A New Distance Geometry Method for Constructing Word and Sentence Vectors

Leo Liberti

A new distance geometry method for constructing word and sentence vectors

Leo Liberti[1]

[1] *CNRS LIX, École Polytechnique, Institut Polytechnique de Paris, 91128 Palaiseau, France*
Email:liberti@lix.polytechnique.fr

June 24, 2020

Abstract

We present a new methodology for producing low-dimensional word vectors based on distance geometry and dimensional reduction techniques. We use these word vectors in order to construct sentence vectors. We evaluate their usefulness in a sentence classification task performed by a simple artificial neural network. Compared to n-gram incidence vectors, the new methodology is shown to yield lower loss values.

1 Introduction

Many Natural Language Processing (NLP) tasks concern clustering and classification applied to excerpts of text (words, sentences, documents) in a corpus [34, 18]. The default computer representation of words, sentences and documents is a string of ASCII characters. A common approach for clustering a set of strings is to define a similarity graph, having strings as vertices and edges representing string similarity, and then use a graph clustering methodology [40, 2]. On the other hand, some of the unsupervised learning methodologies for clustering sets of vectors (e.g. k-means [33, 6]) are known to be fast and effective. This provides a first motivation for a need to transform words, sentences, documents into vectors.

A second motivation is given by supervised learning methodologies, most of which are based on vector input: e.g. support vector machines, logistic regression, artificial neural networks (ANN). A third motivation is given by semantic compositionality. Formal languages are semantically compositional: the meaning of a formal language sentence is completely determined by the meaning of each of its parts [25]. Natural languages have varying degrees of semantic compositionality, but it is clear to every human that the meaning of a natural language sentence is rarely completely determined by the meaning of its parts [24]. While we are still unable to devise a computer program to understand natural language as a human would, the transformation of linguistic units into vectors in a Euclidean space yields an elusive, yet tantalizing promise of multifaceted semantic compositionality possibilities: there are many ways to combine sets of vectors to yield other vectors or sets thereof.

Euclidean vectors representing linguistic units (words, sentences, documents) are required to reflect the relations between the units. In other words, related units should yield related vectors. The nature of the relation varies between methodologies; its effectiveness is usually verified empirically on some common NLP task (e.g. the word vectors proposed in [42] are tested on analogies, similarity and named entity recognition tasks).

It stands to reason, then, that the first step in turning strings to vectors is to form a graph, which is simply a representation of a relation on a set. Word vectors are derived from graphs having words as vertices, and some relation on the words encoded by the edges (e.g. the word vectors in [36] encode some syntactic and semantic relationships). In this paper we consider the graph-of-words representation of [43], which is based on word contiguity in a given text (usually a paragraph or a short text). Other methods

exist: the WordNet graph captures semantic hierarchies [38], and the popular bag-of-words binary vectors are derived from a co-occurrence relation of words in a document of any length [35].

The process of mapping the vertices of an edge-weighted graph to vectors in a Euclidean space $\mathbb{R}^K$ (for some integer K) is called *graph realization* [28]. The name *graph embedding* is more general, and refers to an embedding of a graph in a topological space [14]. If the graph vertices are words, realizing the graphs produces word vectors. Currently, many word vector construction methods (with corresponding sets) are available: e.g. Glove [42], Word2Vec [37], fastText [7]. All of these word vector sets have proved their worth on a number of NLP tasks [20]. Usually, their dimension K ranges in the hundreds, e.g. $K = 300$ for a public Word2Vec word vector set available from Google and constructed from an analysis of text from GoogleNews, see `en.wikipedia.org/wiki/Word2vec`.

By a high-dimensional phenomenon called *distance resolution limit* (also known as "concentration of distances"), as the dimension of a set of vectors sampled from any probability distribution increases, the difficulty of distinguishing between smallest and largest distance of the vectors to a given point also increases [4, 1]. More precisely, as the dimension tends towards infinity, the largest distance is bounded above by a $(1+\varepsilon)$ factor of the smallest distance, for any $\varepsilon > 0$, with probability 1. This phenomenon may have a negative impact on clustering algorithms such as k-means, where one wants to distinguish between smallest and second-smallest (let alone largest) distance to a centroid, as well as on other clustering tasks on vectors. This motivates the need for low-dimensional word vectors.

In this paper we exploit Distance Geometry (DG) and dimensional reduction methodologies for producing low-dimensional word vectors ($K = 10$) out of a given corpus. From these word vectors, by stacking, padding and more dimensional reduction, we derive sentence vectors of dimension around 400. We then test these in a sentence classification task carried out by means of an ANN, and compare the results with sentence vectors obtained by the incidence of consecutive word triplets into the set of all such triplets in the text, showing that the sentence vectors derived with DG perform better (see the algorithm description in Sect. 4) The original contribution of this paper is the application of known methodologies for graph realization and dimensional reduction to NLP, and more specifically to graph embeddings for neural networks. Some technical details are also original, e.g. Sect. 3.4.

