Fast computation of document similarities using optimal transportation distances

Pierre Stock
École Normale Supérieure de Cachan, Paris
pierre.stock@ens-paris-saclay.fr

Abstract

We will focus on defining a reliable and fast-to-compute distance between two documents. A well-known approach consists in computing the optimal transportation distance between bag-of-words histograms for a given word dictionary. To allow for more flexibility and accuracy, word embedding metrics such as word2vec have been recently used to define the transportation cost between two words. However, the transportation problem becomes more and more expensive to solve when the size of the dictionary and thus the size of the documents increases: with more than a few hundred words, the computation cost becomes prohibitive. Our approach consisted in iteratively solving an entropic-regularized version of the transportation problem using the word2vec metric. We implemented a GPU-based version of this algorithm, allowing an extremely fast computation of distances between entire datasets of large documents.

1 Introduction

Measuring the similarity between documents can often be seen as a preprocessing step to some learning tasks that have far-reaching applications, like news categorization and clustering [1, 2], sentiment analysis on customer reviews or social networks posts [3], song identification [4] or multilingual document matching [5].

But can we define a distance or at least a reliable similarity measure between two documents ? Given a word dictionary $\mathcal{D}$, a simple approach to that question would be to use the bag-of-words (BoW) representations of those documents and to compute their similarity using a kernel (e.g. linear or Hellinger).

While this intuitive bag-of-features approach is widely used in computer vision [6], it suffers in our case from a major drawback : words have synonyms. We can assume that a customer using the words "unhappy, expensive, fail" and another using the words "waste, bad, disappointed" have rather the same (negative) opinion about a product, even if their histograms don't share any word in common.

Optimal transportation distances [7] provide a natural way to handle this problem, because they are parametrized by a ground metric, which allows to measure the similarity between two features (e.g. words). Given the good performance of optimal transportation distances in most data analysis tasks involving bag-of features, those distances have generated interest but their computational cost remains prohibitive. When no restrictions are placed upon the ground metric, the underlying linear program computing those distances scales at least in $\mathcal{O}(d^3 \log(d))$ [8].

In this paper, we use a new family of transportation distances introduced by Cuturi [9] that look at the transportation problem from a maximum-entropy perspective by regularizing it with an entropic term. We use Mikolov's word embedding word2vec [10] as the ground metric, and adapted Sinkhorn-Knopp's fixed point iteration proposed by Cuturi [9] to compute the optimal transportation distance between two documents. Our main contributions are a GPU implementation of this algorithm[1] in Python using Tensorflow and a benchmark on some classical datasets.

[1] Available at https://github.com/pierrestock/document-distances

2 Related Work

Learning new low-dimensional document representations that further allow to compute a distance between them has been widely studied. The most classical representations have been introduced by Salton [11] and consisted in defining different combinations of corpus statistics like terms frequencies or counts and to weight and normalize them, yielding for instance bag-of-words or term frequency-inverse document frequency representations.

However, those representations suffer from their frequent near-orthogonality [2], preventing us from properly differentiating the texts and thus computing a reliable similarity measure. A more flexible approach is to use optimal transportation distances, which have been are widely studied by Villani in [7], but are also known as Earth Mover's following the work of Rubner *et al.* [6] who successfully applied them to computer vision.

Word2vec

An important breakthrough in the last years has been the introduction of the word2vec model by Mikolov et al. [10]. The subsequent freely available model generates low-dimensional word embeddings using a 2-layers shallow neural network that has been trained on approximately 100 billion words.

The authors demonstrated that the model learned high-quality word representations that are able to capture precise syntactic and semantic meanings: for instance, the relationship vec(*king*) - vec(*man*) + vec(*women*) has an algebraic meaning since the obtained vector is very close to vec(*queen*). See Fig 1 where semantically close words are in fact spatially close[2] (don't hesitate to zoom in).

More recently, Quoc and Mikolov [12] introduced doc2vec, an unsupervised algorithm that learns fixed-length feature representations from variable-length pieces of texts, such as sentences, paragraphs, and documents. However, their model has to be re-trained depending on the dataset under study.

