Sequence graphs realizations and ambiguity in language models

Sammy Khalife, Yann Ponty, Laurent Bulteau

▶ **To cite this version:**

Sammy Khalife, Yann Ponty, Laurent Bulteau. Sequence graphs realizations and ambiguity in language models. COCOON 2021 - 27th International Computing and Combinatorics Conference, Oct 2021, Tainan, Taiwan. hal-02495333v3

HAL Id: hal-02495333

https://hal.science/hal-02495333v3

Submitted on 13 Jan 2021 (v3), last revised 18 May 2023 (v4)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Sequence graphs realizations and ambiguity in language models

Sammy Khalife
LIX, CNRS, Ecole Polytechnique, Institut Polytechnique de Paris, 91128 Palaiseau, France
khalife@lix.polytechnique.fr

Yann Ponty
LIX, CNRS, Ecole Polytechnique, Institut Polytechnique de Paris, 91128 Palaiseau, France
yann.ponty@lix.polytechnique.fr

Laurent Bulteau
LIGM, CNRS, Université Gustave Eiffel, 77454 Marne-la-Vallée, France
laurent.bulteau@univ-eiffel.fr

Abstract

Several language models rely on an assumption modeling each local context as a (potentially oriented) bag of words, and have proven to be very efficient baselines. Sequence graphs are the natural structures encoding their information. However, a sequence graph may have several realizations as a sequence, leading to a degree of ambiguity. In this paper, we study such degree of ambiguity from a combinatorial and computational point of view. In particular, we present theoretical results concerning the family of sequence graphs. Several combinatorial problems are presented, depending on three levels of generalisation (window size, graph orientation, and weights), and whether some of these are NP-complete is left opened. We establish different algorithms, including an integer program and a dynamic programming formulation to respectively recognize a sequence graph and to count the number of its distinct realizations. This allows us to show that this model assumption can induce an important number of sentences to have the same representations. We empirically compare the representations obtained with a recurrent neural networks for different realizations of sequence graphs.

2012 ACM Subject Classification Mathematics of computing → Combinatoric problems; Mathematics of computing → Combinatorics on words; Mathematics of computing → Graph algorithms; Theory of computation → Complexity classes; Theory of computation → Problems, reductions and completeness

Keywords and phrases Graphs, Sequences, Combinatorics, Inverse problem, Complexity class

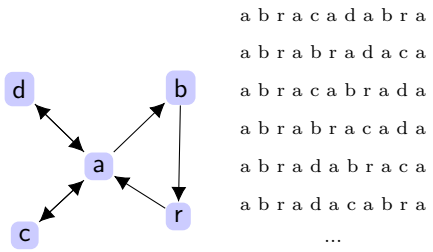
1 Introduction

The automated treatment of familiar objects, either natural or artifacts, always relies on a translation into entities manageable by computer programs. However, the correspondence between the object to be treated and "its" representation is not necessarily one-to-one. The representations used for learning algorithms are no exception to this rule. In particular, natural language words and textual documents representations are essential for several tasks, including document classification [23], role labelling [19], and named entity recognition [16]. The traditional models based on pointwise mutual information, or graph-of-words (GOW), [9, 17, 20], supplement the content of bag-of-words (TF, TFIDF) with statistics of co-occurrences within a **window** of fixed size w, introduced to mitigate the degree of ambiguity. Several models [2, 14, 18, 21] also use the same type of information and constitute strong baselines for natural language processing.

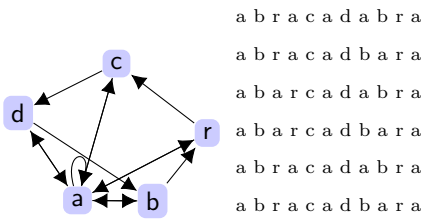
While these representations are more precise than the traditional bag-of-words (e.g Parikh vectors), they still induce some level of ambiguity, *i.e.* a given graph can represent several sequences. Our study is thus motivated by a quantification of the level of ambiguity, seen

(a) No ambiguity ($w = 3$) **(b)** Ambiguity ($w = 2$)

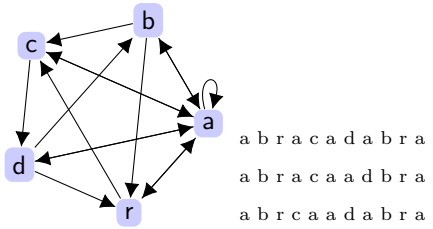
Figure 1 Sequence graphs (or *graphs-of-words*) built for the sentence "Linux is not UNIX but Linux" using window sizes 3 (a) and 2 respectively (b). In the second case, the sequence graph is ambiguous, since any circular permutation of the words admits the same representation.



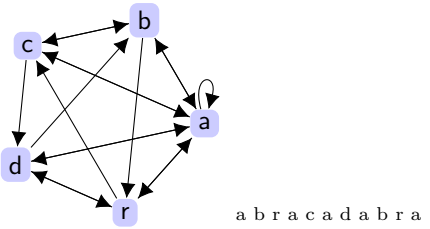
(a) $w = 2$, G has 30 realizations



(b) $w = 3$, G has 6 realizations



(c) $w = 4$, G has 3 realizations



(d) $w = 5$, G has one realization

Figure 2 Sequence graphs (or *graphs-of-words*) built for the sentence "a b r a c a d a b r a" using window sizes 2 (a), 3 (b), 4 (c) and 5 (d).

as an algorithmic problem, coupled with an empirical assessment of the consequences of ambiguity for the representations.

