Distance geometry for word embeddings

Sammy Khalife, Douglas S Gonçalves, Leo Liberti

Distance geometry for word embeddings

Sammy Khalife[1], Douglas S. Gonçalves[2], and Leo Liberti[1]

[1] LIX, CNRS, Ecole Polytechnique, Institut Polytechnique de Paris, 91128 Palaiseau, France
[2] MTM/CFM - Universidade Federal de Santa Catarina, 88040-900, Florianópolis, Brazil
khalife@lix.polytechnique.fr douglas@mtm.ufsc.br
liberti@lix.polytechnique.fr

Abstract. Many machine learning algorithms rely on vector representations as input. In particular, natural language word vector representations that encode semantic information can be constructed using several different methods, all based on solving an unconstrained optimization problem using stochastic gradient descent. Traditionally, these optimization formulations arise either from word co-occurrence based models (e.g word2vec, GloVe, fastText), or encoders combined with a masked language model (e.g BERT). In this work we propose word embedding methods based on the Distance Geometry Problem (DGP): find object positions based on a subset of their pairwise distances. Considering the empirical Pointwise Mutual Information (PMI) as an inner product approximation, we discuss two algorithms to obtain approximate solutions of the underlying Euclidean DGP on large instances. The resulting algorithms are considerably faster than state-of-the-art algorithms such as GloVe, fastText or BERT, with similar performance for classification tasks. The main advantage of our approach for practical use is its significantly lower computational complexity, which allows us to train representations much faster with a negligible quality loss, a useful property for domain specific corpora.

1 Introduction

In recent years, the most successful algorithms for Natural Language Processing (NLP) tasks (e.g. text classification, machine translation, named entity recognition) rely on vector representations of words and sentences constructed with a variety of approaches.

First, following the first vector space models based on index terms (TF-IDF, Software: SMART [Salton, 1971], Lucène [Hatcher and Gospodnetic, 2004]), co-occurrence based models in the early 2010s improved empirical performance for NLP tasks, motivating a geometric conjecture relating analogies on word pairs to proximity between the corresponding word vector differences, as advertised in [Pennington et al., 2014]. This conjecture was studied in [Arora et al., 2016] using a connection between scalar products of word vectors and pointwise mutual information (PMI). Despite the uncertain nature of the assumptions supporting this conjecture, as discussed in [Khalife et al., 2019], the property connecting scalar product of word vectors and PMI hold with high probability at infinity (i.e as the vocabulary size becomes sufficiently large). These properties popularized several representations obtained with methods as word2vec, GloVe or other similar co-occurrence based models [Mikolov et al., 2013b, Pennington et al., 2014, Arora et al., 2017].

Then, to overcome polysemy caused by *static* word embeddings, the family of *dynamic representations* (for which a word can be attributed different vectors, depending on its context) have gained momentum, also due to the increased use of deep learning methods. For instance, ELMo [Peters et al., 2018], and BERT [Devlin et al., 2019] representations are based on bidirectional encoders combined with masked language models incorporating supplementary information such as position, segment (subword information) and improved significantly the empirical performance for a variety of NLP tasks.

Most of these constructions rely on an unconstrained minimization of a loss function using stochastic gradient descent. In this paper we propose a new method for constructing word vectors, based on Euclidean Distance Geometry (DG). The fundamental problem of DG consists in identifying point positions from information about a subset of their pairwise distances [Liberti et al., 2014]. The DG literature provides several tools to address this problems in many situations.

More specifically, we use DG based methods in order to develop faster word vectors construction algorithms. Furthermore, we show empirically that such word vectors behave well on extrinsic tasks [Melamud et al., 2016] such as text classification.

The rest of this paper is organized as follows. Sect. 2 briefly reviews some DG concepts useful to devise our algorithms. Sect. 3 describes the word co-occurrence model and introduce DG methods for word embeddings. These methods are compared to the state-of-the-art in terms of the training model and computational complexity. Sect. 4 shows the performance of the methods on intrinsic and text classification tasks. Conclusions are given in Section 5.

2 Distance geometry

Given an integer $K > 0$ and a simple, undirected and edge-weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, d)$ where $d : \mathcal{E} \to \mathbb{R}_+$, the Distance Geometry problem (DGP, [Liberti et al., 2014]) consists in finding a map $v : \mathcal{V} \to \mathbb{R}^K$, such that

$$\forall \{i, j\} \in \mathcal{E}, \quad \|v_i - v_j\|^2 = d_{ij}^2 \tag{1}$$

where $\|.\|$ denotes the Euclidean norm, $v_i := v(i) \in \mathbb{R}^K$, for all $i \in \mathcal{V}$, and $d_{ij} := d(\{i, j\})$, for all $\{i, j\} \in E$. Let us denote the number of vertices by $n = |\mathcal{V}|$ and the inner product by $\langle \cdot, \cdot \rangle$. A solution for Eq. (1) is called *valid realization.*

The DGP is known to be NP-Hard [Saxe, 1979]. The most relevant polynomial time case is that of complete graphs, which correspond to fully defined Distance Matrices $D = (d_{ij}^2)$. We say that D is an Euclidean Distance Matrix (EDM) when Eq. (1) admits a solution for some dimension K. In this case we can solve Eq. (1) or determine its infeasibility by a process similar to Classic Multidimensional Scaling and Principal Components Analysis (PCA) [Vidal et al., 2016].