The rest of this paper is organized as follows. In Sect. 2 we briefly recall two dimensional reduction methodologies: Principal Component Analysis (PCA) and Random Projections (RP). In Sect. 3 we give a summary of the DG field, and outline a Mathematical Programming (MP) based methodology for graph realization. In Sect. 4 we describe our sentence classification task, the construction of word and sentence vectors, and the ANN used for testing. In Sect. 5 we discuss computational results.

2 Dimensional reduction

Dimensional reduction is a field of data science aiming at lowering the dimension of a possibly large set of high dimensional vectors while keeping some property of the vectors approximately invariant.

2.1 Principal Component Analysis

In the well-known PCA method the approximate invariant is the variance of the data set in an appropriate orthogonal frame of reference [15]. PCA requires a target dimension as input, and produces lower-dimensional versions of the input vectors such that as much as possible of the variability of the original data set is preserved in the lower-dimensional vectors.

2.2 Random projections

Let $X = \{x_i \mid i \leq n\} \in \mathbb{R}^m$ be the initial, high-dimensional set of vectors. We consider X also as a matrix of column vectors in $\mathbb{R}^{m\times n}$. Given an $\varepsilon > 0$ and $k = O(\frac{1}{\varepsilon^2}\ln n)$, a *random projection* is a (random) matrix $T \in \mathbb{R}^{k\times m}$ s.t. $\forall i < j \leq n$,

$$(1-\varepsilon)\|X_i - X_j\|_2 \leq \|TX_i - TX_j\|_2 \leq (1+\varepsilon)\|X_i - X_j\|_2. \tag{1}$$

The Johnson-Lindenstrauss lemma states that RPs exist [17]. If T is a matrix where each component is sampled from a Gaussian distribution with zero mean and $\frac{1}{\sqrt{k}}$ standard deviation, it can be shown that resampling T sufficiently many times can make Eq. (1) hold with arbitrarily high probability [12]. Therefore, RPs are a dimensional reduction technique aiming to keep congruence approximately invariant. In other words, the vector sets X and TX have "almost the same shape" (although X has exponentially many more dimensions than TX). It is interesting to note that the lower dimension k does not depend on the original dimension m, but only on ε (assumed fixed) and the number n of vectors.

2.3 Concentration of measure

A result such as that of Eq. (1) is surprising at first sight. It does not hold in lower dimensional spaces, so it is impossible to visualize. It follows from a high-dimensional geometric phenomenon called "concentration of measure". Quoting [3], "The value of a well behaved function at a random point of a big probability space is very close to the mean value of the function. [...] In a sense, measure concentration can be considered as an extension of the law of large numbers." This was exploited for a number of algorithmic and MP purposes [16, 8, 47, 46].

3 Distance geometry

DG is geometry based on the concept of distance rather than points and lines. For example, a subset Y of a metric space (S, d) is metrically convex if, for all $y, z \in Y$ there is an x between y, z; and for any triplet $x, y, z \in S$ we say that x is between y and z if the $d(x, y) + d(y, z) = d(x, z)$. Thus one can define convexity by distance alone. DG has had a profound impact on mathematics, from Heron's theorem to Cayley-Menger determinants, to a lesser-known theorem of Gödel about DG on a sphere [26]. In this paper we focus on methods for solving the main problem in DG, see below.

3.1 The main problem in DG

The main problem in DG today is dictated by applications: in many cases it is possible to measure some of the distances between a set of entities having unknown position. The Distance Geometry Problem (DGP) consists in finding positions for the entities so that they are compatible with the given distances [27]. The positions are given by vectors in some Euclidean space of given dimension K. Typical applications are to clock synchronization ($K = 1$), sensor network localization ($K = 2$), protein conformation from Nuclear Magnetic Resonance experiments ($K = 3$), rigidity of 3D structures ($K = 3$) and more [28].

A partial set of distances is naturally represented by an edge-weighted graph $G = (V, E, d)$: an edge $\{u, v\}$ with weight d_{uv} is in E if the distance d_{uv} between u and v is known. The DGP is the problem of realizing a graph as a sequence of vectors $x = (x_v \in \mathbb{R}^K \mid v \in V)$ defining the positions of the vertices in $\mathbb{R}^K$. Formally, the DGP consists in solving the following system of quadratic equations:

$$\forall \{u, v\} \in E \quad \|x_u - x_v\|_2^2 = d_{uv}^2. \tag{2}$$

We remark that the realization x is represented as a tall, slim matrix in $\mathbb{R}^{n\times K}$, the v-th row of which is the vector corresponding to vertex $v \in V$.

3.2 MP based solution methods for the DGP

Many different solution methods exist for the DGP, some of which are general, while some are specific to some given graph classes [30]. Most of the general methods are based on MP. In this paper we focus on the algorithm described in [32], which provides an efficient heuristic method for graph realization in a given dimension K.

1. Formulate and solve a Semidefinite Programming (SDP) relaxation [22] of Eq. (2):

$$\left.\begin{array}{lrcl} \min \sum\limits_{\{u,v\}\in E} (X_{uu} + X_{vv} - 2X_{uv}) & + & \alpha\,\mathsf{tr}(X) \\ \forall \{u,v\} \in E \quad X_{uu} + X_{vv} - 2X_{uv} & = & d_{uv}^2 \\ X & \succeq & 0, \end{array}\right\} \tag{3}$$

   where α is a constant, to get an $n \times n$ matrix solution $\bar{X}$, with $n = |V|$.

