\log(\mathbf{u})}{\lambda} - \frac{\log(\mathbf{u})^\top\mathbf{1}}{p}\mathbf{1}$ and $\boldsymbol{\beta}^*_\lambda = \frac{\log(\mathbf{v})}{\lambda} - \frac{\log(\mathbf{v})^\top\mathbf{1}}{p}\mathbf{1}$, where $\mathbf{1}$ is the p-dimensional all ones vector.

3.2 Optimization

Alongside the fast gradient computation process introduced above, we can further speed up the training with a clever initialization and batch gradient descent.

Initialization. The loss function in eq. (7) is non-convex and is thus highly dependent on the initial setting of $\mathbf{A}$ and $\mathbf{w}$. A good initialization also drastically reduces the number of gradient steps required. For $\mathbf{w}$, we simply initialize all its entries to 1, i.e., all words are assigned with the same weights at the beginning. For $\mathbf{A}$, we propose to learn an initial projection within the word centroid distance (WCD), defined as $D'(\mathbf{d}^a, \mathbf{d}^b) = \|\mathbf{X}\mathbf{d}^a - \mathbf{X}\mathbf{d}^b\|_2$, described in Section 2. The WCD should be a reasonble approximation to the WMD as Kusner et al. [19] point out that the WCD is a lower bound on the WMD, as follows,

$$\sum_{i,j=1}^{n} \mathbf{T}^{ab}_{ij}\|\mathbf{A}(\mathbf{x}_i - \mathbf{x}_j)\|_2 \geq \|\mathbf{A}(\mathbf{X}\mathbf{d}^a - \mathbf{X}\mathbf{d}^b)\|_2 = \|\mathbf{A}(\mathbf{c}^a - \mathbf{c}^b)\|_2$$

where $\mathbf{c}$ and $\mathbf{c}'$ are the WCD centroid vectors for documents $\mathbf{d}$ and $\mathbf{d}'$. This is to say that we can construct the WCD dataset: $\{\mathbf{c}^1, \ldots, \mathbf{c}^m\} \subset \mathbb{R}^d$ and apply NCA in the usual way, as described in Section 2. This is equivalent to running NCA in word embedding space using the WCD distance between documents. We call this learned word distance *Supervised Word Centroid Distance (S-WCD)*. As the WCD is an approximation of the WMD metric, the learned metric $\mathbf{A}$ is a good initialization for the S-WMD optimization.

Stochastic Gradient Descent. Once the initial matrix $\mathbf{A}$ is obtained, we minimize the loss $\ell(\mathbf{A}, \mathbf{w})$ in (7) with minibatch stochastic gradient descent. At each iteration, instead of optimizing over the full training set, we randomly pick a batch of documents $\mathcal{B}$ from the training set, and compute the gradient for these documents. We can further speed up training by observing that the vast majority of NCA probabilities p_{ab} are close to zero. This is because most documents are far away from any given document. Thus, for a document $\mathbf{d}^a$ we can use the WCD to get a cheap neighbor ordering and only compute the NCA probabilities for the closest set of documents $\mathcal{N}_a$, based on the WCD. In particular, the gradient is computed as follows,

$$\mathbf{g}_{\mathbf{A},\mathbf{w}} = \sum_{a\in\mathcal{B}}\sum_{b\in\mathcal{N}a}(p_{ab}/p_a)(\delta_{ab} - p_a)\frac{\partial}{\partial(\mathbf{A},\mathbf{w})}D_{(\mathbf{A},\mathbf{w})}(\mathbf{d}^a, \mathbf{d}^b), \tag{12}$$

where again $\mathcal{N}_a$ is the set of nearest neighbors of document a. With the gradient, we update $\mathbf{A}$ and $\mathbf{w}$ with learning rates $\eta_\mathbf{A}$ and $\eta_\mathbf{w}$, respectively. Algorithm 1 summarizes S-WMD in pseudo code.

