From Word Embeddings To Document Distances

Matt J. Kusner MKUSNER@WUSTL.EDU
Yu Sun YUSUN@WUSTL.EDU
Nicholas I. Kolkin N.KOLKIN@WUSTL.EDU
Kilian Q. Weinberger KILIAN@WUSTL.EDU
Washington University in St. Louis, 1 Brookings Dr., St. Louis, MO 63130

Abstract

We present the Word Mover's Distance (WMD), a novel distance function between text documents. Our work is based on recent results in word embeddings that learn semantically meaningful representations for words from local co-occurrences in sentences. The WMD distance measures the dissimilarity between two text documents as the minimum amount of distance that the embedded words of one document need to "travel" to reach the embedded words of another document. We show that this distance metric can be cast as an instance of the Earth Mover's Distance, a well studied transportation problem for which several highly efficient solvers have been developed. Our metric has no hyperparameters and is straight-forward to implement. Further, we demonstrate on eight real world document classification data sets, in comparison with seven state-of-the-art baselines, that the WMD metric leads to unprecedented low k-nearest neighbor document classification error rates.

1. Introduction

Accurately representing the distance between two documents has far-reaching applications in document retrieval (Salton & Buckley, 1988), news categorization and clustering (Ontrup & Ritter, 2001; Greene & Cunningham, 2006), song identification (Brochu & Freitas, 2002), and multilingual document matching (Quadrianto et al., 2009).

The two most common ways documents are represented is via a bag of words (BOW) or by their term frequency-inverse document frequency (TF-IDF). However, these features are often not suitable for document distances due to their frequent near-orthogonality (Schölkopf et al., 2002; Greene & Cunningham, 2006). Another significant drawback of these representations are that they do not capture the distance between individual words. Take for example the two sentences in different documents: *Obama speaks to the media in Illinois* and: *The President greets the press in Chicago*. While these sentences have no words in common, they convey nearly the same information, a fact that cannot be represented by the BOW model. In this case, the closeness of the word pairs: (*Obama, President*); (*speaks, greets*); (*media, press*); and (*Illinois, Chicago*) is not factored into the BOW-based distance.

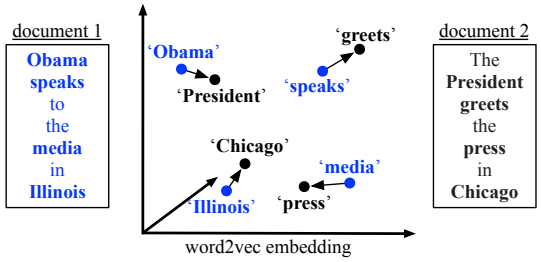

Figure 1. An illustration of the *word mover's distance*. All non-stop words (***bold***) of both documents are embedded into a *word2vec* space. The distance between the two documents is the minimum cumulative distance that all words in document 1 need to travel to exactly match document 2. (Best viewed in color.)

There have been numerous methods that attempt to circumvent this problem by learning a latent low-dimensional representation of *documents*. Latent Semantic Indexing (LSI) (Deerwester et al., 1990) eigendecomposes the BOW feature space, and Latent Dirichlet Allocation (LDA) (Blei et al., 2003) probabilistically groups similar words into topics and represents documents as distribution over these topics. At the same time, there are many competing variants of BOW/TF-IDF (Salton & Buckley, 1988; Robertson & Walker, 1994). While these approaches produce a more coherent *document representation* than BOW, they often do not improve the empirical performance of BOW on distance-based tasks (*e.g.,* nearest-neighbor classifiers) (Petterson et al., 2010; Mikolov et al., 2013c).

Proceedings of the 32^{nd} International Conference on Machine Learning, Lille, France, 2015. JMLR: W&CP volume 37.

In this paper we introduce a new metric for the distance between text documents. Our approach leverages recent results by Mikolov et al. (2013b) whose celebrated *word2vec* model generates word embeddings of unprecedented quality and scales naturally to very large data sets (*e.g.*, we use a freely-available model trained on approximately 100 billion words). The authors demonstrate that semantic relationships are often preserved in vector operations on word vectors, *e.g.,* vec(*Berlin*) - vec(*Germany*) + vec(*France*) is close to vec(*Paris*). This suggests that distances and between embedded word vectors are to some degree semantically meaningful. Our metric, which we call the *Word Mover's Distance* (WMD), utilizes this property of *word2vec* embeddings. We represent text documents as a weighted point cloud of embedded words. The distance between two text documents A and B is the minimum cumulative distance that words from document A need to travel to match exactly the point cloud of document B. Figure 1 shows a schematic illustration of our new metric.

The optimization problem underlying WMD reduces to a special case of the well-studied Earth Mover's Distance (Rubner et al., 1998) transportation problem and we can leverage existing literature on fast specialized solvers (Pele & Werman, 2009). We also compare several lower bounds and show that these can be used as approximations or to prune away documents that are provably not amongst the k-nearest neighbors of a query.

The WMD distance has several intriguing properties: 1. it is **hyper-parameter free** and straight-forward to understand and use; 2. it is highly **interpretable** as the distance between two documents can be broken down and explained as the sparse distances between few individual words; 3. it naturally incorporates the knowledge encoded in the *word2vec* space and leads to **high retrieval accuracy**—it outperforms all 7 state-of-the-art alternative document distances in 6 of 8 real world classification tasks.

2. Related Work

Constructing a distance between documents is closely tied with learning new document representations. One of the first works to systematically study different combinations of term frequency-based weightings, normalization terms, and corpus-based statistics is Salton & Buckley (1988). Another variation is the Okapi BM25 function (Robertson & Walker, 1994) which describes a score for each (word, document) pair and is designed for ranking applications. Aslam & Frost (2003) derive an information-theoretic similarity score between two documents, based on probability of word occurrence in a document corpus. Croft & Lafferty (2003) use a language model to describe the probability of generating a word from a document, similar to LDA (Blei et al., 2003). Most similar to our method is that of Wan (2007) which first decomposes each document into a set of subtopic units via TextTiling (Hearst, 1994), and then measures the effort required to transform a subtopic set into another via the EMD (Monge, 1781; Rubner et al., 1998).