2. From $\bar{X}$ obtain a graph realization of G in $\mathbb{R}^K$, represented by the matrix $x' \in \mathbb{R}^{n\times K}$, using Barvinok's naive algorithm [3] applied to the DGP [32]:
   (a) factor $\bar{X}$ into $YY^\top$, using e.g. spectral decomposition;
   (b) sample each component of a matrix $y \in \mathbb{R}^{n\times K}$ from a Gaussian distribution with zero mean and standard deviation $\frac{1}{\sqrt{K}}$;
   (c) let $x' = Y\,y$ (note $x' \in \mathbb{R}^{n\times K}$).

3. Formulate and locally solve, using a local Nonlinear Programming (*NLP*) solver [48], the following quartic unconstrained formulation [23] of Eq. (2):

$$\min \sum_{\{u,v\}\in E} (\|x_u - x_v\|_2^2 - d_{uv}^2)^2, \tag{4}$$

   using x' as a starting point; call $x^* \in \mathbb{R}^{n\times K}$ the result.

4. Output x^* as the realization of G.

It was shown in [32] that the above algorithm is efficient and finds good solutions for protein graphs. We shall see later (Sect. 4.2) that protein graphs are not so different from graph-of-words.

3.3 Multistart

We remark that Eq. (4) by itself can be used in a MultiStart (MS) heuristic for the DGP over general graphs: namely, Eq. (4) is solved locally, using a local *NLP* solver, starting from randomly sampled starting points, a fixed number of times: the algorithm then outputs the best solution found over its run.

3.4 Barvinok's naive algorithm: DGP error analysis

Since $\bar{X}$ is the Gramian of Y by construction, Y is an n-dimensional solution of Eq. (2), i.e. a high-dimensional realization of G. We note that the random $n \times K$ matrix y acts as a RP on Y. If K is $O(\ln n)$, by Eq. (1) this would show that x' is approximately congruent with Y; moreover, since x^* locally

minimizes the error of Eq. (4) w.r.t x', x^* approximately solves Eq. (2), so we can expect a low feasibility error.

If, on the other hand, the dimension K of the word vectors is given, the above reasoning fails to apply. A different probabilistic guarantee on x' is given in [3] for $K = 1$, and extended in [32] to arbitrary K. For each $\{u, v\} \in E$ let $A_{uv} = \{x \in \mathbb{R}^{n \times K} \mid \|x_u - x_v\|_2^2 = d_{uv}^2\}$. Then

$$\forall \{u, v\} \in E \quad \mathsf{dist}(x', A_{uv}) \leq c\sqrt{\|\bar{X}\|_2 \ln n}, \tag{5}$$

with arbitrarily high probability, where c is a constant which depends (linearly) only on $\log_n m$. The probability that Eq. (5) holds can be made arbitrarily high by choosing a specific tolerance in the proof of [32, Thm. 5].

Although the above argument applies to fixed K, and states that x' is unlikely to be far from satisfying Eq. (2), the error bound on the right hand side of Eq. (5) grows with n and $\|\bar{X}\|_2$. While $\sqrt{\ln n}$ has a slow growth, $\sqrt{\|\bar{X}\|_2}$ might grow faster. We have:

$$\sqrt{\|\bar{X}\|_2} = \sqrt{\|Y^\top Y\|_2} \leq \sqrt{\|Y^\top\|_2 \|Y\|_2} = \sqrt{\|Y\|_2^2} = \|Y\|_2.$$

As we remarked earlier, the $n \times n$ matrix Y represents a realization in $\mathbb{R}^n$ of the graph G with n vertices: specifically, its v-th row is the position vector of vertex $v \in V$. We note that Y is obtained by spectral decomposition of $\bar{X}$: namely $Y = \sqrt{\Lambda}P$, where P is a matrix of orthonormal row eigenvectors and $\Lambda = \mathsf{diag}(\lambda)$ is a diagonal matrix of eigenvalues λ of $\bar{X}$ (we assume that $\lambda = (\lambda_1, \ldots, \lambda_n)$ and $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$). This implies that $\|Y\|_2 = \sqrt{\lambda_1}$, which measures the spread of the realization vectors along the component of maximum variance.

A realization in $\mathbb{R}^n$ is a set of n vectors in $\mathbb{R}^n$. The case maximizing the variance in a single direction vector is when the realization is wholly contained in a one-dimensional subspace, i.e. Y has rank 1. By invariance w.r.t. congruences, we assume that $\lambda_1 > 0$ and $\lambda_j = 0$ for all $j \neq 1$ (meaning the one-dimensional subspace is spanned by the standard basis vector e_1) and that the realization is centered at zero. Let $\bar{d} = \max_{\{u,v\} \in E} d_{uv}$. The variance is maximum when the points are arranged in two separate clusters: half the points are at around $-\frac{1}{2}\bar{d}$ and the other half are at around $\frac{1}{2}\bar{d}$, which implies that all of the intra-cluster distances are bounded above by some small $\delta > 0$ and all the inter-cluster ones are bounded below by $\bar{d} - \delta n$. Hence $Y_{i1}^2 \leq \frac{1}{4}\bar{d}^2$ for all $i \leq n$, and