Complexity. The empirical time complexity of solving the dual transport problem scales quadratically with p [26]. Therefore, the complexity of our algorithm is $O(i|\mathcal{B}||\mathcal{N}|[p^2 + d^2p + d^2r])$, where i denotes the number of batch gradient descent iterations, p the largest number of unique words in a document, $|\mathcal{B}|$ is the batch size, and $|\mathcal{N}|$ is the nearest neighbor set. This is because computing eq. (12) requires $O(p^2)$ to obtain $\mathbf{T}^*_{ij}$, $\boldsymbol{\alpha}^*$ and $\boldsymbol{\beta}^*$, while constructing the gradient from eqs. (9) and (11) takes $O(d^2p)$ time. Finally, multiplying the sum by $2\mathbf{A}$ requires d^2r time. The approximated gradient eq. (12) requires this computation to be repeated $|\mathcal{B}||\mathcal{N}|$ times. In our experiments, we set $|\mathcal{B}| = 32$ and $|\mathcal{N}| = 200$, and computing the gradient at each iteration can be done in seconds.

4 Results

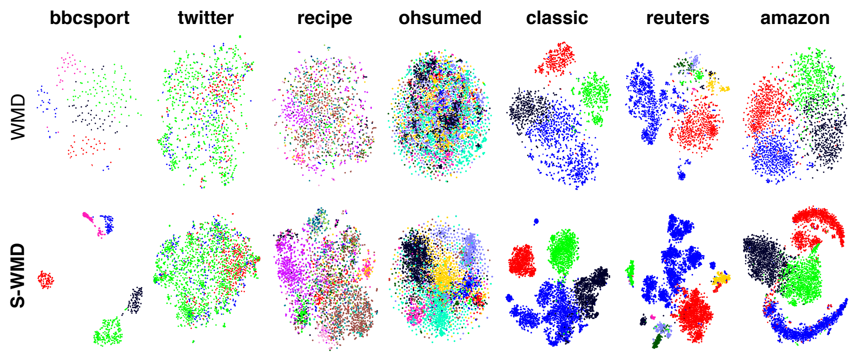

Figure 1: The t-SNE plots of WMD and S-WMD on all datasets.

Algorithm 1 S-WMD

```
Input: word embedding: X,
dataset: {(d^1, y_1), ..., (d^m, y_m)}
c^a = X d^a, ∀a ∈ {1, ..., m}
A = NCA((c^1, y_1), ..., (c^m, y_m))
w = 1
while loop until convergence do
    Select B randomly in {1, ..., m}
    Compute gradient g from Eq. (12)
    A ← A − η_A g_A
    w ← w − η_w g_w
end while
```

We evaluate S-WMD on 8 different document corpora and compare the kNN error with unsupervised WCD, WMD, and 6 document representations. In addition, all 6 document representation baselines are used with and without 3 leading supervised metric learning algorithms—resulting in an overall total of 26 competitive baselines. Our code is implemented in Matlab and is freely available at `http://anonymized`.