New approaches for learning document representations include Stacked Denoising Autoencoders (SDA) (Glorot et al., 2011), and the faster mSDA (Chen et al., 2012), which learn word correlations via dropout noise in stacked neural networks. Recently, the Componential Counting Grid (Perina et al., 2013) merges LDA (Blei et al., 2003) and Counting Grid (Jojic & Perina, 2011) models, allowing 'topics' to be mixtures of word distributions. As well, Le & Mikolov (2014) learn a dense representation for documents using a simplified neural language model, inspired by the word2vec model (Mikolov et al., 2013a).

The use of the EMD has been pioneered in the computer vision literature (Rubner et al., 1998; Ren et al., 2011). Several publications investigate approximations of the EMD for image retrieval applications (Grauman & Darrell, 2004; Shirdhonkar & Jacobs, 2008; Levina & Bickel, 2001). As word embeddings improve in quality, document retrieval enters an analogous setup, where each word is associated with a highly informative feature vector. To our knowledge, our work is the first to make the connection between high quality word embeddings and EMD retrieval algorithms.

Cuturi (2013) introduces an entropy penalty to the EMD objective, which allows the resulting approximation to be solved with very efficient iterative matrix updates. Further, the vectorization enables parallel computation via GPGPUs However, their approach assumes that the number of dimensions per document is not too high, which in our setting is extremely large (all possible words). This removes the main benefit (parallelization on GPGPUs) of their approach and so we develop a new EMD approximation that appears to be very effective for our problem domain.

3. Word2Vec Embedding

Recently Mikolov et al. (2013a;b) introduced *word2vec*, a novel word-embedding procedure. Their model learns a vector representation for each word using a (shallow) neural network language model. Specifically, they propose a neural network architecture (the *skip-gram model*) that consists of an input layer, a projection layer, and an output layer to predict nearby words. Each word vector is trained to maximize the log probability of neighboring words in a corpus, *i.e.,* given a sequence of words $w_1, \ldots, w_T$,

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{j\in nb(t)}\log p(w_j|w_t)$$

where $nb(t)$ is the set of neighboring words of word w_t and $p(w_j|w_t)$ is the hierarchical softmax of the associated word

vectors $\mathbf{v}_{w_j}$ and $\mathbf{v}_{w_t}$ (see Mikolov et al. (2013a) for more details). Due to its surprisingly simple architecture and the use of the hierarchical softmax, the skip-gram model can be trained on a single machine on billions of words per hour using a conventional desktop computer. The ability to train on very large data sets allows the model to learn complex word relationships such as vec(*Japan*) - vec(*sushi*) + vec(*Germany*) $\approx$ vec(*bratwurst*) and vec(*Einstein*) - vec(*scientist*) + vec(*Picasso*) $\approx$ vec(*painter*) (Mikolov et al., 2013a;b). Learning the word embedding is entirely unsupervised and it can be computed on the text corpus of interest or be pre-computed in advance. Although we use *word2vec* as our preferred embedding throughout, other embeddings are also plausible (Collobert & Weston, 2008; Mnih & Hinton, 2009; Turian et al., 2010).

4. Word Mover's Distance

Assume we are provided with a *word2vec* embedding matrix $\mathbf{X} \in \mathbb{R}^{d \times n}$ for a finite size vocabulary of n words. The i^{th} column, $\mathbf{x}_i \in \mathcal{R}^d$, represents the embedding of the i^{th} word in d-dimensional space. We assume text documents are represented as normalized bag-of-words (nBOW) vectors, $\mathbf{d} \in \mathcal{R}^n$. To be precise, if word i appears c_i times in the document, we denote $d_i = \frac{c_i}{\sum_{j=1}^n c_j}$. An nBOW vector $\mathbf{d}$ is naturally very sparse as most words will not appear in any given document. (We remove stop words, which are generally category independent.)

nBOW representation. We can think of the vector $\mathbf{d}$ as a point on the $n-1$ dimensional simplex of word distributions. Two documents with different unique words will lie in different regions of this simplex. However, these documents may still be *semantically close*. Recall the earlier example of two similar, but word-different sentences in one document: "*Obama speaks to the media in Illinois*" and in another: "*The President greets the press in Chicago*". After stop-word removal, the two corresponding nBOW vectors $\mathbf{d}$ and $\mathbf{d}'$ have no common non-zero dimensions and therefore have close to maximum simplex distance, although their true distance is small.

Word travel cost. Our goal is to incorporate the semantic similarity between individual word pairs (*e.g. President* and *Obama*) into the document distance metric. One such measure of word dissimilarity is naturally provided by their Euclidean distance in the *word2vec* embedding space. More precisely, the distance between word i and word j becomes $c(i, j) = \|\mathbf{x}_i - \mathbf{x}_j\|_2$. To avoid confusion between word and document distances, we will refer to $c(i, j)$ as the *cost* associated with "traveling" from one word to another.

Document distance. The "travel cost" between two *words* is a natural building block to create a distance between two *documents*. Let $\mathbf{d}$ and $\mathbf{d}'$ be the nBOW representation of two text documents in the $(n-1)$-simplex. First, we allow each word i in $\mathbf{d}$ to be transformed into any word in $\mathbf{d}'$ in total or in parts. Let $\mathbf{T} \in \mathcal{R}^{n \times n}$ be a (sparse) flow matrix where $\mathbf{T}_{ij} \geq 0$ denotes *how much* of word i in $\mathbf{d}$ travels to word j in $\mathbf{d}'$. To transform $\mathbf{d}$ entirely into $\mathbf{d}'$ we ensure that the entire outgoing flow from word i equals d_i, *i.e.* $\sum_j \mathbf{T}_{ij} = d_i$. Further, the amount of incoming flow to word j must match d'_j, *i.e.* $\sum_i \mathbf{T}_{ij} = d'_j$. Finally, we can define the distance between the two documents as the minimum (weighted) cumulative cost required to move all words from $\mathbf{d}$ to $\mathbf{d}'$, *i.e.* $\sum_{i,j} \mathbf{T}_{ij} c(i, j)$.