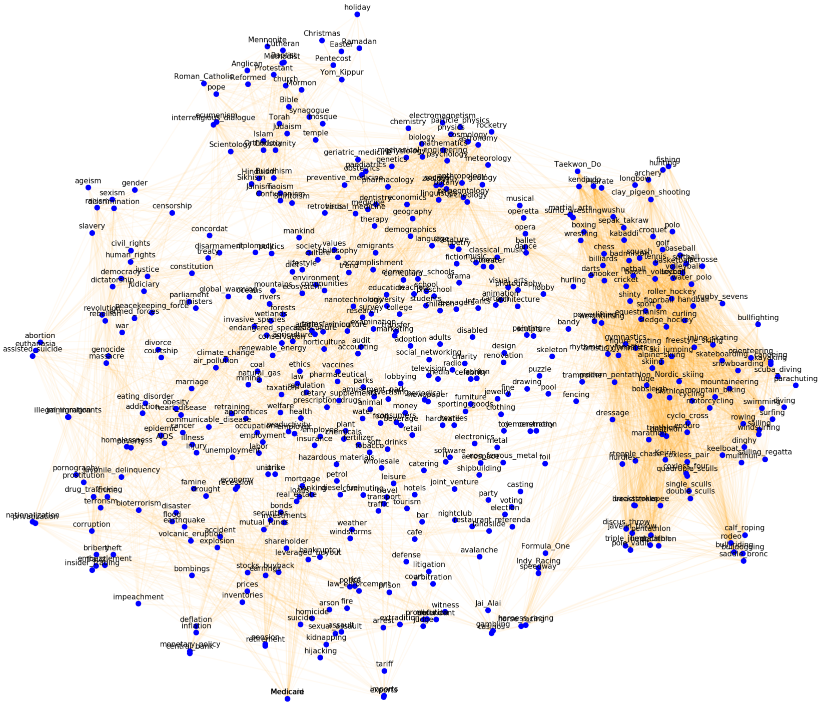

Figure 1: t-SNE projection of some words using the word2vec embedding space

[2] Source: `http://www.trivial.io/word2vec-on-databricks/`

Word Mover's Distance

Leveraging the word2vec embedding, Kusner *et al.* [3] introduced the the Word Mover's Distance (WMD) that measures the dissimilarity between two text documents as the optimal transportation distance using the euclidian distance naturally provided by the embedding space as the ground metric.

Their model has the advantage to be hyperparameter free and to be more flexible by adapting the dictionary to each document comparison (considering all the unique words in the set of the two documents) . However, because of the computational cost of the optimal transportation problem [8], it is not naturally scalable to datasets of large documents.

To circumvent this issue, the authors introduced several cheap lower bounds for their problem that allow to only compute the WMD of a small number of documents by discarding all the others. For very large documents (e.g. thousands of unique words), their approach remains computationally expensive.

Sinkhorn's Algorithm

Cuturi [9] considered the transportation problem from a maximum-entropy perspective and introduced an entropic regularization term to the transportation objective, and showed that the subsequent transportation problem could be solved very efficiently using matrix iterations, leveraging Sinkhorn-Knopp's matrix scaling algorithm and without any restriction placed upon the ground metric. Moreover, the algorithm (called Sinkhorn's algorithm) can be easily parallelized, allowing for fast and high-scale GPGPU computations.

Sinkhorn's algorithm has been further studied by Benamou *et al.* [13] who present a unified framework to numerically solve entropic approximations of several generalized optimal transport problems, using iterative Bergman projections and Dykstra's algorithm. Sinkhorn's algorithm has also been documented by Peyré in its Numerical Tours of Signal Processing [14].

3 Computing Document Distances

3.1 Notations

Given a fixed word dictionary $\mathcal{D} = \{w_1, \dots, w_n\}$ of size n and two documents composed uniquely of words belonging to $\mathcal{D}$, we can compute their respective bag-of-word representations as histograms of word counts p and q, where p_i and q_i are the respective word counts associated to w_i.

Both p and q are then L_1 normalized and thus belong to the simplex in $\mathbf{R}^n$:

$$\Sigma_n = \{p \in \mathbf{R}^n \mid p^T \mathbf{1}_n = 1\}$$

where $\mathbf{1}_n = (1, \dots, 1)^T \in \mathbf{R}^n$. We will refer to p and q as the original documents with a slight abuse of notation, and will thus define a similarity measure between p and q.