Let $\mathbf{1}$ denote the vector of ones of appropriate dimension. For a square matrix Z, $\text{diag}(Z)$ denotes a vector on the diagonal elements of Z. From the relation between the inner product and Euclidean norm:

$$\|v_i - v_j\|^2 = -2\langle v_i, v_j \rangle + \|v_i\|^2 + \|v_j\|^2. \tag{2}$$

Assuming that $\sum_i v_i = 0$, we can define a linear isomorphism $\mathcal{K}$ from the space of symmetric centered matrices $\mathbb{S}_C = \{Y \in \mathbb{R}^{n\times n} \ : \ Y = Y^\top,\ Y\mathbf{1} = 0\}$ to the space of symmetric null diagonal matrices $\mathbb{S}_H = \{Z \in \mathbb{R}^{n\times n} \ : \ Z = Z^\top,\ \mathrm{diag}(Z) = 0\}$, such that $D = \mathcal{K}(G)$ whenever $D_{ij} = \|v_i - v_j\|^2$ and $G_{ij} = \langle v_i, v_j\rangle$ [Al-Homidan and Wolkowicz, 2005]. Such a linear transformation is defined by

$$\mathcal{K}(G) = -2G + \mathbf{1}\mathrm{diag}(G)^\top + \mathrm{diag}(G)\mathbf{1}^\top.$$

Its inverse is given by

$$\mathcal{K}^{-1}(D) = -\frac{1}{2}JDJ \tag{3}$$

where $J = I - (1/n)\mathbf{1}\mathbf{1}^\top$ is known as centering matrix.

Due to this one-to-one correspondence, when all pairwise distances are available, the problem of finding $v : \mathcal{V} \to \mathbb{R}^K$ such that $\forall\ \{i, j\},\ \|v_i - v_j\|^2 = D_{ij}$ is equivalent to finding $v : \mathcal{V} \to \mathbb{R}^K$ such that $\forall\ \{i, j\},\ \langle v_i, v_j\rangle = G_{ij}$.

A remarkable result in DG is Schoenberg's theorem [Schoenberg, 1935, Gower, 1982, Dokmanic et al., 2015], which states that D is Euclidean if and only if $G = (-1/2)JDJ$ is positive semidefinite (PSD). Moreover, if G is PSD, then it is a genuine Gram matrix (matrix of inner products). Let $r = \mathrm{rank}(G)$. A solution for Eq. (1) is given by $V = \sqrt{\Lambda_r}Q_r^\top$, where $G = Q\Lambda Q^\top$ is the eigendecomposition of G, Λ_r is a $r \times r$ diagonal matrix with the top r eigenvalues of G, and the columns of Q_r contain the corresponding eigenvectors. If $K \geq r$, V is a solution of Eq. (1). Else, if $K < r$, we can choose among the PSD matrices X of rank $\leq K$, one that minimizes $\|X - G\|_F$. A solution is given by $Q_K\Lambda_K Q_K^\top$, where Λ_K is diagonal with top K eigenvalues of G and $Q_K \in \mathbb{R}^{K\times n}$ contains the corresponding eigenvectors in its columns. Thus, an *approximate* realization for Eq. (1), in dimension K, is given by $V = \sqrt{\Lambda_K}Q_K^\top$.

Last, but not least, if D is not an EDM (e.g. because some d_{ij} comes from a noisy measurement), then G is not PSD. Still, a solution in the least-squares sense is provided by $V^+ = \sqrt{\Lambda_K^+}Q_K^\top$, where $\Lambda_K^+ = \max(\Lambda_K, 0)$, where the $\max(\cdot,\cdot)$ is componentwise. The approximate realization V^+ is a solution of the following optimization problem

$$\min_{V\in\mathbb{R}^{K\times n}} \quad \|V^\top V - G\|_F^2 = \sum_i \sum_j (\langle v_i, v_j\rangle - G_{ij})^2 \tag{4}$$

where v_i is the i-th column of $V = (v_1\ v_2\ \ldots\ v_n)$.

Since this approach relies on spectral decomposition of a matrix of order n, in general, its complexity is bounded by $O(n^3)$ [Golub and Van Loan, 1996]. See [Demmel et al., 2007] for better estimates.

3 Methodology

A natural question, which is preliminary to the use of DG methods, is whether there exists a distance between words that measures their "semantic difference". Using DG methods, such a distance, even if only partially defined, would yield a set of word vectors satisfying the property: (A) two words are semantically correlated if their corresponding vectors

are close. Here, semantic correlation can be interpreted loosely (e.g synonymy, antonym, or more complicated forms of semantic correlation). However, a function verifying the property (A) may not satisfy the distance axioms. Furthermore, as discussed in [Globerson et al., 2007], co-occurrence rates also do not satisfy metric constraints. It is more reasonable to consider the statistical nature of the co-occurrence data [Turney and Pantel, 2010], and to interpret observed object pairs i and as drawn from a joint distribution that is determined by distances or inner products between vectors of the underlying low-dimensional embedding.

In this work, similarly to [Globerson et al., 2007], we consider the following model:

$$p(i,j) \propto p(i)p(j)e^{\langle v_i, v_j \rangle} \tag{5}$$

where $p(i,j)$ is the probability of finding words i and j in the same window in a corpus (see Sect. 3.1), $p(i)$, $p(j)$ are the marginal probabilities, and $v_i, v_j \in \mathbb{R}^K$ are the corresponding word vectors. This model allows us to devise an approximation for the inner product based on the PMI. It then makes it possible, using DG methods, to construct the vectors to be assigned to words.

3.1 Co-occurrences and PMI estimator

A corpus is a set of documents. Each document is a sequence of elements called tokens whose values are words. The set of all distinct words in the corpus is called vocabulary. We consider a window of w consecutive tokens which slides over a document. By convention, windows do not overlap document boundaries. Let W be the number of windows in the corpus and n the vocabulary size. We denote by $B \in \mathbb{R}^{W \times n}$ a matrix whose columns are binary vectors $B^i \in \{0,1\}^W$, for $i = 1, \ldots, n$ with components B^i_ℓ such that $B^i_\ell = 1$ if word i appears in window ℓ, and $B^i_\ell = 0$ otherwise. We define the symmetric $n \times n$ matrix $C = B^\top B$ as the matrix of word-word co-occurrence counts. Notice that $C_{ij} = \langle B^i, B^j \rangle$ is the number of windows in which words i and j co-occur and C_{ii} is the number of windows in which word i appears. Furthermore, the vectors $(B^i)_{1 \leq i \leq n}$ are sparse and can be efficiently computed in both time and memory. Recall that $p(i,j)$ is the probability of words i and j appearing together in a window in a corpus and $p(i) = \sum_j p(i,j)$ and $p(j) = \sum_i p(i,j)$ the marginal sums. Following [Levy and Goldberg, 2014], we use an information theoretic measure, the pointwise mutual information $\mathsf{PMI}(i,j) = \log(p(i,j)/(p(i)p(j)))$ as a measure of association between words [Church and Hanks, 1990].