$$\mathsf{Var}(Y_{i1}) = \mathbb{E}((Y_{i1} - \mathbb{E}(Y_{i1}))^2) = \mathbb{E}(Y_{i1}^2) = \frac{1}{n}\sum_{i \leq n} Y_{i1}^2 \leq \frac{1}{4}\bar{d}^2,$$

whence $\|Y\|_2 = \sqrt{\lambda_1} \leq \frac{1}{2}\bar{d}$. For tighter bounds, one must take the graph structure into account. We give two examples: (i) if the distances are all the same, then $\bar{d} = \frac{1}{m}\sum_{\{u,v\} \in E} d_{uv}$; (ii) if X satisfies further constraints, such as $\sum_{i \leq n} X_{ii} \leq 1$, then $\|X\|_2 = \mathsf{tr}(X)/n \leq 1/n$, which shows that $\|Y\|_2 \to 0$ as $n \to \infty$ [3].

3.5 Concentration of distances

In this section we explain in more detail the motivation given in the introduction in favor of devising methods for low-dimensional word embeddings.

Let $m \in \mathbb{N}$. For each $i \leq m$ consider an infinite sequence of random variables $\mathcal{X}_i^n$ (for $n \in \mathbb{N}$), each of which is associated to a multivariate distribution of n elements: samples from $\mathcal{X}_i^n$ yield vectors $x_i \in \mathbb{R}^n$. We assume that, for each fixed n, the different $\mathcal{X}_i^n$ are identically distributed. Consider also another sequence of random variables $\mathcal{Z}^n$ (for $n \in \mathbb{N}$), such that its samples are vectors $z \in \mathbb{R}^n$.

We now sample vectors x_i from $\mathcal{X}_i^n$ for $i \leq m$, and a vector z from $\mathcal{Z}^n$. We let $X = \{x_i \mid i \leq m\} \subset \mathbb{R}^n$. We are interested in finding the smallest distance $d_{\min}^n$ and the largest distance $d_{\max}^n$ between z and X,

where $d^n : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}_+$ is a family of distance functions. We then want to show that $d^n_{\max} \leq (1+\varepsilon) d^n_{\min}$ with probability 1 for any $\varepsilon > 0$.

To this purpose we define dependent random variable sequences $\mathcal{D}^n_{\min} = \min_i d^n(Z^n, X^n_i)$ and $\mathcal{D}^n_{\max} = \max_i d^n(Z^n, X^n_i)$. Assuming that for some $i \leq m$ the following holds:

$$\lim_{n\to\infty} \mathsf{Var}\left(\frac{d^n(Z^n, X^n_i)}{\mathbb{E}(d^n(Z^n, X^n_i))}\right) = 0, \tag{6}$$

then it is shown in [4] that, for any $\varepsilon > 0$, we have

$$\lim_{n\to\infty} \mathsf{Prob}(D^n_{\max} \leq (1+\varepsilon) D^n_{\min}) = 1. \tag{7}$$

The proof of this striking result is wholly based on basic properties of distributions. It uses the concept of convergence in probability, Slutsky's theorem [50], and some elementary steps in probability theory. From Eq. (6), it derives the fact that $\frac{\mathcal{D}^n_{\max}}{\mathcal{D}^n_{\min}}$ converges to 1 in probability as n increases, whence the result.

The assumption, Eq. (6), informally states that the growth of the distances between $\mathcal{Z}^n$ and the $\mathcal{X}^n_i$ with n is somehow "regular": the variance of the distances, scaled by its average, tends to zero as the dimension grows. While this is different from concentration of measure (Sect. 2.3), it is at least reminiscent of the phenomenon. Thus the name "concentration of distances".

4 A sentence classification task

In this section we present and discuss our experiment, designed to ascertain which graph embedding method yields the best loss performance of an ANN for clustering sentences in a text. Our pipeline works as follows:

1. it reads a text file, "cleans" it, and splits it into sentences (Sect. 4.1);
2. it constructs a graph-of-words (Sect. 4.2) from each sentence;
3. it uses DG methods to map graphs-of-words to realizations (Sect. 3), which are word vectors;
4. it stacks these realizations onto single vectors (i.e. it concatenates the rows of a realization) and zero-pads the shorter vectors to fit all vectors to a same size, yielding sentence vectors;
5. it shortens the sentence vectors using dimensional reduction (Sect. 2);
6. it clusters the sentence vectors using k-means [33] in order to produce a clustering for each considered DG and dimensional reduction method;
7. it constructs training sets (Sect. 4.4) corresponding to each clustering obtained at the previous step;
8. it trains and tests an ANN (Sect. 4.3) with these training sets, and reports results (Sect. 5).

We note that the clusterings found by k-means at Step 6 are used as "ground truths" that the ANN is supposed to learn. The point here is that an ANN should be able to learn *whatever* ground truth is encoded in its training set, so it does not matter which ground truth is used. We want to remark most emphatically that the point is *not* that of using an ANN instead of k-means.