The polytope of couplings or policy matrixes between p and $q \in \Sigma_n$ is defined as:

$$\Pi(p, q) = \{\pi \in \mathbf{R}_+^{n \times n} \mid \pi \mathbf{1}_n = p, \pi^T \mathbf{1}_n = q\}$$

and represents the set of all nonnegative matrixes whose lines sum to p and whose columns sum to q. The matrix π is often called the policy matrix as its coefficients $\pi_{i,j}$ intuitively denotes the *amount* of the word w_i in the first document which will be *transferred* to the word w_j appearing in the second document (those notions will be further formalized in this section).

We denote $C \in \mathbf{R}^{n \times n}$ the cost matrix, where $C_{i,j} = C_{j,i}$ represents the cost of traveling from word i to word j. More details on the construction of C in section 3.2.4 dedicated to the word embedding.

The entropy of a policy matrix $\pi \in \Pi(p, q)$ is defined as:

$$E(\pi) = -\sum_{i,j=1}^{n} \pi_{i,j}(\log(\pi_{i,j}) - 1)$$

with the convention $0 \log(0) = 0$ taking into account the fact that the word w_i may not appear in the first document ($p_i = 0$) or the word w_j may not appear in the second. Note that we added the constant $\sum_{i,j} \pi_{i,j} = 1$ (because both p and q are normalized) to the classical definition of the entropy for reasons that will be clear in the next sections.

The Kullback-Leibler divergence between $\pi \in \mathbf{R}_+^{n\times n}$ and $\gamma \in (\mathbf{R}_+^*)^{n\times n}$ is defined as follows:

$$\mathrm{KL}(\pi|\gamma) = \sum_{i,j=1}^{n} \pi_{i,j} \left(\log\left(\frac{\pi_{i,j}}{\gamma_{i,j}}\right) - 1 \right)$$

and given a convex set $S \subset \mathbf{R}^{n\times n}$, we define the projection according to the Kullback-Leibler divergence as:

$$P_S^{\mathrm{KL}}(\gamma) = \operatorname*{argmin}_{\pi \in S} \mathrm{KL}(\pi|\gamma)$$

We finally denote by $\odot$ and $\oslash$ the entry-wise multiplication (resp. division) of two matrixes when those operations can be performed.

3.2 Optimal Transportation Distances

Given a cost matrix C, the cost of mapping p to q using the transportation policy π can be computed as $\langle \pi, C\rangle$. The classical transportation problem consists in finding the transportation policy $\pi^\star$ that minimizes this transportation cost:

$$\pi^\star \in \operatorname*{argmin}_{\pi\in\Pi(p,q)} \langle \pi, C\rangle$$

The optimal transportation distance $d(p, q)$ between p and q is then defined as:

$$d(p,q) = \min_{\pi\in\Pi(p,q)} \langle \pi, C\rangle = \langle \pi^\star, C\rangle$$

Villani showed that d was indeed a distance when C is a metric matrix, that is, $C_{i,j} \geq 0$, $C_{i,j} = C_{j,i}$, $C_{i,i} = 0$ and $C_{i,j} \leq C_{i,k} + Ck, j$ for all i, j and $k \in \{1, \ldots, n\}$. We refer to the monograph [7] for more details.

3.2.1 Entropic Regularization

Following Cuturi [9], we introduce a regularization term in the objective function, yielding the entropic-regularized transportation problem

$$T_\lambda(p,q) = \min_{\pi\in\Pi(p,q)} \langle \pi, C\rangle - \lambda E(\pi) \tag{P}$$

Intuitively, the regularization term forces the transportation policy π to be smooth as the regularization strength λ increases, and helps to stabilize the computation by forcing the solutions to have a spread support.

In addition to those regularization properties, the entropic term defines a strongly convex problem with a unique solution [13], and allows for nice computation properties. Indeed, the problem 6 can be recast as a projection according to the Kullback-Leibler divergence:

$$T_\lambda(p,q) = \lambda \min_{\pi\in\Pi(p,q)} \mathrm{KL}(\pi|\xi) = \lambda P_{\Pi(p,q)}^{\mathrm{KL}}(\xi) \tag{P'}$$

with $\xi = e^{-C/\lambda}$ where the exponential is computed element-wise. A straightforward computation shows that ξ minimizes the unconstrained problem:

$$\min_{\pi \in \mathbf{R}_+^{n \times n}} \langle \pi, C \rangle - \lambda E(\pi)$$

This explains somewhat why the constant 1 has ben added to the definition of the entropy. Intuitively, because ξ minimizes the unconstrained problem, we want to find the feasible policy $\pi \in \Pi(p,q)$ that is the closest to ξ in the sense of the Kullback-Leibler divergence to solve the constrained problem 6.