After introducing in Section 2 the formal definition of a sequence graph and the descriptions of our main problems, we establish in Section 3.1 complexity aspects of deciding the existence and counting sequences in GOWs associated with a window size $w = 2$. Then we consider in Section 3.2 the general case $w \geq 3$, and propose a integer program and a dynamic programming algorithm to respectively recognize a sequence graph and count admissible sequences. Finally, we assess the prevalence of ambiguity within a synthetic dataset, and observe that sequences invariant with respect to the GOW representation do not lead to invariance with respect to recurrent neural networks such as Long Short Term Memory networks (LSTMs).

Related work

Sequence graphs encode the information of several co-occurences based models [2, 15, 18]. To the best of our knowledge, the ambiguity and realizability questions addressed in this work were never systematically addressed by prior work in computational linguistics. Furthermore, we believe the problems studied in this paper are interesting from an algorithmic point of view, and appear to be devoid of reduction to other well-known problems.

However, some similarities exist between our problem and others studied in the Distance Geometry (DG) literature. In distance geometry, the input consists of a set of pairwise distances between points, having unknown positions in a d-dimensional space. The problem then consists in determining (the existence of) a set of positions for the points, satisfying the distance constraints. Since a position is fully characterized from $d+1$ constraining neighbors, the problem can be solved by finding a sequential order for processing points, such that the assignment of a point is always by at least $d+1$ among its neighbors [13]. This statement shares some level of similarity with our problem since an admissible sequence for a window $w = d+2$ also represents a linear ordering of its nodes, in which $w-1 = d+1$ of the neighbors have lower value with respect to the order.

The reasons for the insufficiency of linear ordering in DG to solve our realizability problem are threefold. First, each element of the sequence x associated to the protein backbone is associated a unique vertex. This is not the case we investigate here, since a symbol can be repeated several times, but only one vertex is created in the graph. This implies that the vertex associated to the i^{th} element ($i \geq w$) of x can have strictly less than $w-1$ distinct neighbors in its predecessors in x. Second, the absence of loops in distance geometry, because an element is at distance 0 from itself. Finally, the graphs are always undirected in distance geometry.

2 Definitions and problem statement

Let $x = x_1, x_2, ..., x_p$ be a finite sequence of discrete elements among a finite vocabulary X. Without loss of generality, we can suppose that $X = \{1, ..., n\}$. In the following, let $I_p = \{1, ..., p\}$. This motivates the following definition:

▶ **Definition 1.** *$G = (V, E)$ is the graph of the sequence x with window size $w \in \mathbb{N}^*$ if and only if $V = \{x_i \mid i \in I_p\}$, and*

$$(i,j) \in E \iff \exists (k,k') \in I_p^2,\ |k-k'| \leq w-1\ x_k = i \text{ and } x_{k'} = j \tag{1}$$

For digraphs, Eq. (1) is replaced with

$$(i,j) \in E \iff \exists (k,k') \in I_p^2,\ k \leq k' \leq k+w-1, x_k = i \text{ and } x_{k'} = j. \tag{2}$$

Finally, a weighted sequence graph G is endowed with a matrix $\Pi(G) = (\pi_{ij})$ such that

$$\pi_{ij} = \mathsf{Card}\ \{(k,k') \in I_p^2 \mid k \leq k' \leq k+w-1,\ x_k = i \text{ and } x_{k'} = j\} \tag{3}$$

We say that x is a w-admissible sequence for G (or a realization of G), if G is the graph of sequence x with window size w.

The natural integers π_{ij} represent the number of co-occurrences of i and j in a window of size w. Hence, the graph of sequence is unique. An linear time algorithm to construct a weighted sequence digraph is presented in Sec. A of the appendix. Other cases are obtained similarly. The procedure in algorithm 1 defines a correspondence between the sequence set $X^\star$ into the graph set $\mathcal{G}$: $\phi_w \colon X^\star \to \mathcal{G}, x \mapsto G_w(x)$. Based on these definitions, we consider the following problems:

▶ **Problem 1** (Weighted-Realizable (W-Realizable)).
Input: *Possibly directed graph G, matrix weights Π, window size w*
Output: *True if (G, Π) is the w-sequence graph of some sequence x, False otherwise.*

▶ **Problem 2** (Unweighted-Realizable (U-Realizable)).
Input: *Possibly directed graph G, window size w*
Output: *True if G is the w-sequence graph of some sequence x, False otherwise.*

We denote D-Realizable (resp. G-) the restricted version of Realizable where the input graph G is directed (resp. undirected), and W-Realizable (resp. U-) the restricted version of Realizable where the input graph G is weighted (resp. unweighted), possibly in combination with the D- or G- variants. We write Realizable$_w$ for the case where w is a fixed (given) constant. We also consider the variants of W-Realizable, denoted WG-Realizable and WD-Realizable where the input graph is restricted to be respectively undirected and directed. We define UG-Realizable and UD-Realizable similarly. Finally, we write (WG-, WD-, ...)Realizable$_w$ for the case where w is a fixed strictly positive integer.

▶ **Problem 3** (Unweighted-NumRealizations (U-NumRealizations)).
Input: *Possibly directed graph G, window size w*
Output: *The number of **realizations** of G,* i.e. *preimages of G through ϕ_w such that $|\{x \in X^\star \mid \phi_w(x) = G\}|$ if finite, or $+\infty$ otherwise.*

▶ **Problem 4** (Weighted-NumRealizations (W-NumRealizations)).
Input: *Possibly directed graph G, matrix weights Π, window size w*
Output: *The number of **realizations** of G in the weighted sense.*

Similarly, we use the same prefix for the directed or undirected versions of (D-, G-, i.e. DU- for directed and unweighted). We also denote NumRealizations$_w$ for the case where w is a fixed strictly positive integer. Note that NumRealizations strictly generalizes the previous one, as Realizable can be solved by testing the nullity of the number of suitable realization computed by NumRealizations.

DW Directed weighted	**DU** Directed unweighted
GW Undirected weighted	**GU** Undirected unweighted

3 Theoretical results

In this section, we present our main theoretical results. Due to length limitations, some of the proofs are left in the appendix.