Datasets and Baselines. We evaluate all approaches on 8 document datasets in the settings of news categorization, sentiment analysis, and product identification, among others. Table 1 describes the classification tasks as well as the size and number of classes C of each of the datasets. We evaluate against the following document representation/distance methods: 1. *bag-of-words* (BOW): a count of the number of word occurrences in a document, the length of the vector is the number of unique words in the corpus; 2. *term frequency-inverse document frequency* (TF-IDF): the BOW vector normalized by the document frequency of each word across the corpus; 3. *Okapi BM25* [28]: a TF-IDF-like ranking function, first used in search engines; 4. *Latent Semantic Indexing* (LSI) [11]: projects the BOW vectors onto an orthogonal basis via singular value decomposition; 5. *Latent Dirichlet Allocation* (LDA) [2]: a generative probabilistic method that models documents as mixtures of word 'topics'. We train LDA *transductively* (i.e., on the combined collection of training & testing words) and use the topic probabilities as the document representation[1]; 6. *Marginalized Stacked Denoising Autoencoders* (mSDA) [4]: a fast method for training stacked denoising autoencoders, which have state-of-the-art error rates on sentiment analysis tasks [14]. For datasets larger than RECIPE we use either a high-dimensional variant of mSDA or take 20% of the features that occur most often, whichever has better performance.; 7. *Word Centroid Distance* (WCD) [19]: described in [19] as a fast approximation to the WMD; 8. *Word Movers Distance* (WMD) [19]: a method that calculates document distance as the minimum distance to move word embeddings from one document to another by way of the Earth Mover's Distance optimal transport program. We also compare with the Supervised Word Centroid Distance (S-WCD) and the initialization of S-WMD (S-WMD init.), described in Section 3. For methods that propose a document representation (as opposed to a distance), we use the Euclidean distance between these vector representations for visualization and kNN classification. For the supervised metric learning results we first reduce the dimensionality of each representation to 200 dimensions (if necessary) with PCA and then run either NCA, ITML, or LMNN on the projected data. We tune all free hyperparameters in all compared methods (including S-WMD) with Bayesian optimization (BO), using the implementation of Gardner et al. [13][2].

t-SNE visualization. Figure 1 shows a 2D embedding of the test split of each dataset by WMD and S-WMD using t-Stochastic Neighbor Embedding (t-SNE) [34]. The quality of a distance can be visualized by how clustered points in the same class are. Using this metric, S-WMD noticeably improves upon WMD, particularly on BBCSPORT, RECIPE, OHSUMED, CLASSIC, and REUTER.

[1] We use the Matlab Topic Modeling Toolbox [32].

[2] `http://tinyurl.com/bayesopt`

Table 1: The document datasets (and their descriptions) used for visualization and evaluation.

name	description	C	n	ne	BOW dim.	avg words
BBCSPORT	BBC sports articles labeled by sport	5	517	220	13243	117
TWITTER	tweets categorized by sentiment [31]	3	2176	932	6344	9.9
RECIPE	recipe procedures labeled by origin	15	3059	1311	5708	48.5
OHSUMED	medical abstracts (class subsampled)	10	3999	5153	31789	59.2
CLASSIC	academic papers labeled by publisher	4	4965	2128	24277	38.6
REUTERS	news dataset (train/test split [3])	8	5485	2189	22425	37.1
AMAZON	reviews labeled by product	4	5600	2400	42063	45.0
20NEWS	canonical news article dataset [3]	20	11293	7528	29671	72

kNN classification. We show the kNN test error of all document representation and distance methods in Table 3. For datasets that do not have a predefined train/test split: BBCSPORT, TWITTER, RECIPE, CLASSIC, and AMAZON we average results over five $70/30$ train/test splits and report standard errors. For each dataset we highlight the best results in bold (and those whose standard error overlaps the mean of the best result). On the right we also show the average error across datasets, relative to unsupervised BOW (bold indicates the best method). We highlight our new results in red (S-WMD init.) and blue (S-WMD). Despite the very large number of competitive baselines, S-WMD achieves the lowest kNN test error on 5/8 datasets, with the exception of BBCSPORT, CLASSIC and AMAZON. On these datasets it achieves the 4rd lowest on BBCSPORT and CLASSIC, and tied at 2nd on 20NEWS. On average across all datasets it outperforms all other 28 methods. A surprising observation is that S-WMD right after initialization (S-WMD init.) performs competitively well. However, as training S-WMD is quite fast, as described in Table 2 it is often well worth the training time.