Approximating probabilities by relative frequencies, we have

$$\forall \{i,j\}, \qquad \frac{p(i,j)}{p(i)p(j)} \approx \frac{C_{ij}}{\sum_k C_{kj} \sum_k C_{ik}} \sum_{k,l} C_{kl} =: \rho_{ij} \tag{6}$$

where C_{ij} is an entry of the co-occurrence matrix. A natural definition for the empirical PMI matrix is the matrix M whose entries are $M_{ij} = \log \rho_{ij}$. However, notice that entries of M corresponding to zero co-occurrences $C_{ij} = 0$ are not well defined. An alternative, commonly used in NLP [Church and Hanks, 1990, Levy and Goldberg,

2014], is to consider the corrected empirical PMI matrix M^0, where

$$M^0_{ij} = \begin{cases} \log \rho_{ij}, & C_{ij} > 0 \\ 0, & C_{ij} = 0 \end{cases} \tag{7}$$

which is, moreover, a sparse matrix. A variant of Eq. (7), also considered in the literature [Levy and Goldberg, 2014], is the Positive PMI (PPMI): $M^+ = \max(M, 0)$.

As we shall see in Sect. 3.6, many methods for word embeddings employ $\mathsf{PMI}(i, j)$ as a surrogate model for the inner product $\langle v_i, v_j \rangle$, at least implicitly [Levy and Goldberg, 2014, Arora et al., 2016, Hashimoto et al., 2016]. Since there is no evident reason for $\forall i < j; \langle v_i, v_j \rangle \geq 0$, and, in view of model in Eq. (5), we choose Eq. (7) instead of the PPMI matrix.

3.2 Geometric Build-up

In Sect. 2, we saw that if all pairwise squared distances d^2_{ij} are available, then we can solve Eq. (1) by computing an eigendecomposition in $O(n^3)$ operations. It is sometimes possible to do better than $O(n^3)$. Assume the graph $\mathcal{G}$ admits a *vertex order* such that: (i) the first $m \geq K + 1$ vertices form a clique; (ii) for all other vertices $i > m$, vertex i has at least $K + 1$ *adjacent predecessors*. Then a solution to the corresponding DGP can be found in linear time [Dong and Wu, 2002]. Such vertex orders, also known as $(K + 1)$-lateration orders [Cassioli et al., 2015], can be found in polynomial time by a greedy algorithm [Lavor et al., 2012].

For $i > m$, let $\delta(i) \subset U(i) := \{j \in \mathcal{V} : \{j, i\} \in E \wedge j < i\}$ be a subset, of size $M \geq K+1$, of the set $U(i)$ of adjacent predecessors of i, with "$<$" being defined by the vertex order. Let us call vertices in $\delta(i)$ the *reference vertices*. The first m vertices can be realized using the process described in Sect. 2, leading to a cost $O(m^3)$. Assuming $|\delta(i)| = K + 1$, for every $i > m$, then, following the vertex order, the position of every other vertex $i = m + 1, \ldots, n$ can be found by solving the quadratic system:

$$\forall j \in \delta(i), \quad \|v_j - v_i\|^2 = d^2_{ji} \tag{8}$$

where $\delta(i) = \{j_1, \ldots, j_{K+1}\}$ and $v_{j_1}, \ldots, v_{j_{K+1}}$ are the position vectors of $K + 1$ adjacent predecessors of vertex i. It is not hard to show that when Eq. (8) admits a solution, it coincides with the one of a $K \times K$ linear system $Ax = b$, where A is nonsingular, provided $v_{j_1}, \ldots, v_{j_{K+1}}$ are affinely independent. See [Dong and Wu, 2002, Liberti et al., 2014] for further details.

This approach is known in the DG literature as Geometric Build-Up (GBU) [Dong and Wu, 2002, Wu and Wu, 2007]. The total cost of GBU is given by $O(m^3) + (n - m)O(K^3)$, if the involved matrices show no special structure. In the following, we discuss our DG methods for word embeddings, which are based on this idea. We assume that an $(K + 1)$-lateration vertex order is given.

3.3 GBU with inner products and fixed references

If all pairwise distances between vertices in $\delta(i) \cup \{i\} = \{j_1, \ldots, j_M, i\}$, with $M \geq K + 1$, are known, then, in view of Eq. (2), the system in Eq. (8) is equivalent to

$$\forall j \in \delta(i), \quad \langle v_j, v_i \rangle = G_{ij} \tag{9}$$

where G_{ij} denotes an entry of the Gram matrix G which, if not directly available, can be computed from the linear isomorphism between distance and Gram matrices, i.e by applying Eq. (3) to a submatrix of D containing the squared distances corresponding to the subset of vertices $\delta(i) \cup \{i\}$. Notice that Eq. (9) is also a linear system of the form $Ax = b$, with $A^\top = (v_{j_1} \ \dots \ v_{j_M}) \in \mathbb{R}^{K \times M}$ and $b = (G_{j_1,i}, \dots, G_{j_M,i})^\top$.