3.1 A complete characterization of 2-sequence graphs

A graph has a sequential realization with $w = 2$ when there exists a path visiting every vertex and covering all of its edges (at least once for the unweighted case and exactly π_e for the edge e in the weighted case). This characterization enables relatively simple characterization and algorithmic treatment, leading to the results summarized in Table 2.

Table 1 Complexity for various instances of our problems ($w = 2$)

	NumRealizations$_2$		Realizable$_2$	
Data Instance	Complexity	#Sequences	Complexity	Characterization
Unweighted graph	P	$\{0, +\infty\}$	P	G connected
Weighted graph	#P-hard	$\{0, 1\} \cup 2\mathbb{N}^*$	P	$\psi(G)$ (semi) Eulerian
Unweighted digraph	P	$\{0, 1, +\infty\}$	P	Theorem 14
Weighted digraph	P	$\mathbb{N}$ (BEST Theorem)	P	$\psi(G)$ (semi) Eulerian

3.2 General sequence graphs and REALIZABLE$_{w\geq 3}$

The characterization of more general sequence graphs, such as 3-graphs is not the same for 2-graphs, as shows the counter-example in Fig 3a: the depicted graph has no self-edge so there must at least one clique of size 3. Similarly, Fig. 3b depicts a counter example for directed graphs: G does not have loops, so if it had a 3-admissible sequence, such sequence must be of the form $\{1\,2\,3\,1..., 1\,3\,2\,1..., 2\,3\,1\,2..., 3\,2\,1\,3..., 2\,1\,3\,2...\}$ but then $(2,1)$ would form an edge.

(a) G is connected but not a 3-sequence graph

(b) G is strongly connected but not a 3-sequence graph

Figure 3 Counter examples for $w = 3$

3.3 A polynomial time algorithm for GU-REALIZABLE$_w$

Similarly to the procedure in Sec. B, we will use an auxiliary graph built on G. Let $H(G) = (E, H_E)$ be the new graph obtained with the following procedure. Two edges $e = (v_1, v_2)$, $f = (v_3, v_4)$ of E are connected in $H(G)$ if and only if:

$$v_2 = v_3 \;\text{ and }\; (v_1, v_4) \in E \tag{4}$$

This defines an injective function $\tilde{h} : E_H \to V^3$: an edge of $H(G)$ can be seen as an unique triplet v_1, v_2, v_3 where $(v_1, v_2), (v_1, v_3)$ and $(v_2, v_3) \in E$. Therefore, by definition, a walk P in $H(G)$ is always of the form:

$$P = (t_1, t_2), ..., (t_{p-1}, t_p) \;\text{ s.t }\; \forall i \in \{1, ..., p-1\},\; (t_i, t_{i+1}) \in E \tag{5}$$

It is clear that if $H(G)$ is a 2-graph, then G is a 3-graph since there is a walk going through all edges of $H(G)$ (so visiting every non isolated node and creating all edges of G). However, the converse is not true as depicted in Fig. 4. In order to determine if $G = (V, E)$ has an admissible sequence in the general case, a procedure is to recursively merge pairs of vertices, maintaining constraints depending on E. These constraints are similar to Eq. 4. We adopt the following notations, $u_{i,j} = (u_i, u_j)$ and $u_{1:k} = (u_1, ..., u_k)$. The iterative procedure for $w \geq 3$ is summed up in the following equation. Namely, $\forall k \in \{2, ..., w-2\}$, one has

$$E^{(k)} = \{u_{1:k+1} \in V^{k+1} \mid u_{1:k} \in E^{(k-1)}, u_{2:k+1} \in E^{(k-1)} \wedge (u_1, u_{k+1}) \in E\} \tag{6}$$

Let $H^{(k)} = (E^{(k)}, E^{(k+1)})$, it can be defined recursively through:

$$H^{(0)} = G \qquad\qquad \forall k \in \mathbb{N}^*,\;\; H^{(k)} = f(H^{(k-1)}) \tag{7}$$

where f transforms edges into vertices and creates edges between new vertices that verify Eq. 6.

▶ **Definition 2.** *Let u be a vertex of $H^{(k)}$ for $k \in \mathbb{N}$, $u = (u_1, ..., u_k, u_{k+1})$. The sequence $u_1, ..., u_{k+1}$ is the **authentic** sequence of u. We also call an authentic sequence of a walk on $H^{(k)}$: $P = (x_1, ..., x_{k+1}), (x_2, ..., x_{k+2}), ..., (x_v, ..., x_{v+k})$ the sequence $x_1, x_2, ..., x_{v+k}$.*

In order to obtain admissible sequences of length p, the computation of $H^{(p)}$ requires p iterations, and the number of vertices and edges of $H^{(k)}$ can increase during iterations (the complete graph is an example for which theses numbers increase exponentially).

▶ **Proposition 3.** *Let $x = x_1, ..., x_p$ be a w-admissible sequence of a graph (or digraph) $G = (V, E)$. If $w \leq p$, x, then x is an authentic sequence of a walk of length $p - w + 1$ on $H^{(w-2)}$.*

Proof. Due to length limitation, we provide a proof sketch, full proof is left in the appendix. The following property by induction on k:

$$\forall k \in \{w, ..., p\},\ \exists \text{ walk } P \text{ on } H^{(w-2)} \text{ such that :}$$

$$x_{1:k} = P[1]_1, P[2]_1, ..., P[k-w]_1, P[k-(w-1)]_{1:(w-1)}$$

- Initialisation: $k = 1$. By construction of $H^{(w-2)}$, x_1 is the first element of the "static walk": $x_{1:w-1} \in H^{(w-2)}$.
- Induction: Verification that if $x_{1:k}$ is a walk of length $k - w + 1$, one can find a walk of length $(k + 1) - w + 1$ to generate $x_{1:(k+1)}$. ◀

▶ **Theorem 4.** *Let $w \in \mathbb{N}^*$.* GU-Realizable$_w$ *is in P.*

Proof. The case for $w = 1$ is trivial, and $w = 2$ has been treated. For $w \geq 3$, an algorithm is obtained by going through all the connected components of $H^{(w-2)}$. Let $C_1, ..., C_m$ the connected components of $H^{(w-2)}$. On the one hand, it is possible to compute them in polynomial time. On the other hand, it is possible to construct walks covering all of their respective edges in polynomial time (for instance iteratively using shortest paths). Let $W_1, ..., W_m$ such walks and $X_1, ..., X_m$ their respective admissible sequences.