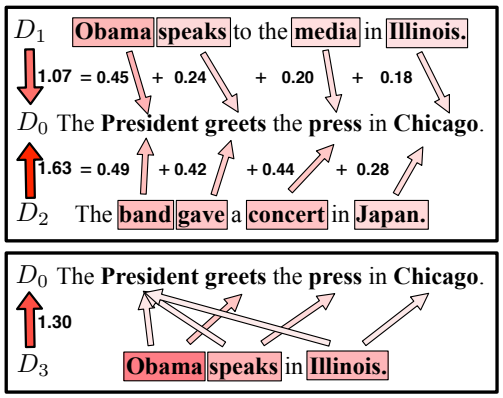

Figure 2. (Top:) The components of the WMD metric between a query D_0 and two sentences D_1, D_2 (with equal BOW distance). The arrows represent flow between two words and are labeled with their distance contribution. *(Bottom:)* The flow between two sentences D_3 and D_0 with different numbers of words. This mismatch causes the WMD to move words to multiple similar words.

Transportation problem. Formally, the minimum cumulative cost of moving $\mathbf{d}$ to $\mathbf{d}'$ given the constraints is provided by the solution to the following linear program,

$$\begin{aligned} \min_{\mathbf{T} \geq 0} \quad & \sum_{i,j=1}^{n} \mathbf{T}_{ij} c(i, j) \\ \text{subject to:} \quad & \sum_{j=1}^{n} \mathbf{T}_{ij} = d_i \quad \forall i \in \{1, \dots, n\} \\ & \sum_{i=1}^{n} \mathbf{T}_{ij} = d'_j \quad \forall j \in \{1, \dots, n\}. \end{aligned} \tag{1}$$

The above optimization is a special case of the earth mover's distance metric (EMD) (Monge, 1781; Rubner et al., 1998; Nemhauser & Wolsey, 1988), a well studied transportation problem for which specialized solvers have been developed (Ling & Okada, 2007; Pele & Werman, 2009). To highlight this connection we refer to the resulting metric as the *word mover's distance* (WMD). As the cost $c(i, j)$ is a metric, it can readily be shown that the WMD is also a metric (Rubner et al., 1998).

Visualization. Figure 2 illustrates the WMD metric on two sentences D_1 and D_2 which we would like to compare

to the query sentence D_0. First, stop-words are removed, leaving *President, greets, press, Chicago* in D_0 each with $\mathbf{d}_i = 0.25$. The arrows from each word i in sentences D_1 and D_2 to word j in D_0 are labeled with their contribution to the distance $\mathbf{T}_{ij}c(i,j)$. We note that the WMD agrees with our intuition, and "moves" words to semantically similar words. Transforming *Illinois* into *Chicago* is much cheaper than is *Japan* into *Chicago*. This is because the *word2vec* embedding places the vector vec(*Illinois*) closer to vec(*Chicago*) than vec(*Japan*). Consequently, the distance from D_0 to D_1 (1.07) is significantly smaller than to D_2 (1.63). Importantly however, both sentences D_1 and D_2 have the same bag-of-words/TF-IDF distance from D_0, as neither shares any words in common with D_0. An additional example D_3 highlights the flow when the number of words does not match. D_3 has term weights $d_j = 0.33$ and excess flow is sent to other similar words. This increases the distance, although the effect might be artificially magnified due to the short document lengths as longer documents may contain several similar words.

4.1. Fast Distance Computation

The best average time complexity of solving the WMD optimization problem scales $O(p^3 \log p)$, where p denotes the number of unique words in the documents (Pele & Werman, 2009). For datasets with many unique words (*i.e.*, high-dimensional) and/or a large number of documents, solving the WMD optimal transport problem can become prohibitive. We can however introduce several cheap lower bounds of the WMD transportation problem that allows us to prune away the majority of the documents without ever computing the exact WMD distance.

Word centroid distance. Following the work of Rubner et al. (1998) it is straight-forward to show (via the triangle inequality) that the 'centroid' distance $\|\mathbf{Xd} - \mathbf{Xd}'\|_2$ must lower bound the WMD between documents $\mathbf{d}, \mathbf{d}'$,

$$\begin{aligned}
&\sum_{i,j=1}^{n} \mathbf{T}_{ij}c(i,j) = \sum_{i,j=1}^{n} \mathbf{T}_{ij}\|\mathbf{x}_i - \mathbf{x}'_j\|_2 \\
&= \sum_{i,j=1}^{n} \|\mathbf{T}_{ij}(\mathbf{x}_i - \mathbf{x}'_j)\|_2 \geq \Big\| \sum_{i,j=1}^{n} \mathbf{T}_{ij}(\mathbf{x}_i - \mathbf{x}'_j) \Big\|_2 \\
&= \Big\| \sum_{i=1}^{n} \Big(\sum_{j=1}^{n} \mathbf{T}_{ij} \Big) \mathbf{x}_i - \sum_{j=1}^{n} \Big(\sum_{i=1}^{n} \mathbf{T}_{ij} \Big) \mathbf{x}'_j \Big\|_2 \\
&= \Big\| \sum_{i=1}^{n} d_i \mathbf{x}_i - \sum_{j=1}^{n} d'_j \mathbf{x}'_j \Big\|_2 = \|\mathbf{Xd} - \mathbf{Xd}'\|_2.
\end{aligned}$$

We refer to this distance as the *Word Centroid Distance* (WCD) as each document is represented by its weighted average word vector. It is very fast to compute via a few matrix operations and scales $O(dp)$. For nearest-neighbor applications we can use this centroid-distance to inform our nearest neighbor search about promising candidates, which allows us to speed up the exact WMD search significantly. We can also use WCD to limit our k-nearest neighbor search to a small subset of most promising candidates, resulting in an even faster approximate solution.