Supposing we found the minimizer $\pi^\star$, the distance between p and q can then easily be computed as:

$$d_\lambda(p,q) = \langle \pi^\star, C \rangle$$

Note that necessarily, $d_\lambda(p,q) \geq d(p,q)$ because of the penalty term.

In the next section, we will introduce the notions needed to iteratively solve the recast problem P', namely iterative Bergman projections. An extensive study of those notions and a proof of convergence can be found in [15].

3.2.2 Iterative Bergman Projections

We want to compute $P_S^{\mathrm{KL}}(\xi)$ where S is a non-empty intersection of affine constraint sets:

$$S = S_1 \cap \cdots \cap S_K$$

This projection can be iteratively solved, starting from $\pi_0 = \xi$ and by iterating

$$\forall i \geq 0,\ \pi_{i+1} = P_{S_i}^{\mathrm{KL}}(\pi_i)$$

where the constraint sets are indexed modulo K, *i.e.* $S_{i+K} = S_i$. Then, π_i converges to $P_S^{\mathrm{KL}}(\xi)$ [15].

When the convex sets are not affine subspaces, iterative Bergman projections do not always converge. Dykstra's algorithm, a more general procedure can be used instead. We refer to [13, 15] for further details.

3.2.3 Sinkhorn's Algorithm

To apply iterative Bergman projections to solve problem P', we need to express $\Pi(p,q)$ as a non-empty intersection of affine constraint sets. This can be done using the following sets:

$$S_1 = \{\pi \in \mathbf{R}_+^{n \times n} \mid \pi \mathbf{1}_n = p\} \quad \text{and} \quad S_2 = \{\pi \in \mathbf{R}_+^{n \times n} \mid \pi^T \mathbf{1}_n = q\}$$

and by noticing that $\Pi(p,q) = S_1 \cap S_2$.

Then, we need a close form for the projections according to the Kullback-Leibler divergence on those sets. They can be easily computed as:

$$P_{S_1}^{\mathrm{KL}}(\pi) = \mathrm{diag}\left(p \oslash (\pi \mathbf{1}_n)\right) \pi \quad \text{and} \quad P_{S_2}^{\mathrm{KL}}(\pi) = \pi \mathrm{diag}\left(q \oslash \left(\pi^T \mathbf{1}_n\right)\right)$$

for any $\pi \in \mathbf{R}_+^{n \times n}$, so that projecting on either S_1 or S_2 simply amounts to properly normalize the rows or the columns of π.

This property allows for a fast implementation of iterative Bergman projections using only basic operations on matrixes and row vectors (matrix or element-wise multiplication and division) as noted in [13].

The key remark is that the iterates π_l always has the form:

$$\pi_i = \mathrm{diag}(a_i) \xi \mathrm{diag}(b_i)$$

where the vectors a_i and $b_i \in \mathbf{R}^n$ satisfy $b_0 = \mathbf{1}_n$ and

$$a_i = p \oslash (\xi b_i) \quad \text{and} \quad b_{i+1} = q \oslash (\xi^T a_i)$$

This algorithm is called Sinkhorn-Knopp's fixed point iteration and was first introduced in [16]. It is summarized below and takes as inputs the two normalized histograms p and q, the regularization parameter λ, the cost matrix C and the number of iterations to perform. Note that it suffices to replace replace the vectors by matrixes of vectors to compute simultaneously the distances between large datasets of documents, as they all share the same dictionary and the same cost matrix.

The gains to using GPUs appears clearly. Assume the size of the dictionary is $n =$10,000 and we want to compute all the 4,950 pairwise similarities between $d = 100$ documents. Then, we could iterate in place in the GPU, which will be very efficient for the highly parallel task of repeatedly performing elementary operations between matrixes of size $n \times d(d-1)/2$. We implemented this algorithm to perform all pairwise comparisons between two datasets of documents in Python using Tensorflow and ran it on a NVIDIA K80 card. Further documentation is available online.