Using Prop. 3, G is a w-sequence graph if and only if there exists a walk $\tilde{W}_{i_0}$ on some C_{i_0} creating exactly the edges of G. However, W_{i_0} creates more edges than any walk on C_{i_0} by construction.

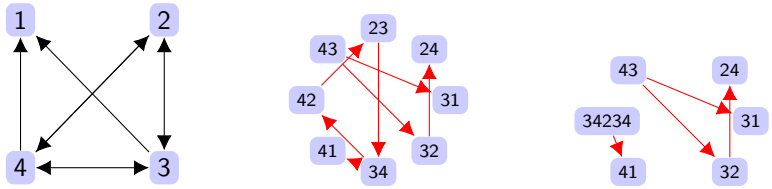
In conclusion, the assertion:

$$\exists i \in \{1, ..., m\},\ \phi_w(X_i) = G$$

is a characterization that G is a w-sequence. This assertion is decidable in polynomial time since for all i, $\phi_w(X_i)$ is computable in polynomial time (cf. Algorithm 1). ◀

For digraphs, the analogue of the aforementioned procedure would consist in enumerating alll paths in the DAG $R(H^{(w-2)})$. However, the number of paths can be exponential, even for a sequence graph. In the next subsection, we will prove that DU-Realizable$_w$ is actually NP-hard. Finally, if $x_1, ..., x_c$ are vertices of a strongly component of $H^{(w-2)}$, which order should be considered to form a new vertex attribute x_C? The following lemma shows that this order is not important, as long as it represents a walk in the component. Moreover, it is possible to reconstruct all admissible sequences from walks on $R(H^{w-2})$. With the same notations:

▶ **Lemma 5.** *Let x a walk on $H^{(w-2)}$ whose authentic sequence is w-admissible for G. If x goes through a strongly component C of $H^{(w-2)}$, adding any supplementary path included in C is stable for w-admissibility. Any graph generated by a walk on $H^{(w-2)}$ can be generated by a walk on $R(H^{(w-2)})$.*



(a) G **(b)** H **(c)** $R(H)$

Figure 4 Procedure to find a 3-admissible sequence. 34234, 41: is 3-admissible, with authentic sequence 3 4 2 3 4 1

Proof. We present a proof sketch. The first statement concerning stability requires a straightforward verification using the definition of $H^{(w-2)}$. Second, a procedure to generate G from a walk on $R(H^{(w-2)})$ using a walk $x_{1:p}$ on $H^{(w-2)})$ is to consider an iterative scheme, and discuss three cases:

- (i) x_i and x_{i+1} are not in a strongly connected component (SCC)
- (ii) x_i is not in a SCC and x_{i+1} is in a SCC
- (iii) x_i and x_{i+1} are both in SCCs

For case (i), we just keep x_i and x_{i+1}. For cases (ii) and (iii), we use the first part result of the Lemma and add covering walks over the strongly connected components. ◀

3.4 Main complexity results

In this subsection we present the remaining complexity results, which are summarized in Table 2. In the previous subsection, we proved that GU-Realizable$_w \in P$, $\forall w \geq 3$. Besides, for GU, the number of realizations of a graph G is either 0 (not realizable), $+\infty$ (realizable and there exists a cycle in a component of H generating G), or 1 (realizable but no cycle in any component of H generating G). These three cases can be tested in polynomial time using our algorithm, showing that GU-NumRealizations$_w \in P$, $\forall w \geq 3$. In the remaining of this section, we present the reductions we used for the other instances.

Table 2 Complexity for various instances of our problems ($w \geq 3$). We remind that a para-NP-hard problem does not admit any XP algorithm unless P=NP.

	Constant w, $w \geq 3$		Parameter w	
	NumRealizations$_w$	Realizable$_w$	NumRealizations	Realizable
Variation	Complexity	Complexity	Complexity	Complexity
GU	P	P	W[1]-hard; XP	W[1]-hard; XP
GW	NP-hard	NP-hard	para-NP-hard	para-NP-hard
DU	NP-hard	NP-hard	para-NP-hard	para-NP-hard
DW	NP-hard	NP-hard	para-NP-hard	para-NP-hard

▶ **Proposition 6.** Clique *admits a polynomial time parameterized reduction into* GU-Realizable.

Proof. Let $G = (V, E)$ be a simple graph. Let G' be a graph constructed from G adding two nodes a and b with loops, such that a and b are connected to each vertex of G. Let k be a strictly positive integer and $w = k + 1$. We will show that G has a k-clique if and only if G' is w-realizable.