For unsupervised baselines, on datasets BBCSPORT and OHSUMED, where the previous state-of-the-art WMD was beaten by LSI, S-WMD reduces the error of LSI relatively by 53% and 19%, respectively. On average, relative to BOW, S-WMD performs 17% and 29% better relative to the second and third place unsupervised methods, WMD and LSI. In general, supervision seems to help all methods on average, save mSDA and LDA. Across all baselines LMNN performs the best with an average error of 0.55 relative to BOW, followed closely by NCA with 0.56 relative error. One reason why NCA with a TF-IDF document representation may be performing better than S-WMD could be because of the long document lengths in BBCSPORT and OHSUMED. Having denser BOW vectors may improve the inverse document frequency weights, which in turn may be a good initialization for NCA to further fine-tune. On datasets with smaller documents such as TWITTER, REUTERS, and CLASSIC, S-WMD outperforms NCA with TF-IDF relatively by 9.2%, 37%, and 42%, respectively. On CLASSIC WMD outperforms S-WMD possibly because of a poor initialization and that S-WMD uses the squared Euclidean distance between word vectors, which may be suboptimal for this dataset. This however, does not occur for any other dataset.

Table 2: The time to compute each distance on the training set. Note that computing S-WMD produces both S-WCD and its initialization for free.

FULL TRAINING TIMES		
DATASET	METRICS	
	S-WCD/S-WMD INIT.	**S-WMD**
BBCSPORT	1m 25s	4m 56s
TWITTER	28m 59s	7m 53s
RECIPE	23m 21s	23m 58s
OHSUMED	46m 18s	29m 12s
CLASSIC	1h 18m	36m 22s
REUTERS	2h 7m	34m 56s
AMAZON	2h 15m	20m 10s
20NEWS	14m 42s	1h 55m

Training time. Table 2 shows the training times on each dataset for the three supervised distances introduced in the paper. We use 25 iterations of Bayesian optimization to select r for S-WCD (for 20NEWS to save time we fix $r = d/2$ beforehand). Computing the S-WMD initialization is free once S-WCD is computed. Relative to the initialization S-WMD is surprisingly fast. This is due to the batch gradient descent and WCD nearest neighbor approximations introduced in Section 3.2. We note that these times are comparable or even faster than the time it takes to train a linear metric on the baseline methods after PCA.

5 Related Work

Metric learning is a vast field that includes both supervised and unsupervised techniques (see Yang & Jin [38] for a large survey). Alongside NCA [15], described in Section 2, there are a number of popular methods for generalized Euclidean metric learning. Large Margin Nearest Neighbors

Table 3: The kNN test error for all datasets and distances.