4.1 The text

We considered the essay *On the duty of civil disobedience* by H. Thoreau [44], initially stored in an text file named `walden.txt` downloaded from `archive.org`. This file has 116608 words organized in 10108 lines. The file size is 661146 bytes. We removed common words, stopwords, punctuation and unusual characters, which reduced the text to 4083 sentences over a set of 11431 words. We note that, since some sentences contained less than three words after text cleaning, only 3940 sentences remained in the sentence set S.

4.2 Graph-of-words

Given a sentence s represented as a sequence of words $s = (s_1, \ldots, s_m)$, an n-gram is a subsequence of n consecutive words of s. Thus, each sentence has at most $(m - n + 1)$ n-grams. In a graph-of-words $G = (V, E)$ of order n, V is the set of words in s, i.e. each unique word in s is associated to a single vertex, even if the same word appears more than once in s. Two words have an edge only if they appear in the same n-gram. The weight of the edge is equal to the number of n-grams in which the two words appear [43]. This graph may also be enriched with semantic relations between the words, obtained e.g. from WordNet.

There is an interesting connection between graphs-of-words and protein graphs: namely that both have an order over the vertex set (the sentence s for graphs-of-words and the backbone for protein graphs), and that both have a minimum subset of edges defined by contiguity (word proximity in the sentence for graphs-of-words, and atomic distances known by bond lengths and angles for protein graphs) [19]. The most prominent difference between these two graph structures is given by the fact that, when the same word appears more than once in a sentence, it is assigned a unique vertex. On the other hand, if the same atom type appears more than once in a protein graph, every occurrence is assigned a separate vertex.

The connection between graphs-of-words and protein graphs is interesting because the structure of protein graphs has been deeply studied in the literature, see e.g. [31, 29, 9]. Adapting these studies to graphs-of-words appears to be a promising line of research.

4.3 The artificial neural network

Like many other approximation models, ANNs are parametric models designed to represent a function $f : \mathbb{R}^n \to \mathbb{R}^k$ with an unknown description, but which can be called as a black-box oracle. In this section we shall give a definition according to the point of view of MP. We note that, given a directed graph (digraph) $G = (V, A)$, we define

$$\forall j \in V \quad N^-(j) = \{i \in V \mid (i, j) \in A\}.$$

An ANN can be represented by a digraph $G = (V, A)$ with a tripartition defined on V, namely $V = I \dot\cup H \dot\cup O$, where $\dot\cup$ denotes the disjoint union, such that $|I| = n$ and $|O| = k$. The n nodes in I are called "input nodes", and the k nodes in O are called "output nodes". The rest of the nodes, in H, are the "hidden nodes".

All other decisions about the topology of G are debatable, but common sense and experience [13] recommend that:

- I has no incoming arc and O has no outgoing arc;
- G is acyclic, i.e. it is a Directed Acyclic Ggraph (DAG);
- H is multipartite, so $H = H_1 \dot\cup \cdots \dot\cup H_h$ for some $h \in \mathbb{N}$;

- each node set in the partition $\{I, H_1, \ldots, H_h, O\}$ induces an empty DAG, i.e. all arcs connect a node in a set to a node in a different set of the partition;
- each node $j \in V$ has an associated value u_j;
- each non-input node $j \in V \setminus I$ has an associated "activation function" $\phi_j : \mathbb{R} \to \mathbb{R}$ (usually, it is assumed that all of the ϕ_j are the same function, say ϕ, aside perhaps for the output nodes in O) and a threshold b_j;
- each arc $\{i, j\} \in A$ has a weight w_{ij};
- the relationship between node values and arc weights is as follows:

$$\forall j \in V \setminus I \quad u_j = \phi_j \left(\sum_{i \in N^-(j)} w_{ij} u_i + b_j \right).$$

Usually, the common activation function ϕ is decided *a priori*, mostly based on experience w.r.t. the task at hand. The ANN parameters, $w \in \mathbb{R}^{|A|}$ and $b \in \mathbb{R}^{|V|-|I|}$, are decided by means of a training set, which we formalize as a pair (X, Y) of sets of vectors such that $|X| = |Y| = N$; $X \subset \mathbb{R}^n$ is known as the input set, and $Y \subset \mathbb{R}^k$ as the output set. Because input/output sets have the same cardinality N, we can form a set T of input/output pairs (x, y) so that x is the j-th vector of X and y is the j-th vector of Y. The optimal solution (w^*, b^*) to the following training problem

$$\left.\begin{array}{rrcl} \min\limits_{w,b,v} & \sum\limits_{t \le N} \|(v_{tj} | j \in O) - y_t\|_2 \\ \forall t \le N, j \in I & v_{tj} & = & x_{tj} \\ \forall t \le N, j \in V \setminus I & v_{tj} & = & \phi\big(\sum\limits_{i \in N^-(j)} w_{ij} v_{ti} + b_j \big) \end{array}\right\} \tag{8}$$

gives the values of the ANN parameters w, b. We note there there are many alternative objective functions in the literature to that of Eq. (8), all having the general form $\min \beta(v, y)$ with β being some sort of similarity between v and y. In general, the purpose of the objective is to decrease the difference between the values at the output nodes and the t-th output vector from a training set.