First, let us suppose that G has a k-clique. Let C be an arbitrary sequence of the vertices of one of its k-clique. Let $v_1, \ldots, v_{|V|}$ be the vertices of G and $(u_1, u'_1), \ldots, (u_{|E|}, u'_{|E|})$ be its edges. In the following x^w represents the w-repetition of x. Then, the following sequence is a w-realization of G':

$$a^w\ u_1\ u'_1\ a^w\ u_2\ u'_2\ a^w\ \ldots\ a^w\ u_{|E|}\ u'_{|E|}\ a^w \quad C \quad b^w\ v_1\ b^w\ v_2\ b^k\ \ldots b^w v_{|V|}$$

Now let us suppose that G' is w-realizable and let $x = x_1, \ldots, x_p$ be a w-realization of G'. Without loss of generality, let us suppose a appears before b in x. Let i_b be the index of the first appearance of b and let i_a be the largest index of the appearance of a before i_b. Then $i_b - i_a \geq w$, otherwise there would be an edge between a and b. Furthermore, since G is simple, there cannot be two repetitions of a vertex in the sequence $x_{i_a+1}, \ldots, x_{i_a+w-1}$. Due to the definition of a sequence graph, all vertices $\{x_{i_a+1}, \ldots, x_{i_a+w-1}\}$ are connected, forming a clique in G of size $w - 1 = k$, which ends the proof. ◀

▶ **Corollary 7.** GU-Realizable *is* W[1]*-hard for parameter* w*.*

DU-Realizable is NP-hard for $w \geq 3$

Consider the following intermediate problem:

OptionalRealizable$_w$ Given a directed unweighted graph $D = (V, A)$, a subset $A' \subseteq A$ of *compulsory arcs*, two distinguished vertices $s, s' \in V$. Is there a sequence S such that the graph of S contains only arcs in A and (at least) all arcs in A'.

We first prove that this problem is NP-hard, then show how it reduces to DU-Realizable.

OptionalRealizable$_w$, $w \geq 3$ is NP-hard

Given $G = (V, A)$ and a start vertex s, build a directed weighted graph $G' = (V', A')$ as follows:

- Vertex set: $V = \bigcup_{v \in V} \{v_0 \mid v_1\} \cup \{x^i_p, 1 \leq p \leq 2n+1, 1 \leq i \leq w-2\}$
- Arc set,
  - optional arcs (x^i_{2p-1}, v_0), (v_0, x^i_{2p}), (x^i_{2p}, v_1), (v_1, x^i_{2p+1}) for each $v \in V$, $1 \leq p \leq n$, $1 \leq i \leq w-2$.
  - optional arcs (u_1, v_0) for each (u, v) in A
  - compulsory arcs (v_0, v_1) for each $v \in V$
  - optional arcs (x^i_p, x^j_p) for $i < j$ and (x^i_p, x^j_{p+1}) for $j \leq i$

Start vertices are $(x^1_0, \ldots, x^{k-2}_0, s)$.

G' is a yes-instance $\Leftrightarrow$ G admits a hamiltonian path

$\Leftarrow$ Let v^p be the pth vertex of V in the hamiltonian path. Let X^p be the sequence $x^0_{2p-1} \ldots x^{w-2}_{2p-1} v^p_0 x^0_{2p} \ldots x^{w-2}_{2p} v^p_1$. Let $X^{n+1} = x^0_{2n} \ldots x^{w-2}_{2n}$, and S be the concatenation $X^1 \ldots X^{n+1}$. It can be checked that S contains only arcs of A and all compulsory arcs.

$\Rightarrow$ Consider a sequence S, an occurrence of x^i_p in S for some $1 \leq i \leq w-2$, $1 \leq p \leq n$ (note that $p \neq n+1$), and let S' be the subsequence of S containing the $w-1$ characters following x^i_{2p+1}. Let $T = x^{i+1}_p \ldots x^{w-2}_p$ and $T = x^1_{p+1} \ldots x^i_{p+1}$ (note that T is possibly empty). T and U are seen both as strings and as sets of vertices. The out-neighborhood of x^i_p contains all vertices of $T \cup U$, as well as all vertices v_q for $v \in V$, where $q = 0$ if p is odd and $q = 1$ if p is even. Since there are $k-2$ vertices in $T \cup U$, and no vertex has a self-loop, then by the pigeon-hole principle string S' must contain at least one vertex v^q, $v \in V$. Since there are no arc (v^q, v'^q) for $v, v' \in V$, S' contains exactly one vertex v^q, thus

it also contains all vertices of $T \cup U$. Based on the direction of the arcs in $T \cup U \cup \{v^q\}$, it follows that $S' = T \cdot v^q \cdot U$.

Let X_p be the string $x_p^1 \dots x_p^{w-2}$. From the arguments above, and the fact that S starts with X_1, there exist indices $i_1, j_1, \dots, i_n, j_n$ such that

$$S = X_1 v_{i_1}^0 X_2 v_{j_1}^1 X_3 v_{i_2}^0 X_4 v_{j_3}^1 X_5 \dots X_{2n+1}$$

From the window size w, there must exist an arc $(v_{i_p}^0, v_{j_p}^1)$ for each p, so by construction $i_p = j_p$. Furthermore, these arcs are compulsory for each vertex v^0, so $(i_1, \dots, i_n)$ is a permutation of $\{1, \dots, n\}$. Finally, there also exist an arc $(v_{j_p}^1, v_{i_{p+1}}^0)$ in G', so there exists an arc $(v_{i_p}, v_{i_{p+1}})$ in G. Thus, $(v_{i_1}, \dots, v_{i_n})$ is a hamiltonian path in G.

DU-Realizable$_w$ is NP-hard

By reduction from OptionalRealizable$_w$. Given a directed unweighted graph $G = (V, A)$, a subset $A' \subseteq A$ of compulsory arcs (let $A'' = A \setminus A'$ be the set of optional arcs), an integer w, and $w-1$ distinguished vertices $s_1 \dots s_{w-1} \in V$.