DATASET	BBCSPORT	TWITTER	RECIPE	OHSUMED	CLASSIC	REUTERS	AMAZON	20NEWS	AVERAGE-RANK
UNSUPERVISED									
BOW	20.6 ± 1.2	43.6 ± 0.4	59.3 ± 1.0	61.1	36.0 ± 0.5	13.9	28.5 ± 0.5	57.8	26.1
TF-IDF	21.5 ± 2.8	33.2 ± 0.9	53.4 ± 1.0	62.7	35.0 ± 1.8	29.1	41.5 ± 1.2	54.4	25.0
OKAPI BM25 [28]	16.9 ± 1.5	42.7 ± 7.8	53.4 ± 1.9	66.2	40.6 ± 2.7	32.8	58.8 ± 2.6	55.9	26.1
LSI [11]	4.3 ± 0.6	31.7 ± 0.7	45.4 ± 0.5	44.2	6.7 ± 0.4	6.3	9.3 ± 0.4	28.9	12.0
LDA [2]	6.4 ± 0.7	33.8 ± 0.3	51.3 ± 0.6	51.0	5.0 ± 0.3	6.9	11.8 ± 0.6	31.5	16.6
MSDA [4]	8.4 ± 0.8	32.3 ± 0.7	48.0 ± 1.4	49.3	6.9 ± 0.4	8.1	17.1 ± 0.4	39.5	18.0
ITML [10]									
BOW	7.4 ± 1.4	32.0 ± 0.4	63.1 ± 0.9	70.1	7.5 ± 0.5	7.3	20.5 ± 2.1	60.6	23.0
TF-IDF	1.8 ± 0.2	31.1 ± 0.3	51.0 ± 1.4	55.1	9.9 ± 1.0	6.6	11.1 ± 1.9	45.3	14.8
OKAPI BM25 [28]	3.7 ± 0.5	31.9 ± 0.3	53.8 ± 1.8	77.0	18.3 ± 4.5	20.7	11.4 ± 2.9	81.5	21.5
LSI [11]	5.0 ± 0.7	32.3 ± 0.4	55.7 ± 0.8	54.7	5.5 ± 0.7	6.9	10.6 ± 2.2	39.6	17.6
LDA [2]	6.5 ± 0.7	33.9 ± 0.9	59.3 ± 0.8	59.6	6.6 ± 0.5	9.2	15.7 ± 2.0	87.8	22.5
MSDA [4]	25.5 ± 9.4	43.7 ± 7.4	54.5 ± 1.3	61.8	14.9 ± 2.2	5.9	37.4 ± 4.0	47.7	23.9
LMNN [37]									
BOW	2.4 ± 0.4	31.8 ± 0.3	48.4 ± 0.4	49.1	4.7 ± 0.3	3.9	10.7 ± 0.3	40.7	11.5
TF-IDF	4.0 ± 0.6	30.8 ± 0.3	43.7 ± 0.3	40.0	4.9 ± 0.3	5.8	6.8 ± 0.3	28.1	7.8
OKAPI BM25 [28]	1.9 ± 0.7	30.5 ± 0.4	41.7 ± 0.7	59.4	19.0 ± 9.3	9.2	6.9 ± 0.2	57.4	14.4
LSI [11]	2.4 ± 0.5	31.6 ± 0.2	44.8 ± 0.4	40.8	3.0 ± 0.1	**3.2**	6.6 ± 0.2	**25.1**	5.1
LDA [2]	4.5 ± 0.4	31.9 ± 0.6	51.4 ± 0.4	49.9	4.9 ± 0.4	5.6	12.1 ± 0.6	32.0	14.6
MSDA [4]	22.7 ± 10.0	50.3 ± 8.6	46.3 ± 1.2	41.6	11.1 ± 1.9	5.3	24.0 ± 3.6	27.1	17.3
NCA [15]									
BOW	9.6 ± 0.6	31.1 ± 0.5	55.2 ± 0.6	57.4	4.0 ± 0.1	6.2	16.8 ± 0.3	46.4	17.5
TF-IDF	$\mathbf{0.6 \pm 0.3}$	30.6 ± 0.5	41.4 ± 0.4	35.8	5.5 ± 0.2	3.8	6.5 ± 0.2	29.3	5.4
OKAPI BM25 [28]	4.5 ± 0.5	31.8 ± 0.4	45.8 ± 0.5	56.6	20.6 ± 4.8	10.5	8.5 ± 0.4	55.9	17.9
LSI [11]	2.4 ± 0.7	31.1 ± 0.8	41.6 ± 0.5	37.5	3.1 ± 0.2	3.3	7.7 ± 0.4	30.7	6.3
LDA [2]	7.1 ± 0.9	32.7 ± 0.3	50.9 ± 0.4	50.7	5.0 ± 0.2	7.9	11.6 ± 0.8	30.9	16.5
MSDA [4]	21.8 ± 7.4	37.9 ± 2.8	48.0 ± 1.6	40.4	11.2 ± 1.8	5.2	23.6 ± 3.1	26.8	16.1
DISTANCES IN THE WORD MOVER'S FAMILY									
WCD [19]	11.3 ± 1.1	30.7 ± 0.9	49.4 ± 0.3	48.9	6.6 ± 0.2	4.7	9.2 ± 0.2	36.2	13.5
WMD [19]	4.6 ± 0.7	28.7 ± 0.6	42.6 ± 0.3	44.5	$\mathbf{2.8 \pm 0.1}$	3.5	7.4 ± 0.3	26.8	6.1
S-WCD	4.6 ± 0.5	30.4 ± 0.5	51.3 ± 0.2	43.3	5.8 ± 0.2	3.9	7.6 ± 0.3	33.6	11.4
S-WMD INIT.	2.8 ± 0.3	28.2 ± 0.4	39.8 ± 0.4	38.0	3.3 ± 0.3	3.5	$\mathbf{5.8 \pm 0.2}$	28.4	4.3
S-WMD	2.1 ± 0.5	$\mathbf{27.5 \pm 0.5}$	$\mathbf{39.2 \pm 0.3}$	**34.3**	3.2 ± 0.2	**3.2**	$\mathbf{5.8 \pm 0.1}$	26.8	**2.4**