Relaxed word moving distance. Although the WCD is fast to compute, it is not very tight (see section 5). We can obtain much tighter bounds by relaxing the WMD optimization problem and removing one of the two constraints respectively (removing both constraints results in the trivial lower bound $\mathbf{T} = 0$.) If just the second constraint is removed, the optimization becomes,

$$\begin{aligned}
&\min_{\mathbf{T} \geq 0} \sum_{i,j=1}^{n} \mathbf{T}_{ij}c(i,j) \\
&\text{subject to: } \sum_{j=1}^{n} \mathbf{T}_{ij} = d_i \quad \forall i \in \{1, \ldots, n\}.
\end{aligned}$$

This relaxed problem must yield a lower-bound to the WMD distance, which is evident from the fact that every WMD solution (satisfying both constraints) must remain a feasible solution if one constraint is removed.

The optimal solution is for each word in $\mathbf{d}$ to move all its probability mass to the most similar word in $\mathbf{d}'$. Precisely, an optimal $\mathbf{T}^*$ matrix is defined as

$$\mathbf{T}^*_{ij} = \begin{cases} d_i \text{ if } j = \operatorname{argmin}_j c(i,j) \\ 0 \text{ otherwise.} \end{cases} \tag{2}$$

The optimality of this solution is straight-forward to show. Let $\mathbf{T}$ be any feasible matrix for the relaxed problem, the contribution to the objective value for any word i, with closest word $j^* = \operatorname{argmin}_j c(i,j)$, cannot be smaller:

$$\begin{aligned}
&\sum_j \mathbf{T}_{ij}c(i,j) \geq \sum_j \mathbf{T}_{ij}c(i,j^*) = c(i,j^*) \sum_j \mathbf{T}_{ij} \\
&= c(i,j^*)d_i = \sum_j \mathbf{T}^*_{ij}c(i,j).
\end{aligned}$$

Therefore, $\mathbf{T}^*$ must yield a minimum objective value. Computing this solution requires only the identification of $j^* = \operatorname{argmin}_i c(i,j)$, which is a nearest neighbor search in the Euclidean *word2vec* space. For each word vector $\mathbf{x}_i$ in document D we need to find the most similar word vector $\mathbf{x}_j$ in document D'. The second setting, when the first constraint is removed, is almost identical except that the nearest neighbor search is reversed. Both lower bounds ultimately rely on pairwise distance computations between word vectors. These computations can be combined and reused to obtain both bounds jointly at little additional overhead. Let the two relaxed solutions be $\ell_1(\mathbf{d}, \mathbf{d}')$ and $\ell_2(\mathbf{d}, \mathbf{d}')$ respectively. We can obtain an

even tighter bound by taking the maximum of the two, $\ell_r(\mathbf{d}, \mathbf{d}') = \max(\ell_1(\mathbf{d}, \mathbf{d}'), \ell_2(\mathbf{d}, \mathbf{d}'))$, which we refer to as the *Relaxed WMD* (RWMD). This bound is significantly tighter than WCD. The nearest neighbor search has a time complexity of $O(p^2)$, and it can be sped up further by leveraging out-of-the-box tools for fast (approximate or exact) nearest neighbor retrieval (Garcia et al., 2008; Yianilos, 1993; Andoni & Indyk, 2006).

Prefetch and prune. We can use the two lower bounds to drastically reduce the amount of WMD distance computations we need to make in order to find the k nearest neighbors of a query document. We first sort all documents in increasing order of their (extremely cheap) WCD distance to the query document and compute the exact WMD distance to the first k of these documents. Subsequently, we traverse the remaining documents. For each we first check if the RWMD lower bound exceeds the distance of the current k^{th} closest document, if so we can prune it. If not, we compute the WMD distance and update the k nearest neighbors if necessary. As the RWMD approximation is very tight, it allows us to prune up to 95% of all documents on some data sets. If the exact k nearest neighbors are not required, an additional speedup can be obtained if this traversal is limited to $m < n$ documents. We refer to this algorithm as *prefetch and prune*. If $m = k$, this is equivalent to returning the k nearest neighbors of the WCD distance. If $m = n$ it is exact as only provably non-neighbors are pruned.

5. Results

We evaluate the word mover's distance in the context of kNN classification on eight benchmark document categorization tasks. We first describe each dataset and a set of classic and state-of-the-art document representations and distances. We then compare the nearest-neighbor performance of WMD and the competing methods on these datasets. Finally, we examine how the fast lower bound distances can speedup nearest neighbor computation by prefetching and pruning neighbors. Matlab code for the WMD metric is available at `http://matthewkusner.com`

5.1. Dataset Description and Setup

We evaluate all approaches on 8 supervised document datasets: BBCSPORT: BBC sports articles between 2004-2005, TWITTER: a set of tweets labeled with sentiments 'positive', 'negative', or 'neutral' (Sanders, 2011) (the set is reduced due to the unavailability of some tweets), RECIPE: a set of recipe procedure descriptions labeled by their region of origin, OHSUMED: a collection of medical abstracts categorized by different cardiovascular disease groups (for computational efficiency we subsample the dataset, using the first 10 classes), CLASSIC: sets of sentences from academic papers, labeled by publisher name, REUTERS: a classic news dataset labeled by news topics (we use the 8-class version with train/test split as described in Cardoso-Cachopo (2007)), AMAZON: a set of Amazon reviews which are labeled by category product in {*books, dvd, electronics, kitchen*} (as opposed to by sentiment), and 20NEWS: news articles classified into 20 different categories (we use the "bydate" train/test split[1] Cardoso-Cachopo (2007)). We preprocess all datasets by removing all words in the SMART stop word list (Salton & Buckley, 1971). For 20NEWS, we additionally remove all words that appear less than 5 times across all documents. Finally, to speed up the computation of WMD (and its lower bounds) we limit all 20NEWS documents to the most common 500 words (in each document) for WMD-based methods.