Consider an ANN based on a DAG with a multipartite architecture. Once trained, it models the black-box function $\xi = f(\chi)$, with input argument $\chi \in \mathbb{R}^n$. The function output $\xi \in \mathbb{R}^k$ can be evaluated as follows:

$$\left.\begin{array}{rrcl} \forall j \in I & u_j & = & \chi_j \\ \forall j \in V \setminus I & u_j & = & \phi_j\big(\sum\limits_{i \in N^-(j)} w_{ij} u_i + b_j \big) \\ \forall j \in O & \xi_j & = & u_j. \end{array}\right\} \tag{9}$$

Since each set of the node partition induces an empty subgraph, it suffices to evaluate u over each $H_1, \ldots, H_h, O$ in layers.

In our experiments, the ANN topology was based on a single hidden node set H with $|H| = 20$, and an output set O with $|O| = 1$. We used a rectified linear unit (ReLu) activation function [49] for all $j \in H$ and a sigmoid function $\phi_j(z) = \frac{1}{1+e^{-z}}$ for the single output node. Both activation functions map to $[0, 1]$. The output part of our training set encodes a cluster label, which must be mapped to $[0, 1]$. We achieved this by using consecutive integers as labels (say up to $r \in \mathbb{N}$), encoding the integers $\{1, \ldots, r\}$ in $[0, 1]$ by dividing the interval into r sub-intervals of equal length, and mapping $i \le r$ to the i-th sub-interval of $[0, 1]$ (this is just one of several methods for handling a discrete output using an ANN).

Finally, we measured the ANN performance using the mean squared error loss function value

$$L = \frac{1}{N} \sum_{t \le N} \|y_t - \xi_t\|_2^2,$$

where ξ_t is the output of the ANN when given as input x_t, and (x_t, y_t) are the input/output pairs ranging over the part of the training set reserved for testing.

4.4 The training sets

We aim at comparing the ANN performance over different training sets (X, Y). The construction of the input part X depends on the DG method μ employed to transform graphs-of-words representations of sentences to realizations, and on the dimensional reduction method ρ used to shorten the sentence vectors derived from stacking and padding the realization. For each $X(\mu, \rho)$ we derived a ground truth $Y(\mu, \rho)$ by using the k-means algorithm in order to cluster $X(\mu, \rho)$ in a number $q(\mu, \rho)$ of clusters. We recall that output parts of training sets are 1-dimensional: we encode cluster labels by assigning a corresponding sub-interval of the $[0, 1]$ scalar interval, as explained above.

4.4.1 "Bag of n-grams" vectors

As a comparison benchmark, we used a variant of the popular "bag of words" approach: each sentence s is mapped to a corresponding binary vector x_s encoding the incidence of the n-grams of s in the set $\mathscr{G}$ of all of the n-grams in the text. In other words, the i-th component of x_s is 1 if the i-th n-gram occurs in sentence s, and 0 otherwise. The vector x_s is then shortened using dimensional reduction methods. Setting $n = 3$ yields $|\mathscr{G}| = 48087$, so each sentence vector in our testbed is in $\{0, 1\}^{48087}$.

4.4.2 Graph embedding methods

We let μ range over the set $M = \{\mathsf{inc}, \mathsf{qrt}, \mathsf{sdp}\}$, the set of methods used to derive training sets:

- inc are the "bag of 3-grams" incidence vectors;
- qrt is the quartic unconstrained DG formulation in Eq. (4) solved by a MS heuristic (Sect. 3.3) on a single iteration around the IpOpt [11] local *NLP* solver; each graph-of-words is mapped to a realization with $K = 10$, which is then stacked and padded to a sentence vector;
- sdp is the method presented in Sect. 3.2: the SDP formulation in Eq. (3) is solved by the Mosek [39] SDP solver, followed by Barvinok's naive algorithm for realization rank reduction to $K = 10$ dimensions, followed by a local *NLP* solution of Eq. (4); each graph-of-words is mapped to a realization with $K = 10$, which is then stacked and padded to a sentence vector.

4.4.3 Dimensional reduction methods

We let ρ range over the set $R = \{\mathsf{pca}, \mathsf{rp}\}$, the set of methods used to reduce dimension of the vectors:

- pca denotes the PCA method: the target dimension was the smallest such that the residual variance in the neglected components was almost zero; for cases where the reduced dimension is set to 400, the residual variance was nonzero;
- rp denotes the RP methodology: it yields vector sets having the same dimension because it only depends on the number of vectors (Sect. 2.2) in the set, which is always equal to the number of sentences, i.e. 3940, as mentioned in Sect. 4.1; the ε in the Johnson-Lindenstrauss lemma was set to 0.2.

We report the dimensionality of input sets and number of clusters encoded in output sets in Table 1.

Dimensionality of input vectors			
μ	$\lvert\sigma\rvert = 3940$		
ρ	inc	qrt	sdp
pca	3	400	400
rp	373	373	373
original	*48087*	*1460*	*1460*
Number of clusters to learn			
pca	3	9	14
rp	3	16	14

Table 1: Training set statistics for $X(\mu, \rho)$ and number of clusters in the corresponding output sets.