(LMNN) [37] learns a metric that encourages inputs with similar labels to be close in a local region, while encouraging inputs with different labels to be farther by a large margin. Information-Theoretic Metric Learning (ITML) [10] learns a metric by minimizing a KL-divergence subject to generalized Euclidean distance constraints. Cuturi & Avis [7] was the first to consider learning the ground distance in the Earth Mover's Distance (EMD). In a similar work, Wang & Guibas [35] learns a ground distance that is not a metric, with good performance in certain vision tasks. Most similar to our work Wang et al. [36] learn a metric within a generalized Euclidean EMD ground distance using the framework of ITML for image classification. They do not, however, consider re-weighting the histograms, which allows our method extra flexibility. Until recently, there has been relatively little work towards learning supervised word embeddings, as state-of-the-art results rely on making use of large unlabeled text corpora. Tang et al. [33] propose a neural language model that uses label information from emoticons to learn sentiment-specific word embeddings.

6 Conclusion

We proposed a powerful method to learn a supervised word mover's distance, and demonstrated that it may well be the best performing distance metric for documents to date. Similar to WMD, our S-WMD benefits from the large unsupervised corpus, which was used to learn the word2vec embedding [22, 23]. The word embedding gives rise to a very good document distance, which is particularly forgiving when two documents use syntactically different but conceptually similar words. Two words may be similar in one sense (topic) but dissimilar in another (authorship), depending on the articles in which they are contained. It is these differences that S-WMD manages to capture through supervised training. By learning a linear metric and histogram re-weighting through the optimal transport of the word mover's distance, we are able to produce state-of-the-art classification results in a surpisingly short training time.

References

[1] Bertsimas, D. and Tsitsiklis, J. N. *Introduction to linear optimization*. Athena Scientific, 1997.

[2] Blei, D. M., Ng, A. Y., and Jordan, M. I. Latent dirichlet allocation. *JMLR*, 2003.

[3] Cardoso-Cachopo, A. Improving Methods for Single-label Text Categorization. PdD Thesis, Instituto Superior Tecnico, Universidade Tecnica de Lisboa, 2007.

[4] Chen, M., Xu, Z., Weinberger, K. Q., and Sha, F. Marginalized denoising autoencoders for domain adaptation. In *ICML*, 2012.

[5] Collobert, R. and Weston, J. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *ICML*, pp. 160–167. ACM, 2008.

[6] Cuturi, M. Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in Neural Information Processing Systems*, pp. 2292–2300, 2013.

[7] Cuturi, M. and Avis, D. Ground metric learning. *JMLR*, 2014.

[8] Cuturi, M. and Doucet, A. Fast computation of wasserstein barycenters. In Jebara, Tony and Xing, Eric P. (eds.), *ICML*, pp. 685–693. JMLR Workshop and Conference Proceedings, 2014.

[9] Cuturi, M. and Peyre, G. A smoothed dual approach for variational wasserstein problems. *SIAM Journal on Imaging Sciences*, 9(1):320–343, 2016.

[10] Davis, J.V., Kulis, B., Jain, P., Sra, S., and Dhillon, I.S. Information-theoretic metric learning. In *ICML*, pp. 209–216, 2007.

[11] Deerwester, S. C., Dumais, S. T., Landauer, T. K., Furnas, G. W., and Harshman, R. A. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.

[12] Frogner, C., Zhang, C., Mobahi, H., Araya, M., and Poggio, T.A. Learning with a wasserstein loss. In *Advances in Neural Information Processing Systems*, pp. 2044–2052, 2015.