Table 1. Dataset characteristics, used in evaluation.

NAME	n	BOW DIM.	UNIQUE WORDS (AVG)	$\lvert\mathcal{Y}\rvert$
BBCSPORT	517	13243	117	5
TWITTER	2176	6344	9.9	3
RECIPE	3059	5708	48.5	15
OHSUMED	3999	31789	59.2	10
CLASSIC	4965	24277	38.6	4
REUTERS	5485	22425	37.1	8
AMAZON	5600	42063	45.0	4
20NEWS	11293	29671	72	20

We split each dataset into training and testing subsets (if not already done so). Table 1 shows relevant statistics for each of these training datasets including the number of inputs n, the bag-of-words dimensionality, the average number of unique words per document, and the number of classes $|\mathcal{Y}|$. The word embedding used in our WMD implementation is the freely-available word2vec word embedding[2] which has an embedding for 3 million words/phrases (from Google News), trained using the approach in Mikolov et al. (2013b). Words that are not present in the pre-computed word2vec model are dropped for the WMD metric (and its lower bounds), but kept for all baselines (thus giving the baselines a slight competitive advantage).

We compare against 7 document representation baselines:

bag-of-words (BOW). A vector of word counts of dimensionality d, the size of the dictionary.

TFIDF term frequency-inverse document frequency (Salton & Buckley, 1988): the bag-of-words representation divided by each word's document frequency.

BM25 Okapi: (Robertson et al., 1995) a ranking function that extends TF-IDF for each word w in a document D:

$$BM25(w, D) = \frac{IDF(w)TF(w,D)(k_1+1)}{TF(w,D)+k_1(1-b+b\frac{|D|}{D_{avg}})}$$

where $IDF(w)$ is the inverse document frequency of word

[1] `http://qwone.com/~jason/20Newsgroups/`
[2] `https://code.google.com/p/word2vec/`

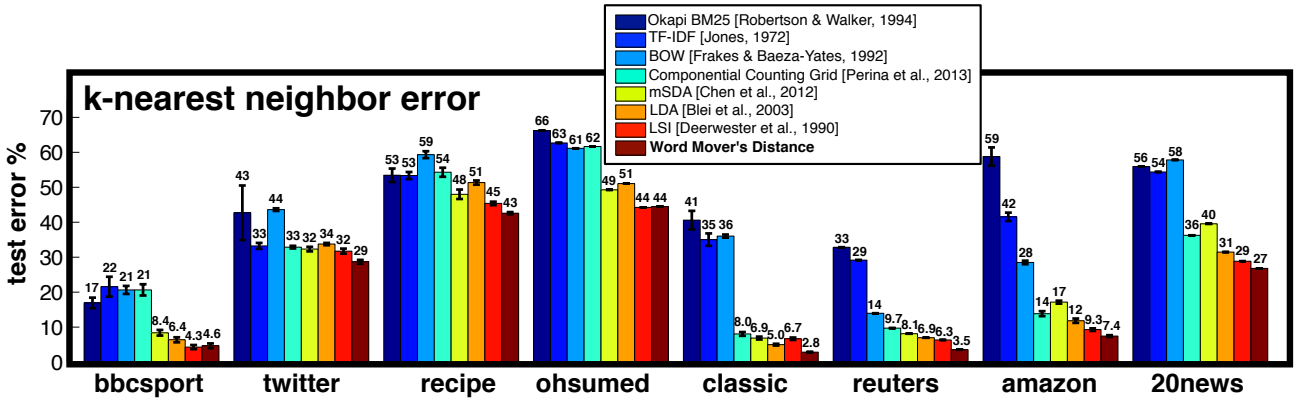

Figure 3. The kNN test error results on 8 document classification data sets, compared to canonical and state-of-the-art baselines methods.

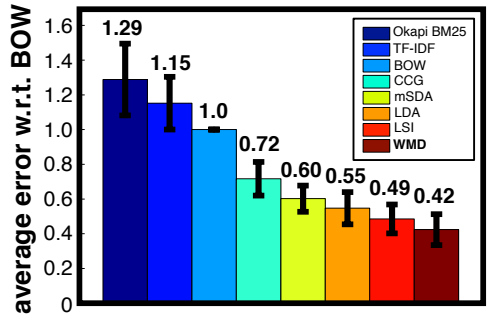

Figure 4. The kNN test errors of various document metrics averaged over all eight datasets, relative to kNN with BOW.

w, $TF(w, D)$ is its term frequency in document D, $|D|$ is the number of words in the document, D_{avg} is the average size of a document, and k_1 and b are free parameters.

LSI Latent Semantic Indexing (Deerwester et al., 1990): uses singular value decomposition on the BOW representation to arrive at a semantic feature space.

LDA Latent Dirichlet Allocation (Blei et al., 2003): a celebrated generative model for text documents that learns representations for documents as distributions over word topics. We use the Matlab Topic Modeling Toolbox Steyvers & Griffiths (2007) and allow 100 iterations for burn-in and run the chain for 1000 iterations afterwards. Importantly, for each dataset we train LDA *transductively*, *i.e.* we train on the union of the training and holdout sets.

mSDA Marginalized Stacked Denoising Autoencoder (Chen et al., 2012): a representation learned from stacked denoting autoencoders (SDAs), marginalized for fast training. In general, SDAs have been shown to have state-of-the-art performance for document sentiment analysis tasks (Glorot et al., 2011). For high-dimensional datasets (*i.e.*, all except BBCSPORT, TWITTER, and RECIPE) we use either the high-dimensional version of mSDA (Chen et al., 2012) or limit the features to the top 20% of the words (ordered by occurence), whichever performs better.