Let us suppose that the reference vertices for every vertex $i > m \geq K+1$ are fixed as $\delta(i) = \{1, \dots, K+1, \dots, m\}$. In this case, in the linear system of Eq. (9), although the right hand side vector $b = (G_{1,i}, \dots, G_{m,i})^\top$ changes in function of i, the coefficient matrix $A^\top = (v_1 \ \dots \ v_m)$ is the same for every $i > m$. Thus, concerning the solution of linear systems $Av_i = b_i$, for $i = m+1, \dots, n$, we can factor the matrix $A \in \mathbb{R}^{m \times K}$ *only once* and exploit its factorization to actually solve triangular systems of order K for each $i > m$. Recall that the least-squares solution of an overdetermined, full-rank system of linear equations $Ax = b$, i.e x that minimizes $\|Ax - b\|^2$, is given by the solution of the triangular system $Rx = Q^\top b$, where $A = QR$, with $R \in \mathbb{R}^{K \times K}$ and $Q \in \mathbb{R}^{m \times K}$ is the "economy size" QR decomposition of A.

This scheme leads to a cost of $O(m^3) + (n-m)O(K^2)$, where the first submatrix of G of order m is realized using spectral decomposition (see Sect. 2), followed by QR decomposition of $A = (v_1 \ \dots \ v_m)^\top$, and the positions of the remaining $n-m$ vertices are found by solving the triangular systems $Rx = Q^\top b_i$, for $i = m+1, \dots, n$.

3.4 Vertex order and loss function

Usually DGP graphs arising from many problems are quite sparse in practice. Even if they admit a $(K+1)$-lateration order, the set of reference vertices $\delta(i)$ usually changes for each $i > m \geq K+1$. However, when the underlying graph $\mathcal{G}$ is complete, *any* vertex order is in fact a $(K+1)$-lateration order. We remark that this is our case, since we know all of the entries of the co-occurrence (or empirical PMI) matrix, from which we obtain the adjacency information. Thus, we are able to apply GBU with fixed references from Sect. 3.3, where $\delta(i) = \{1, \dots, m\}$, for every $i > m \geq K+1$.

Therefore, in the GBU-based methods discussed in this paper, the proposed vertex (word) orders are simply aimed at improving the quality of the word vectors. These orders will determine which entries of the Gram matrix G ($\approx$ PMI matrix) are taken into account in the GBU method. Thus, in the objective function of Eq. 10, the vertex order determines the weight of the terms $(\langle v_i, v_j \rangle - G_{i,j})^2$ in the objective Eq. 10: 1 for edges $\{i, j\}$ used in the sequential build-up process and 0 for the others. This leads to the unconstrained optimization problem

$$\min_{V \in \mathbb{R}^{K \times n}} \quad \sum_{i=1}^{m} \sum_{j=i+1}^{m} (\langle v_i, v_j \rangle - G_{ij})^2 + \sum_{i=m+1}^{n} \sum_{j \in \delta(i)} (\langle v_i, v_j \rangle - G_{ij})^2 \tag{10}$$

where $m \geq K+1$ is the size of the initial clique. We remark that, when $|\mathcal{E}| = n(n-1)/2$ and $\delta(i) = U(i)$, problem (10) is equivalent to (4).

3.5 Divide and conquer (DC)

Given a Gram matrix $G \in \mathbb{R}^{n \times n}$, the DC method consists in the two following steps:

• Divide: consider submatrices G_i (for $i \leq P$) of G, each having size $n_i \times n_i$, such that the following conditions hold. (i) Each G_i is centered along the diagonal; (ii) G_i and G_{i+1} have at least $K+1$ points in common indexed by I_i; (iii) $\sum_{i=1}^{P-1}(n_i-|I_i|)+n_P = n$. Each submatrix defines a DGP sub-instance. Each such sub-instance is solved with a method such as matrix factorization or GBU to realize the corresponding points.

• Conquer (Merge): after the solution of each sub-instance, we have to combine the partial realizations consistently in order to obtain a realization of the whole graph. This operation is carried out sequentially as follows. The solution of the first sub-instance is saved. Then, for an instance $i+1$, for $i \geq 1$, the number I of common points between sub-instances i and $i+1$ must be at least $K+1$ in order to define unique translations and rotations for the common points to be aligned. Let X_{i+1} be the current solution obtained in the divide step, $V_i \in \mathbb{R}^{K\times n_i}$ be aligned vectors obtained at the previous step i, and let $A(:,j)$ denote the j-th column of a matrix A. Then X_{i+1} can be aligned by using Procrustes analysis [Schönemann, 1966]: the best alignment rotation $\hat{Q}_i$ and translation $\hat{T}_i$ are

$$\hat{Q}_i, \hat{T}_i = \underset{Q\in O_d, T\in\mathbb{R}^d}{\operatorname{argmin}} \sum_{k=1}^{I} \|V_i(:, n_i - I + k) - (QX_{i+1}(:,k) + T)\|_2^2. \tag{11}$$

The aligned vectors are then given by $V_{i+1} = \hat{Q}_i V_i + \hat{T}_i \mathbf{1}^\top$.

3.6 Relationship with other PMI-based methods

In this section, we perform a theoretical comparison between the DG methods proposed in this paper and other word embedding methods. To assess similarities and differences between them, we analyze their underlying optimization problems.

We recall that the empirical PMI matrix M^0, defined by Eq. (6) and Eq. (7), is used as an approximation of the Gram matrix G [Levy and Goldberg, 2014, Arora et al., 2016].