Let $m = |A''|$, write $A'' = \{(u_1, v_1), \dots, (u_m, v_m)\}$. Create G' by adding $w(m+1)$ *separator* vertices y_p^i, $1 \leq p \leq m+1$ and $1 \leq i \leq w$ and m vertices z_p. Build the strings

$$Z = \left(\prod_{p=1}^{m} (y_p^1 \dots y_p^w u_p z_p v_p) \right) y_{m+1}^1 \dots y_{m+1}^w$$

$$Z' = Z s_1 \dots s_{w-1}$$

. Add all arcs realized by Z' involving y_p^i and/or z_p to G'.

G has a realization with optional arcs $\Leftrightarrow$ G' has a realization

$\Rightarrow$ Build a realization for G' by concatenating Z with the realization for G starting with $s_1 \dots s_{w-1}$. All optional arcs of G' are realized in Z, all compulsory arcs of G' are realized in the suffix (the realization of G'), and all arcs involving a separator are realized in Z'. No forbidden arc is realized.

$\Leftarrow$ Let S be a realization of G'. The set of in-neighbors of any separator has size at most $w-1$ and induce a tournament in G' (this is clear for all arcs involving separators, it is also true for a potential pair of vertices (u_i, v_i) of G since G has no length-2 cycle. So the $w-1$ characters before any separator are ordered as in Z. Furthermore each separator (except y_1^1) contains at least one other separator in each in-neighborhood, so any occurrence of a separator is actually the last character of a substring of S equal to a prefix of Z. Since y_1^1 has in-degree 0, it may only appear as the first character of S, and any prefix of Z in S is also a prefix of S. Moreover since y_{m+1}^w must appear in S, we have $S = ZS'$ with no separator appearing in S'. Thus S' realizes only arcs from G. From the out-neighborhood of $y_{m+1}^1, \dots, y_{m+1}^w$, we have that S starts with $s_1, \dots s_{w-1}$. Moreover no compulsory arc of G is realized in Z, nor with one vertex in Z and one in S' (since such arcs start with a separator), so all compulsory arcs are realized in S'. Overall, G is a yes-instance of OptionalRealizable$_w$ with sequence S'.

GW-Realizable$_w$, DW-Realizable$_w$ are NP-hard for all $w \geq 3$

By reduction from a variant of hamiltonian path:
Input: Undirected graph G with two degree-1 vertices.
Question: Does G have a hamiltonian path?

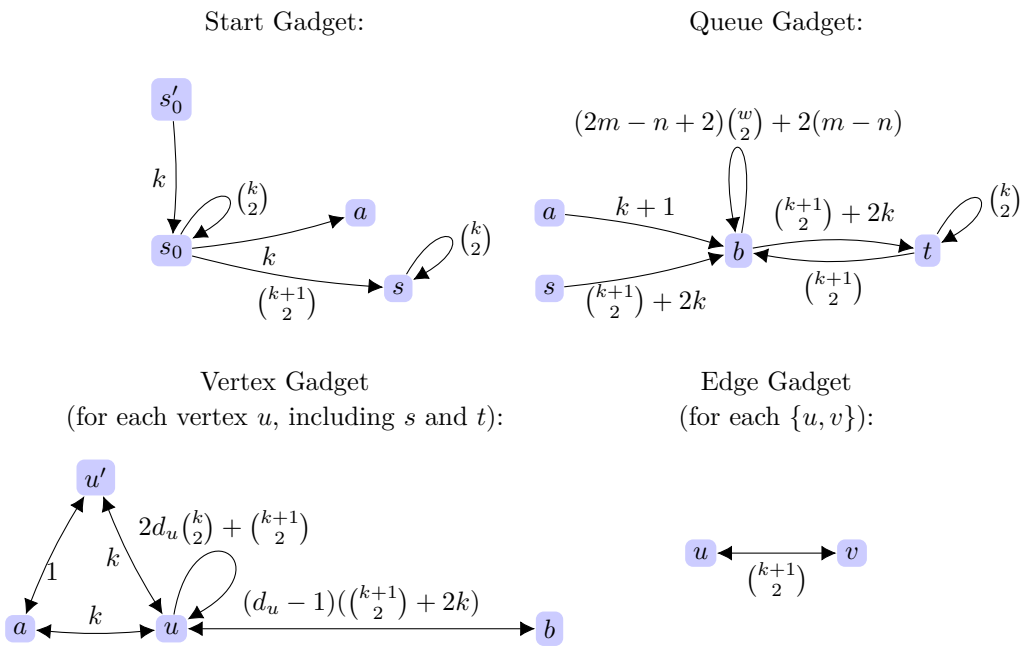


Figure 5 Subgraphs used in the reduction from Hamiltonian Path to DW-Realizable$_3$. Weights on double arcs apply to both directions. Note that some arcs appear in different gadgets, in which case the weights should be summed (in particular, so loops on s and t have total weight $2d_u\binom{k}{2}+\binom{k+1}{2}+\binom{k}{2}$)

Note that this variant of HP is easily shown to be NP-hard from Hamiltonian cycle via the following reduction: given a graph G on which we need to find a hamiltonian cycle, pick any vertex v, duplicate it into v_1, v_2 (each edge $\{u,v\}$ becomes two edges $\{u,v_1\}$ and $\{u,v_2\}$), and add pending vertices s and t connected to v_1 and v_2 respectively.

Reduction for DW-Realizable

Given $G=(V,E)$ with degree-1 vertices s and t, build a directed weighted graph $G'=(V',A)$ as follows:

Vertex set. For each $u \in V$, create a vertex denoted u'. Create two additional dummy vertices a and b. Let $V' := \{a,b,s_0,s'_0\} \cup \bigcup_{u\in V}\{u,u'\}$. The arcs are given in Figure 5, as the union of the start gadget, the queue gadget, and the vertex and edge gadgets respectively for each vertex and edge of G.