CCG Componential Counting Grid (Perina et al., 2013): a generative model that directly generalizes the Counting Grid (Jojic & Perina, 2011), which models documents as a mixture of word distributions, and LDA (Blei et al., 2003). We use the grid location admixture probability of each document as the new representation.

Table 2. Test error percentage and standard deviation for different text embeddings. NIPS, AMZ, News are *word2vec* (w2v) models trained on different data sets whereas HLBL and Collo were also obtained with other embedding algorithms.

DOCUMENT k-NEAREST NEIGHBOR RESULTS					
DATASET	HLBL	CW	NIPS (W2V)	AMZ (W2V)	NEWS (W2V)
BBCSPORT	4.5	8.2	9.5	**4.1**	5.0
TWITTER	33.3	33.7	29.3	**28.1**	28.3
RECIPE	47.0	51.6	52.7	47.4	**45.1**
OHSUMED	52.0	56.2	55.6	50.4	**44.5**
CLASSIC	5.3	5.5	4.0	3.8	**3.0**
REUTERS	4.2	4.6	7.1	9.1	**3.5**
AMAZON	12.3	13.3	13.9	7.8	**7.2**

For each baseline we use the Euclidean distance for kNN classification. All free hyperparameters were set with Bayesian optimization for all algorithms (Snoek et al., 2012). We use the open source MATLAB implementation "bayesopt.m" from Gardner et al. (2014).[3]

5.2. Document classification

Document similarities are particularly useful for classification given a supervised training dataset, via the kNN decision rule (Cover & Hart, 1967). Different from other classification techniques, kNN provides an interpretable certificate (*i.e.*, in the form of nearest neighbors) that allow practitioners the ability to verify the prediction result. Moreover, such similarities can be used for ranking and recommendation. To assess the performance of our metric on classification, we compare the kNN results of the WMD with each of the aforementioned document representations/distances. For all algorithms we split the train-

[3] http://tinyurl.com/bayesopt

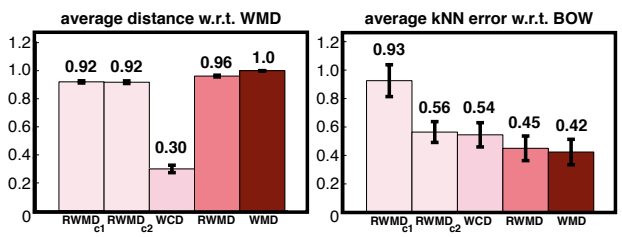

Figure 5. (*Left:*) The average distances of lower bounds as a ratio w.r.t WMD. (*Right:*) kNN test error results on 8 datasets, compared to canonical and state-of-the-art baseline methods.

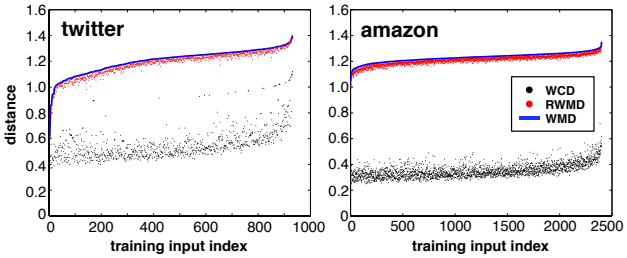

Figure 6. The WCD, RWMD, and WMD distances (sorted by WMD) for a random test query document.

ing set into a 80/20 train/validation for hyper-parameter tuning. It is worth emphasizing that BOW, TF-IDF, BM25 and WMD have no hyperparameters and thus we only optimize the neighborhood size ($k \in \{1, \ldots, 19\}$) of kNN.

Figure 3 shows the kNN test error of the 8 aforementioned algorithms on the 8 document classification datasets. For datasets without predefined train/test splits (BBCSPORT, TWITTER, RECIPE, CLASSIC, AMAZON) we averaged over 5 train/test splits and we report means and standard errors. We order the methods by their average performance. Perhaps surprisingly, LSI and LDA outperform the more recent approaches CCG and mSDA. For LDA this is likely because it is trained transductively. One explanation for why LSI performs so well may be the power of Bayesian Optimization to tune the single LSI hyperparameter: the number of basis vectors to use in the representation. Fine-tuning the number of latent vectors may allow LSI to create a very accurate representation. On all datasets except two (BBCSPORT, OHSUMED), WMD achieves the lowest test error. Notably, WMD achieves almost a 10% reduction in (relative) error over the second best method on TWITTER (LSI). It even reaches error levels as low as 2.8% error on classic and 3.5% error on REUTERS, even outperforming transductive LDA, which has direct access to the features of the test set. One possible explanation for the WMD performance on OHSUMED is that many of these documents contain technical medical terms which may not have a word embedding in our model. These words must be discarded, possibly harming the accuracy of the metric.

Figure 4 shows the average improvement of each method, relative to BOW, across all datasets. On average, WMD results in only 0.42 of the BOW test-error and outperforms all other metrics that we compared against.

5.3. Word embeddings.

As our technique is naturally dependent on a word embedding, we examine how different word embeddings affect the quality of k-nearest neighbor classification with the WMD. Apart from the aforementioned freely-available Google News *word2vec* model, we trained two other *word2vec* models on a papers corpus (*NIPS*) and a product review corpus (*AMZ*). Specifically, we extracted text from all NIPS conference papers within the years 2004-2013 and trained a Skip-gram model on the dataset as per Mikolov et al. (2013b). We also used the $340,000$ Amazon review dataset of Blitzer et al. (2006) (a superset of the *amazon* classification dataset used above) to train the review-based *word2vec* model. Prior to training we removed stop words for both models, resulting in 36,425,259 words for the NIPS dataset (50-dimensional) and 90,005,609 words for the Reviews dataset (100-dimensional) (compared to the 100 billion word dataset of Google News (*NEWS*)).