4.4.4 Construction of the training sets

We remark that, although each ground truth (output set) $Y(\mu, \rho)$ is derived from the corresponding input set $X(\mu, \rho)$, we want to test the ability of the ANN to learn a variety of ground truths. Therefore, it makes sense to form combinations of training sets consisting of different input and output sets. Accordingly, multiple input/output pair sets $T(\mu_1, \mu_2, \rho_1, \rho_2)$ are formed by input $X(\mu_1, \rho_1)$ and output $Y(\mu_2, \rho_2)$ sets.

5 Computational results

All tests were carried out on a 4-CPU Intel Xeon 8-core CPU with 64GB ram running Linux CentOS. The code was mostly written in Python 3 [45].

5.1 Implementation details

We parsed and cleaned the input text using some basic NLTK [5] functions with Python 3. For the construction of graphs-of-words we re-used a code provided by a colleague, S. Khalife. We used the k-means implementation in `scikit-learn` [41]. Graph realizations were obtained by solving the optimization problems in Eq. (3) and (4) with off-the-shelf solvers Mosek [39] and IpOpt [11]. We implemented the ANN using the `keras` python library [10]. The default configuration was chosen for all layers. We used the Adam solver [21] in order to solve the training problem Eq. (8). For each training set we tested, 35% was used for training, 35% for validation, and 30% for testing.

5.2 Comparison

The computational results are reported in Table 2. Each cell reports the mean squared error loss value of the ANN trained with input set $X(\mu_1, \rho_1)$ (first column) in order to predict the output set $Y(\mu_2, \rho_2)$ (first row). The cells in italics are those where $\mu_1 = \mu_2$. The last colum sums the losses. It is clear that that sentence vectors obtained with RP-based dimensional reduction yield better training sets than those with PCA-based dimensional reduction. Moreover, the best methods are DG-based rather than incidence vectors: both methods scored very close loss values, with qrt slightly outperforming sdp.

Training set inputs	Training set outputs							
	μ / ρ	inc pca	inc rp	qrt pca	qrt rp	sdp pca	sdp rp	sum $\mu \in M$
	inc pca	*0.052*	*0.013*	0.106	0.164	0.079	0.161	0.575
	inc rp	*0.001*	*0.000*	0.106	0.167	0.080	0.159	0.513
	qrt pca	0.063	0.022	*0.038*	*0.218*	0.079	0.159	0.579
	qrt rp	0.062	0.024	*0.120*	*0.035*	0.076	0.164	**0.481**
	sdp pca	0.063	0.021	0.126	0.195	*0.033*	*0.149*	0.587
	sdp rp	0.059	0.021	0.121	0.176	*0.083*	*0.037*	0.497

Table 2: Comparison test results.

6 Conclusion

In this paper we presented a methodology for deriving low-dimensional word vector, and correspondingly low-dimensional sentence vectors. The methodology is based on many existing techniques: distance geometry, mathematical programming, concentration of measure, random projections. These sentence vectors were tested as training sets of a simple artificial neural network, against a standard benchmark consisting of incidence vectors of 3-grams, and found to give better results at predicting various ground truths about the vector sets.

References

[1] C. Aggarwal, A. Hinneburg, and D. Keim. On the surprising behavior of distance metrics in high dimensional space. In J. Van den Bussche and V. Vianu, editors, *Proceedings of ICDT*, volume 1973 of *LNCS*, pages 420–434, Berlin, 2001. Springer.

[2] D. Bader, P. Sanders, and D. Wagner, editors. *Graph Partitioning and Graph Clustering*, volume 588 of *Contemporary Mathematics*. AMS, Providence, RI, 2013.

[3] A. Barvinok. Measure concentration in optimization. *Mathematical Programming*, 79:33–53, 1997.

[4] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. When is "nearest neighbor" meaningful? In C. Beeri and P. Buneman, editors, *Proceedings of ICDT*, volume 1540 of *LNCS*, pages 217–235, Heidelberg, 1998. Springer.

[5] S. Bird, E. Klein, and E. Loper. *Natural Language Processing with Python*. O'Reilly, Cambridge, 2009.

[6] J. Blömer, C. Lammersen, M. Schmidt, and C. Sohler. Theoretical analysis of the k-means algorithm: A survey. In L. Kliemann and P. Sanders, editors, *Algorithm Engineering*, volume 9220 of *LNCS*, pages 81–116, Cham, 2016. Springer.

[7] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*, 2016.

[8] C. Boutsidis, A. Zouzias, and P. Drineas. Random projections for k-means clustering. In *Advances in Neural Information Processing Systems*, NIPS, pages 298–306, La Jolla, 2010. NIPS Foundation.

[9] A. Cassioli, O. Günlük, C. Lavor, and L. Liberti. Discretization vertex orders for distance geometry. *Discrete Applied Mathematics*, 197:27–41, 2015.

[10] F. Chollet and *et al.* Keras. `https://keras.io`, 2015.

[11] COIN-OR. *Introduction to IPOPT: A tutorial for downloading, installing, and using IPOPT*, 2006.