PMI-eigs. Word vectors obtained from the spectral decomposition of the empirical PMI matrix M^0 have been used in NLP literature [Levy and Goldberg, 2014]. In view of Eq. (4), these word vectors are obtained from

$$\min_{V\in\mathbb{R}^{K\times n}} \quad \|V^\top V - M^0\|_F^2 = \sum_i \sum_j (\langle v_i, v_j\rangle - M^0_{ij})^2. \tag{12}$$

The solution of Eq. (12) is constructed from the top K eigenpairs of M^0 as discussed in Sect. 2. Due to the sparsity of $M^0 \in \mathbb{R}^{n\times n}$, such eigenpairs are usually computed by an Implicitly Restarted Lanczos method (IRLM) [Sorensen, 1992] as implemented in the Matlab routine `eigs`. Its cost per iteration is given by $q\gamma n + (6K+9)qn + 4q^2 n + 2K^2 n + O((K+q)^3)$, where q is the number of shifts and γ is the average number of nonzero elements of rows of M^0, see [Lehoucq et al., 1998] for more details. If we denote $\tilde{m} = K + q$, then the above cost is $(\tilde{m} - K)\gamma n + nO(K^2) + O(\tilde{m}^3)$.

GBU. By considering the GBU method of Sect. 3.3 with m fixed references, where $\delta(i) = \{1,\ldots,m\}$, for all $i > m$, we aim at solving the optimization problem (10), with M^0_{ij} in place of G_{ij}. In this case, the objective function is similar to (12), but the

sum is not over all pairs $\{i, j\}$, but only those implied by the vertex order. For GBU, we consider a vertex order in which words are sorted in decreasing order of frequency in the corpus.

DC. The underlying objective function of DC is a variation of (12) but summing over the pairs $\{i, j\}$ corresponding to rows and columns of at least one of the submatrices $M_1^0 \approx G_1, \ldots, M_P^0 \approx G_P$ in the divide step. Concerning the vertex order, in DC we have ordered the vertices by decreasing coreness. The *coreness* of a vertex is defined as the maximum k such that it belongs to a k-core and not to a $(k+1)$-core, where a k-core is the maximum cardinality subgraph such that each vertex has an induced degree at least k. Computing k-cores can be done in linear time [Batagelj and Zaversnik, 2003].

Notice that the above methods try to fit the inner products $\langle v_i, v_j \rangle$ to the empirical PMI M_{ij}^0. In [Arora et al., 2016], the relation between $\langle v_i, v_j \rangle$ and $\mathsf{PMI}(i, j)$ is studied based on a generative model, whereas in [Hashimoto et al., 2016] the authors suggest that when the corpus size tends to infinity, for a window of size w sufficiently large, for each $\{i, j\}$, there exist a_i e b_i, such that $\|v_i - v_j\|^2 \approx -\log(C_{ij}) - a_i - b_j$. Both models claim to be consistent with matrix factorization methods [Levy and Goldberg, 2014] and others based on regression [Pennington et al., 2014] under certain assumptions.

GloVe. In [Pennington et al., 2014], the goal is to find an embedding $v : \mathcal{V} \to \mathbb{R}^K$, by solving a weighted least-squares regression problem

$$\min_{v, \hat{a}, \hat{b}} \quad \sum_i \sum_j f(C_{ij}) \left(\langle v_i, v_j \rangle + \hat{a}_i + \hat{b}_j - \log(C_{ij}) \right)^2 \tag{13}$$

where $f(C_{ij}) = \min(C_{ij}, 100)^{3/4}$. Therefore, if $\hat{a}$ and $\hat{b}$ were known, one could see (13) as a variant of (12) weighted by $f(C_{ij})$, by using $G_{ij} \approx \log(C_{ij}) - \hat{a}_i - \hat{b}_j$. Furthermore, for $\hat{a}_i = \log(C_i/\sqrt{S})$ and $\hat{b}_j = \log(C_j/\sqrt{S})$, where $C_i = \sum_j C_{ij}$ and $S = \sum_i \sum_j C_{ij}$, we have $G_{ij} \approx M_{ij}^0 = \log \rho_{ij}$, for $C_{ij} > 0$, i.e our empirical approximation for $\mathsf{PMI}(i, j)$. Whenever $C_{ij} = 0$, the corresponding term does not appear in Eq. (13), but in Eq. (10), it may contribute to the objective function if either the pair $\{j, i\}$ is in the initial clique or $j \in \delta(i)$.

fastText. In [Bojanowski et al., 2017], an extension of the skip-gram with negative sampling (SGNS) [Mikolov et al., 2013a] is proposed. This extension takes into account the morphology of words: a vector representation is associated to each character s-gram and words are represented as the sum of these vectors. It was shown in [Levy and Goldberg, 2014] that, under certain assumptions, SGNS corresponds to a matrix factorization problem whose objective is to factor a shifted PMI matrix.

Therefore, all the methods discussed in this section are somehow associated, at least implicitly, to a weighted factorization problem where the matrix to be factored is some variant of the PMI matrix.

Table 1. Computational complexity for several word vectors realization algorithms

Method	Complexity
PMI-eigs	$O((nK^2 + \tilde{m}^3)n_{\text{iter}})$
GBU (fixed references)	$O(m^3 + (n-m)K^2)$
Divide and conquer (DC)	$O(\sum_{i=1}^{P}(n_i K^2 + \tilde{m}_i^3)n_{iter}'^{(i)} + (K+1)^3(P-1))$
GloVe	$O(p^{0.8} n_{\text{epochs}} K)$

3.7 Complexity analysis

The complexity of the co-occurrence based methods discussed in the previous section are summed up in Table 1. Recall that $n = |\mathcal{V}|$ is the size of the vocabulary and K the dimension. Computation of the top K eigenpairs of the PMI matrix requires $O((nK^2 + \tilde{m}^3)n_{\text{iter}})$, where n_{iter} is the number of Implicitly Restarted Lanczos method (IRLM) iterations, and $\tilde{m} = K + q$, where q is the number of shifts, discussed in Sect. 3.6. For the DC method, $n_{\text{iter}}'^{(i)}$ represents the number of iterations to solve each sub-instance i using IRLM. We consider exactly $K+1$ anchors (also for our experiments), i.e $|I_1| = ... = |I_{P-1}| = K + 1$. Finally, it should be noted that usually $n'_{\text{iter}} \ll n_{\text{iter}}$ hence PMI-eigs complexity is not necessarily lower. The number of shifts q, q_i is internally set in the implementation of `eigs` in Matlab and it is difficult to estimate, even though it is likely that $K \leq q \ll n$ and $K \leq q_i \ll n_i$. Besides, the number of iterations $n'_{\text{iter}}, n_{\text{iter}}$ depends on the distribution of the eigenvalues of the corresponding matrices. For these reasons, we do not know how to compare their theoretical complexity. However, we will compare their empirical running times in Sect. 4.