Additionally, we experimented with the pre-trained embeddings of the hierarchical log-bilinear model (*HLBL*) (Mnih & Hinton, 2009) and the model of Collobert & Weston (2008) (CW)[4]. The HLBL model contains $246,122$ unique 50-dimensional word embeddings and the Collobert model has $268,810$ unique word embeddings (also 50-dimensional). Table 2 shows classification results on all data sets except 20NEWS (which we dropped due to running time constraints). On the five larger data sets, the 3 million word Google NEWS model performs superior to the smaller models. This result is in line with those of Mikolov et al. (2013a), that in general more data (as opposed to simply relevant data) creates better embeddings. Additionally, the three word2vec (w2v) models outperform the HLBL and Collobert models on all datasets. The classification error deteriorates when the underlying model is trained on very different vocabulary (*e.g.* NIPS papers vs cooking recipes), although the performance of the Google NEWS corpus is surprisingly competitive throughout.

5.4. Lower Bounds and Pruning

Although WMD yields by far the most accurate classification results, it is fair to say that it is also the slowest metric to compute. We can therefore use the lower bounds from section 4 to speed up the distance computations. Figure 6 shows the WMD distance of all training inputs to two randomly chosen test queries from TWITTER and AMAZON in increasing order. The graph also depicts the WCD and RWMD lower bounds. The RWMD is typically very close to the exact WMD distance, whereas the cheaper WCD

[4]Both available at http://metaoptimize.com/projects/wordreprs/

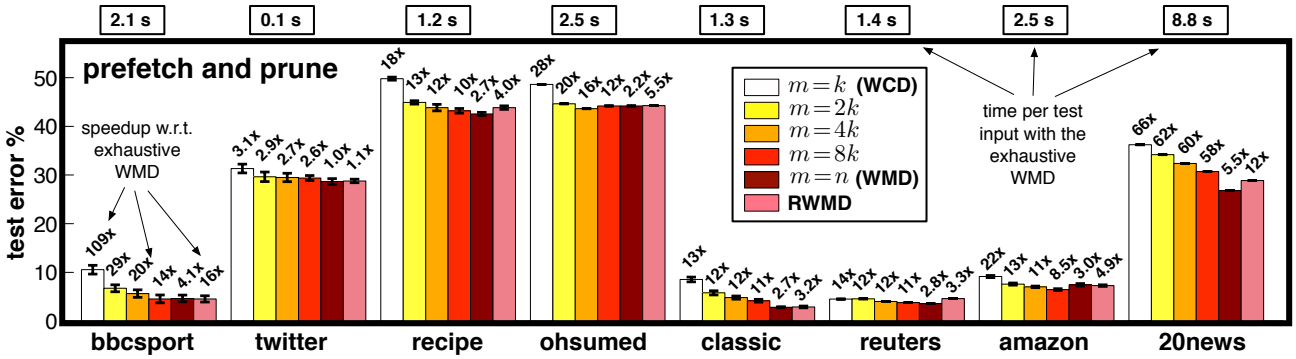

Figure 7. Test errors (in %) and speedups of the prefetch and prune algorithm as the document cutoff, m, changes. All speedups are relative to the exhaustive k-NN search with the WMD (shown on the very top). The error bars indicate standard error (if applicable).

approximation is rather loose. The tightness of RWMD makes it valuable to prune documents that are provably not amongst the k nearest neighbors. Although the WCD is too loose for pruning, its distances increase with the exact WMD distance, which makes it a useful heuristic to identify promising nearest neighbor candidates.

The tightness of each lower bound can be seen in the left image in Figure 5 (averaged across all test points). $\text{RWMD}_{C1}, \text{RWMD}_{C2}$ correspond to WMD with only constraints #1 and #2 respectively, which result in comparable tightness. WCD is by far the loosest and RWMD the tightest bound. Interestingly, this tightness does not directly translate into retrieval accuracy. The right image shows the average kNN errors (relative to the BOW kNN error) if the lower bounds are used directly for nearest neighbor retrieval. The two most left columns represent the two individual lower bounds of the RWMD approximation. Both perform poorly (worse than WCD), however their maximum (RWMD) is surprisingly accurate and yields kNN errors that are only a little bit less accurate than the exact WMD. In fact, we would like to emphasize that the average kNN error with RWMD (0.45 relative to BOW) still outperforms all other baselines (see Figure 4).

Finally, we evaluate the speedup and accuracy of the exact and approximate versions of the *prefetch and prune* algorithm from Section 4 under various values of m (Figure 7). When $m = k$ we use the WCD metric for classification (and drop all WMD computations which are unnecessary). For all other results we prefetch m instances via WCD, use RWMD to check if a document can be pruned and only if not compute the exact WMD distance. The last bar for each dataset shows the test error obtained with the RWMD metric (omitting all WMD computations). All speedups are reported relative to the time required for the exhaustive WMD metric (very top of the figure) and were run in paralell on 4 cores (8 cores for 20NEWS) of an Intel L5520 CPU with 2.27Ghz clock frequency.

First, we notice in all cases the increase in error through prefetching is relatively minor whereas the speedup can be substantial. The exact method ($m = n$) typically results in a speedup between $2\times$ and $5\times$ which appears pronounced with increasing document lengths (*e.g.* 20NEWS). It is interesting to observe, that the error drops most between $m = k$ and $m = 2k$, which might yield a sweet spot between accuracy and retrieval time for time-sensitive applications. As noted before, using RWMD directly leads to impressively low error rates and average retrieval times below $1s$ on all data sets. We believe the actual timing could be improved substantially with more sophisticated implementations (our code is in MATLAB) and parallelization.