[12] S. Dasgupta and A. Gupta. An elementary proof of a theorem by Johnson and Lindenstrauss. *Random Structures and Algorithms*, 22:60–65, 2002.

[13] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, Cambridge, MA, 2016.

[14] J. Gross and T. Tucker. *Topological graph theory*. Wiley, New York, 1987.

[15] H. Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24(6):417–441, 1933.

[16] P. Indyk and A. Naor. Nearest neighbor preserving embeddings. *ACM Transactions on Algorithms*, 3(3):Art. 31, 2007.

[17] W. Johnson and J. Lindenstrauss. Extensions of Lipschitz mappings into a Hilbert space. In G. Hedlund, editor, *Conference in Modern Analysis and Probability*, volume 26 of *Contemporary Mathematics*, pages 189–206, Providence, RI, 1984. AMS.

[18] D. Jurafsky and J. Martin. *Speech and Language Processing*. Stanford University, Stanford, Draft 191016.

[19] S. Khalife. Sequence graphs: characterization and counting of admissible elements. In C. Gentile et al., editor, *Proceedings of Cologne-Twente Workshop 2020*, AIRO Springer Series. Springer, New York, accepted.

[20] S. Khalife, L. Liberti, and M. Vazirgiannis. Geometry and analogies: a study and propagation method for word representation. In *Statistical Language and Speech Processing*, volume 7 of *SLSP*, 2019.

[21] D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Proceedings of ICLR*, San Diego, 2015.

[22] E. De Klerk. *Aspects of Semidefinite Programming*. Number 65 in Applied Optimization. Kluwer, Dordrecht, 2004.

[23] C. Lavor, L. Liberti, and N. Maculan. Computational experience with the molecular distance geometry problem. In J. Pintér, editor, *Global Optimization: Scientific and Engineering Case Studies*, pages 213–225. Springer, Berlin, 2006.

[24] G. Leech. *Semantics*. Pelican Books. Penguin, Harmondsworth, 1974.

[25] R. Levine, T. Mason, and D. Brown. *Lex and Yacc*. O'Reilly, Cambridge, second edition, 1995.

[26] L. Liberti and C. Lavor. Six mathematical gems in the history of distance geometry. *International Transactions in Operational Research*, 23:897–920, 2016.

[27] L. Liberti and C. Lavor. *Euclidean Distance Geometry: An Introduction*. Springer, New York, 2017.

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

[29] L. Liberti, C. Lavor, and A. Mucherino. The discretizable molecular distance geometry problem seems easier on proteins. In A. Mucherino, C. Lavor, L. Liberti, and N. Maculan, editors, *Distance Geometry: Theory, Methods, and Applications*, pages 47–60. Springer, New York, 2013.

[30] L. Liberti, C. Lavor, A. Mucherino, and N. Maculan. Molecular distance geometry methods: from continuous to discrete. *International Transactions in Operational Research*, 18:33–51, 2010.

[31] L. Liberti, B. Masson, C. Lavor, J. Lee, and A. Mucherino. On the number of realizations of certain Henneberg graphs arising in protein conformation. *Discrete Applied Mathematics*, 165:213–232, 2014.

[32] L. Liberti and K. Vu. Barvinok's naive algorithm in distance geometry. *Operations Research Letters*, 46:476–481, 2018.

[33] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proc. 5th Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. University of California Press, 1967.

[34] C. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA, 1999.

[35] M. McTear. *The Conversational Interface*. Springer, Cham, 2016.

[36] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26 of *NIPS*, pages 3111–3119, La Jolla, 2013. NIPS Foundation.

[37] T. Mikolov, W.-T. Yih, and G. Zweig. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751, 2013.

[38] G. Miller. Wordnet: A lexical database for English. *Communications of the ACM*, 38(11):39–41, 1995.

[39] Mosek ApS. *The `mosek` manual, Version 8*, 2016.

[40] M. Newman. Modularity and community structure in complex networks. *Proceedings of the National Academy of Sciences*, 103(23):8577–8582, 2006.

[41] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[42] J. Pennington, R. Socher, and C. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[43] F. Rousseau and M. Vazirgiannis. Graph-of-word and TW-IDF: new approach to ad hoc IR. In *Proceedings of CIKM*, New York, 2013. ACM.

[44] H. Thoreau. Resistance to civil government. In E. Peabody, editor, *Æsthetic papers*. J. Wilson, Boston, MA, 1849.

[45] G. van Rossum and *et al. Python Language Reference, version 3*. Python Software Foundation, 2019.

[46] K. Vu, P.-L. Poirion, and L. Liberti. Random projections for linear programming. *Mathematics of Operations Research*, 43(4):1051–1071, 2018.

[47] K. Vu, P.-L. Poirion, and L. Liberti. Gaussian random projections for Euclidean membership problems. *Discrete Applied Mathematics*, 253:93–102, 2019.

[48] A. Wächter and L. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, 2006.

[49] Wikipedia. Rectifier (neurl networks), 2019. [Online; accessed 190807].

[50] Wikipedia. Slutsky's theorem, 2019. [Online; accessed 190802].