Reduction for GW-Realizable

Build the directed graph G' as above, and let G'_u be the undirected version of G': remove arc orientations, for $u \neq v$ the weight of $\{u,v\}$ is the sum of the weight of (u,v) and (v,u) in G' (the weight of loops is unchanged).

Main claims

We prove the following three claims:

(i) G hamiltonian $\Rightarrow$ G' has a realization

(ii) G' has a realization $\Rightarrow$ G'_u has a realization

(iii) G'_u has a realization $\Rightarrow$ G hamiltonian

All together, they show the correctness of the reductions for both GW-Realizable and DW-Realizable since they yield :

G hamiltonian $\Leftrightarrow$ G' has a realization

G hamiltonian $\Leftrightarrow$ G'_u has a realization

Proof of Claim (i). G has a hamiltonian path, let $(u_1 = s, u2, \ldots, u_n = t)$ be its hamiltonian path and $(v_1, w_1), \ldots (v_{m'}, w_{m'})$ be the pairs of connected verices except pairs (u_i, u_{i+1}) (i.e. the set $\bigcup_{\{u,v\}\in E}\{(u,v),(v,u)\} \setminus \{(u_i, u_{i+1}) \mid 1 \leq i < n\}$. Note that $m' = 2m - (n-1)$. Define sequence S as follows.

$$S := \left| \begin{array}{l} s'_0\, s_0^k\, a\, s^k\, s'\, s^k\, a\, u_2^k\, u'_2\, u_2^k\, a \ldots a\, u_{n-1}^k\, u'_{n-1}\, u_{n-1}^k\, a\, t^k\, t'\, t^k\, a \\ b^w\, v_1^k\, b\, w_1^k\, b^w\, v_2^k\, b\, w_2^k \ldots b^w\, v_{m-n}^k\, b\, w_{m-n}^k\, b^w \end{array} \right.$$

Note that a sequence of the form $x^k\, a\, y^k$ yields $\binom{k}{2}$ loops for x, $\binom{k}{2}$ loops for y, as well as $\binom{k+1}{2}$ arcs (x, y) (indeed, there are $1 + 2 + \ldots + w - 2 = \binom{k+1}{2}$ such arcs). A sequence of the form $b\, x^k\, b^w$ yields in particular an arc (b, x) of weight k and arc (x, b) of weight $\binom{k+1}{2} + k$. ◀

Proof of Claim (ii). Clear, any realization for G' is a realization for G'_u. ◀

Proof of Claim (iii). Pick a realization S of G'_u. Define the weight of a vertex in G_u as the sum of the weights of its incident edges (counting loops twice). From the construction, we obtain the following weights for a selection of vertices:

- s'_0 has weight $w - 1$
- u' has weight $2(w-1)$ for $u \in V$
- a has weight $2(n+1)(w-1)$

From the weight of s'_0, it follows that this vertex must be an endpoint of S (wlog, S starts with s'_0). It follows that for any other vertex v with weight $2i(w-1)$, v must have exactly i occurrences in S (in general it can be either i or $i+1$, but if v has $i+1$ occurrences it must be both the first and last character of S, i.e. $v = s'_0$: a contradiction). Thus each u' occurs once and a occurs $n+1$ times in S.

Each u' occurs once, so order vertices of V according to their occurrence in S (i.e. $V = \{u_1, \ldots, u_n\}$ with u'_1 appearing before u'_2, etc.). For each i, the neighborhood of u'_i in S contains a twice, one a on each side (since there is no (a, a) loop). Other neighbors of u'_i may only be occurrences of u_i, so each u'_i belongs to a factor, denoted X_i, of the form $a u_i^* u'_i u_i^* a$. Two consecutive factors X_i, X_{i+1} may overlap by at most one character (a), and if they do, then there exists an edge $\{u_i, u_{i+1}\}$ (since $w \geq 3$) in G. There are n such factors X_{u_i}, and only $n+1$ occurrences of a, so all as except extreme ones belong to the overlap of two consecutive X_is, and there exists an edge $\{u_i, u_{i+1}\}$ for each i. Thus $(u_1, \ldots, u_n)$ is a hamiltonian path of G. ◀

4 Effective general algorithms

4.1 REALIZABLE$_w$ Linear integer programming formulation

Let $G = (V, E)$ be a graph with integer weights $\pi_{e\in E}$. In this model, we represent a sequence x over the alphabet $\{1, ...n\}$, as a $(0-1)$ matrix $X \in \mathbb{M}_{n,p}(\{0,1\})$ encoding the sequence x:

$$X_{i,j} = \begin{cases} 1 & \text{if } x_j = i \\ 0 & \text{otherwise} \end{cases}$$

It should be noted that the set sequence of sequences over the alphabet $\{1, ...n\}$ is exactly represented by the $(0-1)$ matrices such that

$$\forall j \in \{1, ..., p\} \qquad \sum_{i=1}^{n} X_{i,j} = 1$$

Given a window size w, a unit of $\pi_{e=(v_1,v_2)}$ corresponds to the appearance of two elements v_1, v_2 at a distance $i \in \{1, ..., w-1\}$ in the sequence. Now, let us consider a fixed distance i, and a starting index $j \in \{1, ..., p-i\}$, we use a intermediary slack variable $y_j^e(i) \in \{0, 1\}$ to model the presence of such appearance using the constraint:

$$X_{v1,j} X_{v_2,j+i} = y_j^e(i) \tag{8}$$

Then, the Boolean variable $y_j^e(i)$ is equal to 1 when v_1 is located at position j and v_2 at position $j+i$. We linearise Eq. 8 as:

$$\begin{aligned} -X_{v_1,j} \qquad\qquad + y_j^e(i) &\leq 0 \\ - X_{v_2,j+i} + y_j^e(i) &\leq 0 \\ X_{v_1,1} + X_{v_2,j+i} - y_j^e(i) &\leq 1 \end{aligned} \tag{9}$$

Each slack variable $y_k^e(i)$ is attributed to an edge e, a relative distance $i \in \{1, ..., w-1\}$ and a starting position $k \in \{1, ..., p-i\}$. Given our constraint formulation, every slack variable is attributed 3 constraints. For a digraph, the number of possible pair positions for a unit of $\pi_{e=(v_1,v_2)}$ is given by:

$$C = \sum_{i=1}^{w-1} (p-i) = p(w-1) - \frac{w(w-1)}{2} = (w-1)(p - \frac{w}{2})$$

Therefore, in our model, C corresponds to the number of slack variables attributed to constraints for an edge of the graph.