6. Discussion and Conclusion

It is worthwhile considering why the WMD metric leads to such low error rates across all data sets. We attribute this to its ability to utilize the high quality of the *word2vec* embedding. Trained on billions of words, the *word2vec* embedding captures knowledge about text documents in the English language that may not be obtainable from the training set alone. As pointed out by Mikolov et al. (2013a), other algorithms (such as LDA or LSI) do not scale naturally to data sets of such scale without special approximations which often counteract the benefit of large-scale data (although it is a worthy area of future work). Surprisingly, this "latent" supervision benefits tasks that are different from the data used to learn the word embedding.

One attractive feature of the WMD, that we would like to explore in the future, is its interpretability. Document distances can be dissected into sparse distances between words, which can be visualized and explained to humans. Another interesting direction will be to incorporate document structure into the distances between words by adding penalty terms if two words occur in different sections of similarly structured documents. If for example the WMD metric is used to measure the distance between academic papers, it might make sense to penalize word movements between the *introduction* and *method section* more than word movements from one *introduction* to another.

Acknowledgments

KQW and MJK are supported by NSF grants IIA-1355406, IIS-1149882, EFRI-1137211. We thank the anonymous reviewers for insightful feedback.

References

Andoni, A. and Indyk, P. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *FOCS*, pp. 459–468. IEEE, 2006.

Aslam, J. A. and Frost, M. An information-theoretic measure for document similarity. In *SIGIR*, volume 3, pp. 449–450. Citeseer, 2003.

Blei, D. M., Ng, A. Y., and Jordan, M. I. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3: 993–1022, 2003.

Blitzer, J., McDonald, R., and Pereira, F. Domain adaptation with structural correspondence learning. In *EMNLP*, pp. 120–128. ACL, 2006.

Brochu, E. and Freitas, N. D. Name that song! In *NIPS*, pp. 1505–1512, 2002.

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

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

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.

Cover, T. and Hart, P. Nearest neighbor pattern classification. *Information Theory, IEEE Transactions on*, 13(1): 21–27, 1967.

Croft, W. B. and Lafferty, J. *Language modeling for information retrieval*, volume 13. Springer, 2003.

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

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.

Garcia, Vincent, Debreuve, Eric, and Barlaud, Michel. Fast k nearest neighbor search using gpu. In *CVPR Workshop*, pp. 1–6. IEEE, 2008.

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

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

Grauman, K. and Darrell, T. Fast contour matching using approximate earth mover's distance. In *CVPR*, volume 1, pp. I–220. IEEE, 2004.

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

Hearst, M. A. Multi-paragraph segmentation of expository text. In *ACL*, pp. 9–16. Association for Computational Linguistics, 1994.

Jojic, N. and Perina, A. Multidimensional counting grids: Inferring word order from disordered bags of words. In *UAI*. 2011.

Le, Quoc V and Mikolov, Tomas. Distributed representations of sentences and documents. In *ICML*, 2014.

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.

Ling, Haibin and Okada, Kazunori. An efficient earth mover's distance algorithm for robust histogram comparison. *Pattern Analysis and Machine Intelligence*, 29(5): 840–853, 2007.

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

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, 2013b.

Mikolov, T., Yih, W., and Zweig, G. Linguistic regularities in continuous space word representations. In *NAACL*, pp. 746–751. Citeseer, 2013c.

Mnih, A. and Hinton, G. E. A scalable hierarchical distributed language model. In *NIPS*, pp. 1081–1088, 2009.

Monge, G. *Mémoire sur la théorie des déblais et des remblais*. De l'Imprimerie Royale, 1781.

Nemhauser, G. L. and Wolsey, L. A. *Integer and combinatorial optimization*, volume 18. Wiley New York, 1988.

Ontrup, J. and Ritter, H. Hyperbolic self-organizing maps for semantic navigation. In *NIPS*, volume 14, pp. 2001, 2001.

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

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

Petterson, J., Buntine, W., Narayanamurthy, S. M., Caetano, T. S., and Smola, A. J. Word features for latent dirichlet allocation. In *NIPS*, pp. 1921–1929, 2010.

Quadrianto, N., Song, L., and Smola, A. J. Kernelized sorting. In *NIPS*, pp. 1289–1296, 2009.

Ren, Z., Yuan, J., and Zhang, Z. Robust hand gesture recognition based on finger-earth mover's distance with a commodity depth camera. In *Proceedings of ACM international conference on multimedia*, pp. 1093–1096. ACM, 2011.

Robertson, S. E. and Walker, S. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *Proceedings ACM SIGIR conference on Research and development in information retrieval*, pp. 232–241. Springer-Verlag New York, Inc., 1994.

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.

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

Salton, G. and Buckley, C. The smart retrieval systemexperiments in automatic document processing. 1971.

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

Sanders, N. J. Sanders-twitter sentiment corpus, 2011.

Schölkopf, B., Weston, J., Eskin, E., Leslie, C., and Noble, W. S. A kernel approach for learning from almost orthogonal patterns. In *ECML*, pp. 511–528. Springer, 2002.

Shirdhonkar, S. and Jacobs, D. W. Approximate earth movers distance in linear time. In *CVPR*, pp. 1–8. IEEE, 2008.

Snoek, J., Larochelle, H., and Adams, R. P. Practical bayesian optimization of machine learning algorithms. In *NIPS*, pp. 2951–2959, 2012.

Steyvers, M. and Griffiths, T. Probabilistic topic models. *Handbook of latent semantic analysis*, 427(7):424–440, 2007.

Turian, J., Ratinov, L., and Bengio, Y. Word representations: a simple and general method for semi-supervised learning. In *ACL*, pp. 384–394. Association for Computational Linguistics, 2010.

Wan, X. A novel document similarity measure based on earth movers distance. *Information Sciences*, 177(18): 3718–3730, 2007.

Yianilos, Peter N. Data structures and algorithms for nearest neighbor search in general metric spaces. In *Proceedings of ACM-SIAM Symposium on Discrete Algorithms*, pp. 311–321. Society for Industrial and Applied Mathematics, 1993.