The complexity study for Glove is detailed in [Pennington et al., 2014, Sect. 3.2] and indicates $O(p^{0.8} n_{\text{epochs}})$ where p is the total number of tokens in the corpus, and n_{epochs} the number of epochs of the stochastic gradient descent [Carpentier and Cohen, 2017, Bottou et al., 2018]. For a fair comparison with other methods, this complexity also depends linearly on the dimension. For the first three methods of Table 1, the dimension seems to be a drawback, but their complexities are good in practice. For example, for a corpus composed of $p = 2.66 \times 10^8$ tokens (corpus described in Sect. 4), $n_{\text{epochs}} = 15$ (standard corpus size and parameters), containing about $n = 10^5$ different words, dimension $K = 50$, with $m = 200$ references, we have $\mathcal{C}_{\mathsf{Glove}} = p^{0.8} n_{\text{epochs}} K \approx 4.13 \times 10^9$ and $\mathcal{C}_{\mathsf{GBU}} = m^3 + (n-m)K^2 \approx 2.50 \times 10^8$.

These complexity estimates are consistent with running times in our experiments. It should be noted that these estimates do not take pre-processing of the corpus into account, which is $O(p)$ in all cases: this corresponds to one pass through the corpus in order to construct the vocabulary, and possibly ignore low-frequency terms.

4 Experiments

To construct our word vectors, we used a corpus of 10^6 documents from Wikipedia 2016, which we cleaned using standard pre-processing methods in NLP (stop words and punctuation removal). The corpus is composed of 266561061 tokens and 81653 distinct

words. We used a window size of $w = 10$. We provide two experimental evaluations of our word vectors. First, an intrinsic evaluation of word vectors, using QVEC [Tsvetkov et al., 2015], which was shown to have good correlation with the performance of the word vectors on semantic evaluation tasks, based on alignment of features extracted from lexical resources. These evaluations are reported in Table 2. Second, we evaluate the quality of these representations over three text classification tasks compared with other word vectors. Our implementation includes 3 datasets: WebKb (Multiclass), Subjectivity (Binary) and Amazon (Binary). Results are reported in Table 3. We compare with a baseline of random word vectors whose components are drawn from a standard Gaussian. For our classification experiments, we use an implementation of a Convolutional neural network (CNN) [Kim, 2014] using Tensorflow library [Abadi et al., 2016] version 1.12[3]. We also compare with bidirectional encoders (BERT).

Table 2. Intrinsic evaluation (QVEC). First and second best are in bold and underline, respectively.

Representation	$K = 50$	$K = 100$	$K = 200$
Random	9.59	14.89	21.82
GBU ($m = M = 4K$)	22.72	30.01	37.41
DC ($n_1 = 20000, n_{i\geq 2} = 800$)	26.83	34.21	41.42
PMI-eigs	**29.20**	**36.55**	**43.74**
Glove	<u>28.22</u>	35.43	42.06
fastText	28.17	<u>36.06</u>	<u>43.51</u>

Table 3. F1 score - Text classification. First and second best are in bold and underline, respectively.

	$K = 50$			$K = 100$			$K = 200$		
Representation	Subject	WebKB	Amazon	Subject	WebKB	Amazon	Subject	WebKB	Amazon
Random	77.07	90.01	74.96	80.09	91.12	74.26	81.34	91.84	76.33
GBU ($m = M = 4K$)	87.67	92.33	79.65	<u>88.21</u>	93.04	80.58	88.05	**93.68**	81.39
DC ($n_1 = 20000, n_{i\geq 2} = 800$)	<u>87.87</u>	91.42	<u>81.97</u>	87.86	92.03	80.61	<u>88.28</u>	92.65	<u>82.49</u>
PMI-eigs	87.85	91.67	79.23	88.17	92.19	80.0	88.10	92.52	81.86
Glove	86.34	**92.99**	79.35	87.96	<u>93.07</u>	79.75	87.62	93.24	79.76
fastText	87.65	<u>92.84</u>	78.67	87.71	**93.37**	<u>80.64</u>	88.02	<u>93.57</u>	81.57
BERT + fine tuning	**91.19**	91.56	**84.28**	**91.19**	91.56	**84.28**	**91.19**	91.56	**84.28**

We provide some practical computing times, in the line of our complexity study. The word vectors and corresponding times for PMI-eigs, GBU and DC were generated using Matlab [v.2018.b], running on a CPU of 2 cores Intel(R) Core i5 1.8Ghz with 8 Gb of Ram. GloVe and fastText were compiled in the same machine using GCC Apple LLVM version 10.0.0 (clang-1000.10.44.4).

[3] We noticed a significant drop of performance when using version 2.1, the reasons remain unknown.

Table 4 reports the times for obtaining the word vectors (left) only, for dimensions $K = 50, 100, 200$, and the total time including the CNN training for $K = 200$ (right); parsing time is not included. Besides, for PMI-eigs, DC and GBU, we added the time for computing the matrix M^0 from co-occurrence counts ($\approx 120s$) and for DC and GBU we also consider the time for computing the corresponding vertex order ($\approx 90s$ and $\approx 30s$, respec.). The parameters used in GBU were $M = m = 4K$ and the ones of DC were $n_1 = 20000$ and $n_{p\geq 2} = 800$. For GloVe and fastText we kept the default parameters from the official code.