Algorithm 1 Sinkhorn's algorithm

```
input p, q, lambda, C, niter
xi = exp($-$C $*$ C / lambda)
b = ones(1, size(p))
for i = 1..niter do
    a = p / (xi $*$ b)
    b = q / (xi.T $*$ a)
end for
pi = diag(a) $*$ xi $*$ diag(b)
d = sum(C $*$ pi)
return d
```

3.2.4 Word2Vec Embedding

Now that we are able to compute the solution of the problem 6 and thus the distance between two documents $d_\lambda(p, q)$, we need to address one last issue regarding the cost matrix C. Recall that its coefficients $C_{i,j}$ indicate the distance between the words w_i and w_j.

We will leverage the recent works of Mikolov *et al*. [10] by embedding the words in a 300-dimensional space. The distance between two words can then be computed using a given norm (e.g. L_p with $p > 0$) in that space. By default, we will use the euclidian L_2 norm, but we will study the influence of the norm in section 4.

The authors trained a 2-layers shallow neural network on approximately 100 billion words to predict the context of a given word, e.g. to predict of to words often appear nearby. The embedding is obtained as the weights of hidden layer. Interestingly, the model learned high-quality word representations that are able to capture precise syntactic and semantic meanings. See Fig 2 for an illustration on Countries and Capitals.

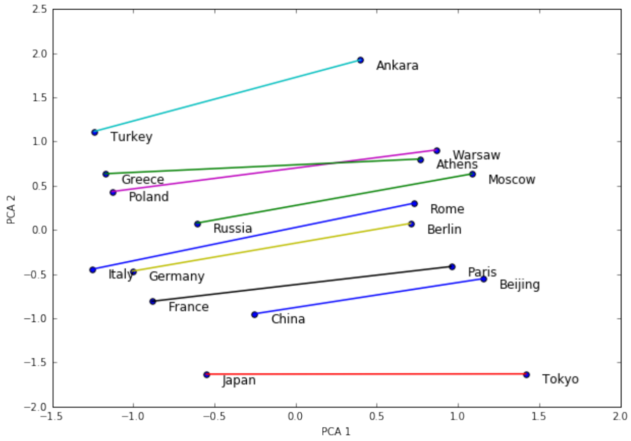

Figure 2: PCA projections on some Country/Capital pairs

4 Results

We present the main results obtained using our GPU implementation of Sinkhorn's algorithm and study the influence of the regularization parameter and the embedding metric.

4.1 Setup

Dictionary. We used the 10,000 most common words in English as determined by n-gram frequency analysis of the Google's Trillion Word Corpus[3]. According to analysis of the Oxford English Corpus, the 7,000 most common English lemmas account for approximately 90% of usage. In practice and unless mention of the contrary, we use a 5,000 word corpus.

Distance matrix. We used the freely-available pre-trained word2vec model[4] to compute once for all the distance matrix using the L_p norm for different values of p. Note that p may not necessarily be an integer, we assume $p > 0$. By default and unless contrary indication, we use $p = 2$.

Datasets. We used the Reuters-21578 dataset[5] which is widely used in text categorization research. It consists in documents appeared on the Reuters newswire in 1987 and were manually classified by personnel from Reuters Ltd. We use its R52 version, comprising 52 categories, 6,532 training documents and 2,568 test documents.

Preprocessing. The preprocessing steps include lowering all the words (no uppercases), removing the digits, tokenizing the documents and finally removing the stopwords in the SMART stopword list defined in [11]. Stopwords are words like *the* or *and* that appear very frequently in texts but that carry little to no meaning.

Implementation. We implemented Sinkhorn's algorithm in Python 3.5 using Tensorflow 0.12 and ran it on a NVIDIA K80[6]. We gained a factor 100 in time over the entire computation compared to CPU implementation.

Number of iterations. We use a tolerance of $\varepsilon = 0.01$ on the L_2 norm of the difference of two successive iterations to calibrate the number of iterations on a few samples of a dataset and then ran the algorithm on the entire dataset while keeping this determined number of iterations.