[13] Gardner, J., Kusner, M. J., Xu, E., Weinberger, K. Q., and Cunningham, J. Bayesian optimization with inequality constraints. In *ICML*, pp. 937–945, 2014.

[14] Glorot, X., Bordes, A., and Bengio, Y. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *ICML*, pp. 513–520, 2011.

[15] Goldberger, J., Hinton, G.E., Roweis, S.T., and Salakhutdinov, R. Neighbourhood components analysis. In *NIPS*, pp. 513–520. 2005.

[16] Gopalan, P. K., Charlin, L., and Blei, D. Content-based recommendations with poisson factorization. In *NIPS*, pp. 3176–3184, 2014.

[17] Greene, D. and Cunningham, P. Practical solutions to the problem of diagonal dominance in kernel document clustering. In *ICML*, pp. 377–384. ACM, 2006.

[18] Hinton, G.E. and Roweis, S.T. Stochastic neighbor embedding. In *NIPS*, pp. 833–840. MIT Press, 2002.

[19] Kusner, M. J., Sun, Y., Kolkin, N. I., and Weinberger, K. Q. From word embeddings to document distances. In *ICML*, 2015.

[20] Levina, E. and Bickel, P. The earth mover's distance is the mallows distance: Some insights from statistics. In *ICCV*, volume 2, pp. 251–256. IEEE, 2001.

[21] Levy, O. and Goldberg, Y. Neural word embedding as implicit matrix factorization. In *NIPS*, 2014.

[22] Mikolov, T., Chen, K., Corrado, G., and Dean, J. Efficient estimation of word representations in vector space. In *Workshop at ICLR*, 2013.

[23] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. Distributed representations of words and phrases and their compositionality. In *NIPS*, pp. 3111–3119, 2013.

[24] Mohan, A., Chen, Z., and Weinberger, K. Q. Web-search ranking with initialized gradient boosted regression trees. *JMLR*, 14:77–89, 2011.

[25] Ontrup, J. and Ritter, H. Hyperbolic self-organizing maps for semantic navigation. In *NIPS*, 2001.

[26] Pele, O. and Werman, M. Fast and robust earth mover's distances. In *ICCV*, pp. 460–467. IEEE, 2009.

[27] Perina, A., Jojic, N., Bicego, M., and Truski, A. Documents as multiple overlapping windows into grids of counts. In *NIPS*, pp. 10–18. 2013.

[28] Robertson, S. E., Walker, S., Jones, S., Hancock-Beaulieu, M. M., Gatford, M., et al. Okapi at trec-3. *NIST SPECIAL PUBLICATION SP*, pp. 109–109, 1995.

[29] Rubner, Y., Tomasi, C., and Guibas, L. J. A metric for distributions with applications to image databases. In *ICCV*, pp. 59–66. IEEE, 1998.

[30] Salton, G. and Buckley, C. Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523, 1988.

[31] Sanders, N. J. Sanders-twitter sentiment corpus, 2011.

[32] Steyvers, M. and Griffiths, T. Probabilistic topic models. *Latent semantic analysis: a road to meaning*, 2007.

[33] Tang, D., Wei, F., Yang, N., Zhou, M., Liu, T., and Qin, B. Learning sentiment-specific word embedding for twitter sentiment classification. In *ACL*, pp. 1555–1565, 2014.

[34] Van der Maaten, L. and Hinton, G. Visualizing data using t-sne. *JMLR*, 9(2579-2605):85, 2008.

[35] Wang, F. and Guibas, L. J. Supervised earth movers distance learning and its computer vision applications. In *ECCV*. 2012.

[36] Wang, X-L., Liu, Y., and Zha, H. Learning robust cross-bin similarities for the bag-of-features model. Technical report, Peking University, China, 2009.

[37] Weinberger, K.Q. and Saul, L.K. Distance metric learning for large margin nearest neighbor classification. *JMLR*, 10:207–244, 2009.

[38] Yang, L. and Jin, R. Distance metric learning: A comprehensive survey. 2, 2006.