Table 4. Computing times (Left: Word Vectors, Right: Total). First and second best are in bold and underline, respectively. NA: BERT embeddings are trained only for the end-task (right table).

	Dimensions		
Representations	50	100	200
Glove	44m	1h07m	2h07m
fastText	26m	30m	48m
PMI-Eigs	<u>284s</u>	412s	836s
DC ($n_1 = 20000$, $n_{i\geq 2} = 800$)	307s	<u>348s</u>	<u>452s</u>
GBU ($m = M = 4K$)	**159s**	**168s**	**188s**
BERT	NA	NA	NA

	Datasets		
Represent. + Classif.	Subject	Amazon	WEBKB
Glove + CNN	7815s	8640s	9120s
fastText + CNN	3075s	3900s	4380s
PMI-Eigs + CNN	1031s	1856s	2336s
DC + CNN	<u>602s</u>	<u>1368s</u>	<u>1952s</u>
GBU + CNN	**383s**	**1208s**	**1688s**
BERT + fine tuning	663s	3260s	4647s

From Table 4 we observe that the training times for the DG based methods are remarkably smaller than those of standard word vectors construction (and also than the training time of bidirectional encoders such as BERT). They also improve the computational time with respect to the spectral decomposition of the PMI matrix. The price to be paid for these extremely fast word vectors, whose performance in text classification is close to the state-of-the-art (Table 3), is possibly a slightly inferior performance in intrinsic tasks (Table 2).

5 Conclusion

We proposed a formulation based on the Distance Geometry (DG) problem to generate word vectors. The resulting Geometric Build-up and Divide and Conquer algorithms are considerably faster than state-of-the-art algorithms, as GloVe and fastText.

The word vectors obtained by DG methods have performance close to state-of-the-art in our intrinsic evaluation (QVEC). Also, combined with a convolutional neural network, DG word vectors lead to F1 scores close to those of BERT in three text classification tasks, but demanding about half to a quarter of the computing time, depending on the dataset.

We believe there are two possible ways to extend this work. The first one is to enrich the information in order to construct context-aware and out of training samples representations, which would help to address other NLP problems. A potential way to address their construction would be to consider "multichannel" distance geometry, stacking different distances (or a corresponding scalar product) in different channels, and solve the different instances in order to obtain a family of realizations. An example of

channel could correspond to subword information (subword similarities), or information from other networks (e.g dictionaries, reference networks [Niwa and Nitta, 1994]).

Second, the employed DG algorithms to obtain word vectors scale well with the vocabulary size and dimension. Therefore, Distance Geometry seems a promising paradigm not only for natural language processing but also for representation learning for other tasks, such as graph classification or other applications in machine learning.

Broader Impact

1 - NLP

The superior speed of our proposed methodology goes towards the aim of rapidly generating word embeddings from given corpora relating to specific applications. Contextual word embeddings (ELMo, BERT) have significantly improved performance for many NLP tasks recently. However, these models have been minimally explored on specialty corpora, such as clinical text, as reported in [Alsentzer et al., 2019]. In such cases, there is no publicly-available pre-trained BERT models. A first possibility is to retrain contextualized word embeddings on the new corpus, which can turn out to be time consuming. In this context, representations which can be trained much faster with a negligible quality loss can turn out to be useful. In particular, we showed in this paper that DG based (non contextual) word embeddings have this property, and are also competitive on general NLP classification tasks. These preliminary results motivate us to investigate the construction of contextualized representations with DG by capitalizing on the time gain.

2 - DG and Machine Learning

In machine learning, DG represents a different paradigm for representation learning based on distances between objects. The advantages of DG over standard metric-based representation learning are

a) only a proportion of distances is required to obtain realizations, e.g standard methods such as GBU require only $O(\kappa n)$, where n is the number of objects and κ is a constant such that $\kappa \ll n$.

b) the existence of scalable algorithms in several situations, and a dedicated literature with theoretical guarantees.

References

Abadi et al., 2016. Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al. (2016). Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 265–283.

Al-Homidan and Wolkowicz, 2005. Al-Homidan, S. and Wolkowicz, H. (2005). Approximate and exact completion problems for euclidean distance matrices using semidefinite programming. *Linear Algebra and its Applications*, 406:109–141.

Alsentzer et al., 2019. Alsentzer, E., Murphy, J., Boag, W., Weng, W.-H., Jindi, D., Naumann, T., and McDermott, M. (2019). Publicly available clinical BERT embeddings. In *Proceedings of the 2nd Clinical Natural Language Processing Workshop*, pages 72–78, Minneapolis, Minnesota, USA. Association for Computational Linguistics.

Arora et al., 2016. Arora, S., Li, Y., Liang, Y., Ma, T., and Risteski, A. (2016). A latent variable model approach to PMI-based word embeddings. *Transactions of the ACL*, 4:385–399.

Arora et al., 2017. Arora, S., Liang, Y., and Ma, T. (2017). A simple but tough-to-beat baseline for sentence embeddings. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

Batagelj and Zaversnik, 2003. Batagelj, V. and Zaversnik, M. (2003). An $O(m)$ algorithm for cores decomposition of networks. *arXiv preprint cs/0310049*.

Bojanowski et al., 2017. Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the ACL*, 5:135–146.

Bottou et al., 2018. Bottou, L., Curtis, F. E., and Nocedal, J. (2018). Optimization methods for large-scale machine learning. *SIAM Review*, 60:223–311.

Carpentier and Cohen, 2017. Carpentier, P. and Cohen, G. (2017). Vue d'ensemble de la méthode du gradient stochastique. In *Décomposition-coordination en optimisation déterministe et stochastique*, pages 167–193. Springer.

Cassioli et al., 2015. Cassioli, A., Günlük, O., Lavor, C., and Liberti, L. (2015). Discretization vertex orders in distance geometry. *Discrete Applied Mathematics*, 197:27–41.