Evaluating the results. We evaluated the k-NN performance of our algorithm for different values of k ranging from $k = 1$ to 20 on the test sets.

4.2 Toy example

We ran our implementation on a toy example made of 4 news articles from the New York Times (available online). Two articles were about sports (one about football, the other about tennis), and two others about politics (domestic and foreign).

As expected, the articles are more similar to the other article belonging to their category. Note also that beyond a certain value of the regularization parameter, the documents are more similar and the similarity order may change due to the entropic term favourising minimum information.

On this toy example, the regularization parameter seems to play a major role: not properly tuning it may change the classification result, whereas the embedding norm parameter only seems to contribute to better separating the documents between them.

[3] Available at `https://github.com/first20hours/google-10000-english`

[4] Available at `https://code.google.com/archive/p/word2vec/`

[5] Available at `http://ana.cachopo.org/datasets-for-single-label-text-categorization`

[6] We used a p2.xlarge instance on AWS thanks to the GitHub Student Developer Pack Program

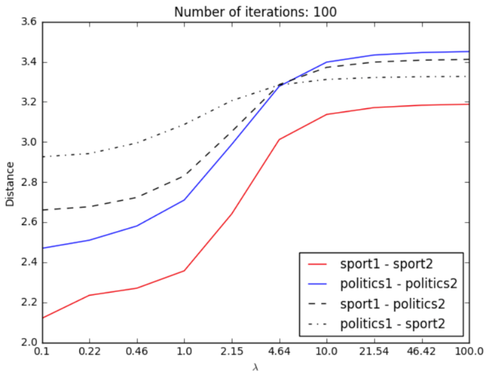

Figure 3: Influence of λ

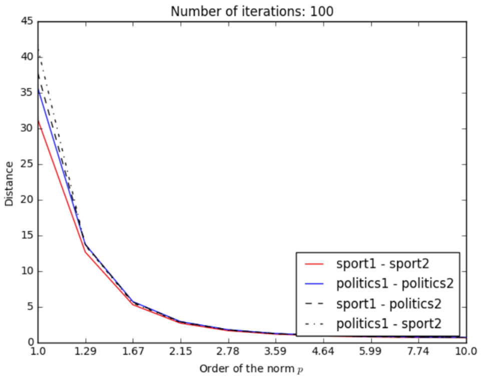

Figure 4: Influence of the order norm

4.3 Influence of the Regularization Parameter λ

We observe that the value $k = 10$ gives the best classification error (20.08%) for a small value of the regularization parameter.

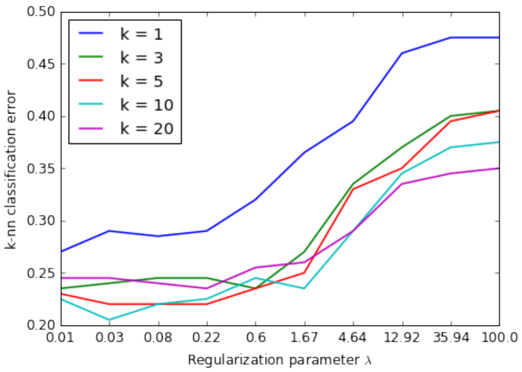

Figure 5: Influence of the regularization parameter λ

4.4 Influence of the Embedding Metric

The embedding metric order p should be as small as possible while keeping the norm convex(e.g. $p = 1$), whereas values of $p > 3$ are not substantially changing the classification error.

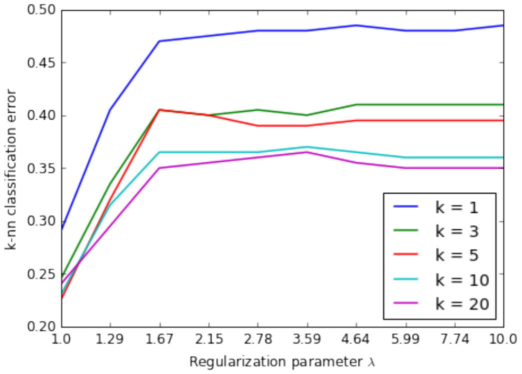

Figure 6: Influence of the norm in the embedding space

5 Conclusion and perspectives

As we have seen in this paper, entropic regularization of the optimal transportation problem has nice theoretical properties: it transforms the problem into a strongly convex one, thus ensuring the unicity of the solution, and it smoothes the optimal policy matrix by forcing it to have a sparse support as λ gets large, thus seeking to maximize more and more the entropy of the solution.