Church and Hanks, 1990. Church, K. W. and Hanks, P. (1990). Word association norms, mutual information, and lexicography. *Computational Linguistics*, 16(1):22–29.

Demmel et al., 2007. Demmel, J., Dumitriu, I., and Holtz, O. (2007). Fast linear algebra is stable. *Numerische Mathematik*, 108:59–91.

Devlin et al., 2019. Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In Burstein, J., Doran, C., and Solorio, T., editors, *Proceedings of the Conference of the North American Chapter of the ACL*, volume Human Language Technologies 1 of *NAACL*, pages 4171–4186, Minneapolis. ACL.

Dokmanic et al., 2015. Dokmanic, I., Parhizkar, R., Ranieri, J., and Vetterli, M. (2015). Euclidean distance matrices: Essential theory, algorithms, and applications. *Signal Processing Magazine, IEEE*, 32(6):12–30.

Dong and Wu, 2002. Dong, Q. and Wu, Z. (2002). A linear-time algorithm for solving the molecular distance geometry problem with exact inter-atomic distances. *Journal of Global Optimization*, 22:365–375.

Globerson et al., 2007. Globerson, A., Chechik, G., Pereira, F., and Tishby, N. (2007). Euclidean embedding of co-occurrence data. *Journal of Machine Learning Research*, 8:2265–2295.

Golub and Van Loan, 1996. Golub, G. H. and Van Loan, C. F. (1996). *Matrix Computations (3rd Ed.)*. Johns Hopkins University Press, USA.

Gower, 1982. Gower, J. C. (1982). Euclidean distance geometry. *Mathematical Scientist*, 7(1):1–14.

Hashimoto et al., 2016. Hashimoto, T. B., Alvarez-Melis, D., and Jaakkola, T. S. (2016). Word embeddings as metric recovery in semantic spaces. *Transactions of the ACL*, 4:273–286.

Hatcher and Gospodnetic, 2004. Hatcher, E. and Gospodnetic, O. (2004). *Lucene in action*. Manning Publications.

Khalife et al., 2019. Khalife, S., Liberti, L., and Vazirgiannis, M. (2019). Geometry and analogies: a study and propagation method for word representations. In *International Conference on Statistical Language and Speech Processing*, pages 100–111. Springer.

Kim, 2014. Kim, Y. (2014). Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar. ACL.

Lavor et al., 2012. Lavor, C., Lee, J., Lee-St. John, A., Liberti, L., Mucherino, A., and Sviridenko, M. (2012). Discretization orders for distance geometry problems. *Optimization Letters*, 6(4):783–796.

Lehoucq et al., 1998. Lehoucq, R., Sorensen, D., and Yang, C. (1998). *ARPACK Users' Guide*. SIAM, Philadelphia, PA.

Levy and Goldberg, 2014. Levy, O. and Goldberg, Y. (2014). Neural word embedding as implicit matrix factorization. In *Advances in Neural Information Processing Systems*, pages 2177–2185.

Liberti et al., 2014. Liberti, L., Lavor, C., Maculan, N., and Mucherino, A. (2014). Euclidean distance geometry and applications. *SIAM Review*, 56(1):3–69.

Melamud et al., 2016. Melamud, O., McClosky, D., Patwardhan, S., and Bansal, M. (2016). The role of context types and dimensionality in learning word embeddings. In *Proceedings of the Conference of the North American Chapter of the ACL*, volume Human Language Technologies of *NAACL*, pages 1030–1040, San Diego. ACL.

Mikolov et al., 2013a. Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. In Bengio, Y. and LeCun, Y., editors, *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*.

Mikolov et al., 2013b. Mikolov, T., Yih, W.-t., and Zweig, G. (2013b). Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the ACL: Human Language Technologies*, pages 746–751.

Niwa and Nitta, 1994. Niwa, Y. and Nitta, Y. (1994). Co-occurrence vectors from corpora vs. distance vectors from dictionaries. In *Proceedings of the 15th conference on Computational Linguistics - Volume 1*, pages 304–309. ACL.

Pennington et al., 2014. Pennington, J., Socher, R., and Manning, C. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. ACL.

Peters et al., 2018. Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the ACL: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. ACL.

Salton, 1971. Salton, G. (1971). *The SMART Retrieval System—Experiments in Automatic Document Processing*. Prentice-Hall, Inc., USA.

Saxe, 1979. Saxe, J. B. (1979). Embeddability of weighted graphs in k-space is strongly NP-hard. In *Proceedings of* 17^{th} *Allerton Conference in Communications, Control and Computing*, pages 480–489, Monticello, IL.

Schoenberg, 1935. Schoenberg, I. J. (1935). Remarks to Maurice Frechet's article: Sur la definition axiomatique d'une classe d'espaces vectoriels distancies applicables vectoriellement sur l'espace de Hilbert. *Annals of Mathematics*, 36:724–732.

Schönemann, 1966. Schönemann, P. H. (1966). A generalized solution of the orthogonal Procrustes problem. *Psychometrika*, 31(1):1–10.

Sorensen, 1992. Sorensen, D. C. (1992). Implicit application of polynomial filters in a k-step arnoldi method. *SIAM Journal on Matrix Analysis and Applications*, 13(1):357–385.

Tsvetkov et al., 2015. Tsvetkov, Y., Faruqui, M., Ling, W., Lample, G., and Dyer, C. (2015). Evaluation of word vector representations by subspace alignment. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2049–2054.

Turney and Pantel, 2010. Turney, P. D. and Pantel, P. (2010). From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, 37:141–188.

Vidal et al., 2016. Vidal, R., Ma, Y., and Sastry, S. (2016). *Generalized Principal Component Analysis*. Springer, New York.

Wu and Wu, 2007. Wu, D. and Wu, Z. (2007). An updated geometric build-up algorithm for solving the molecular distance geometry problem with sparse distance data. *Journal of Global Optimization*, 37:661–673.