Entropic regularization also presents some very nice practical properties, allowing to compute the solution of the optimal transportation problem using a Bergman iteration scheme. The obtained algorithm is easily implementable on GPUs to allow very fast and large-scale computations of distances between documents.

Using our implementation of this algorithm in Python, we have shown that the regularization parameter has to be tuned carefully according to type of document to classify, and also that the embedding metric played an important role in trying to separate the documents.

Possible extensions of this work include improving the fixed dictionary by including celebrity names for example and computing its optimal size and testing the performance of this algorithm on other datasets or other tasks like sentiment analysis.

The main intrinsic limitation of this approach is that Sinkhorn's algorithm becomes unstable when the regularization parameter is small, because the algorithm involves computing terms of the form $e^{-x/\lambda}$, effect that is aggravated if we use float32 computations on the GPU to further speed up the computations.

An other limitation is that BoW features loose the ordering of the words in the documents, ordering that is only partially retrieved using the word2vec embedding by associating short distances to words that are similar, *i.e.* that often appear nearby in the same context. Long Short-Term Memory Networks have been introduced to take the context of a given word more into account.

References

[1] Jorg Ontrup and Helge J. Ritter. Hyperbolic self-organizing maps for semantic navigation. In Thomas G. Dietterich, Suzanna Becker, and Zoubin Ghahramani, editors, *NIPS*, pages 1417–1424. MIT Press, 2001.

[2] Derek Greene and Padraig Cunningham. Practical solutions to the problem of diagonal dominance in kernel document clustering. In William W. Cohen and Andrew Moore, editors, *ICML*, volume 148 of *ACM International Conference Proceeding Series*, pages 377–384. ACM, 2006.

[3] Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Q. Weinberger. From word embeddings to document distances. *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 957–966, 2015.

[4] Eric Brochu and Nando de Freitas. "name that song!" A probabilistic approach to querying on music and text. In *Advances in Neural Information Processing Systems 15 [Neural Information Processing Systems, NIPS 2002, December 9-14, 2002, Vancouver, British Columbia, Canada]*, pages 1505–1512, 2002.

[5] Novi Quadrianto, Alexander J. Smola, Le Song, and Tinne Tuytelaars. Kernelized sorting. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(10):1809–1821, 2010.

[6] Yossi Rubner, Carlo Tomasi, and Leonidas J. Guibas. A metric for distributions with applications to image databases. In *Proceedings of the Sixth International Conference on Computer Vision*, ICCV '98, pages 59–, Washington, DC, USA, 1998. IEEE Computer Society.

[7] Cédric Villani. *Optimal transport : old and new*. Grundlehren der mathematischen Wissenschaften. Springer, Berlin, 2009.

[8] Ofir Pele and Michael Werman. Fast and robust earth mover's distances. In *ICCV*, 2009.

[9] Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. *Advances in Neural Information Processing Systems 26*, pages 2292–2300, 2013.

[10] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. *Advances in Neural Information Processing Systems 26*, pages 3111–3119, 2013.

[11] Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. pages 513–523, 1988.

[12] Quoc V. Le and Tomas Mikolov. Distributed representations of sentences and documents. *CoRR*, abs/1405.4053, 2014.

[13] Jean-David Benamou, Guillaume Carlier, Marco Cuturi, Luca Nenna, and Gabriel Peyré. Iterative Bregman Projections for Regularized Transportation Problems. *SIAM Journal on Scientific Computing*, 2(37):A1111–A1138, 2015.

[14] G. Peyré. The numerical tours of signal processing - advanced computational signal and image processing. *IEEE Computing in Science and Engineering*, 13(4):94–97, 2011.

[15] Heinz H. Bauschke and Adrian S. Lewis. Dykstra's algorithm with bregman projections: a convergence proof. *Optimization*, 48:409–427, 1998.

[16] Richard Sinkhorn. A relationship between arbitrary positive matrices and doubly stochastic matrices. *Ann. Math. Statist.*, 35(2):876–879, 06 1964.