On the contrary, the absence of an edge $e = (v_1, v_2)$, corresponding to $\pi_e = 0$, can be modeled for a distance $i \in \{1, ..., w-1\}$ and a starting position $j \in \{1, ..., p-i\}$ as:

$$X_{v1,j} + X_{v_2,j+i} \leq 1$$

Then, $\textsc{Realizable}_w$ can be formulated as the following linear integer program:

$$\min_{X \in \{0,1\}^{p \times n}, y \in \{0,1\}^{|E| \times C}} \sum_{e \in E} \sum_{i \in \{1,...,w-1\}} y_1^e(i) + ... + y_{p-i}^e(i)$$

under the constraints

$$\forall j \in \{1, ..., p\} \qquad \sum_{i=1}^{n} X_{i,j} = 1$$

$$\begin{array}{l} \forall e = (v_1, v_2) \in E \\ \forall e^{'} = (v_1^{'}, v_2^{'}) \notin E \\ \forall i \in \{1, ..., w-1\} \end{array} \left\{ \begin{array}{lll} -X_{v_1,1} \qquad\qquad + y_1^e(i) \leq 0 & & \\ \qquad - X_{v_2,1+i} + y_1^e(i) \leq 0 & & \\ X_{v_1,1} + X_{v_2,1+i} - y_1^e(i) \leq 1 & & X_{v_1^{'},1} + X_{v_2^{'},1+i} \leq 1 \\ \vdots & & \vdots \\ -X_{v_1,p-i} \qquad\qquad + y_{p-i}^e(i) \leq 0 & & X_{v_1^{'},p-i} + X_{v_2^{'},p} \leq 1 \\ \qquad - X_{v_2,p} + y_{p-i}^e(i) \leq 0 & & \\ X_{v_1,p-i} + X_{v_2,p} - y_{p-i}^e(i) \leq 1 & & \end{array} \right.$$

$$\text{and } \forall e \in E \qquad \sum_{i \in \{1,\dots,w-1\}} y_1^e(i) + \dots + y_{p-i}^e(i) \geq \pi_e$$

If the objective function reaches $\sum_{e \in E} \pi_e$ at its minimum then the output of $\textsc{Realizable}_w(G, \Pi)$ is True, and False otherwise.

4.2 $\textsc{NumRealizations}_w$ Dynamic programming formulation

We did not present a way to count admissible sequences in the general case. Although the tractability of our problems (NP-hardness of $\textsc{Realizable}_w$, #P-hardness of $\textsc{NumRealizations}_w$) currently remains open for some cases, we present in this subsection a method based on dynamic programming valid for all cases.

The recursion proceeds by extending a partial sequence, initially set to be empty, keeping track of for represented edges along the way. Namely, consider $N_w[\Pi, p, \mathbf{u}]$ to be the number of w-admissible sequences of length p for the graph $G = (V, E)$, respecting a weight matrix $\Pi = (\pi_{ij})_{i,j \in V^2}$, preceded by a sequence of nodes $\mathbf{u} := (u_1, \dots, u_{|\mathbf{u}|}) \in V^\star$. It can be shown that, for all $\forall p \geq 1$, $\Pi \in \mathbb{N}^{|V^2|}$ and $\mathbf{u} \in V^{\leq w}$, $N_w[\Pi, p, \mathbf{u}]$ obeys the following formula, using the notations of Section 3.2:

$$N_w\left[\Pi, p, \mathbf{u}\right] = \sum_{v \in V} \begin{cases} N_w\left[\Pi'_{(\mathbf{u},v)}, p-1, (u_1, ..., u_{|u|}, v)\right] & \text{if } |\mathbf{u}| < w-1 \\ N_w\left[\Pi'_{(\mathbf{u},v)}, p-1, (u_2, ..., u_{w-1}, v)\right] & \text{if } |\mathbf{u}| = w-1 \end{cases} \tag{10}$$

with $\Pi'_{(\mathbf{u},v)} := (\pi_{ij} - |\{k \in [1, |\mathbf{u}|] \mid (u_k, v) = (i, j)\}|)_{(i,j) \in V^2}$. The base case of this recurrence corresponds to $p = 0$, and is defined as

$$\forall\, \Pi,\ N_w[\Pi, 0, \mathbf{u}] = \begin{cases} 1 & \text{if } \Pi = (0)_{(i,j) \in V^2} \\ 0 & \text{otherwise.} \end{cases} \tag{11}$$

The total number of admissible sequences is then found in $N_w[\Pi, p, \varepsilon]$, *i.e.* setting $\mathbf{u}$ to the empty prefix ε, allowing the sequence to start from any node.

The recurrence can be computed in $\mathcal{O}(|V|^w \times \prod_{i,j \in V^2}(\pi_{i,j} + 1))$ time using memoization, for p the sequence length. The complexity can be refined by noting that:

$$\sum_{i,j \in V^2} \pi_{i,j} \leq w \times p$$

To investigate the worst case scenario, we can consider the optimisation problem:

$$\max_\Pi \ \prod_{i,j \in V^2}(\pi_{i,j} + 1) \text{ such that } \ \sum_{i,j} \pi_{i,j} = w\, p. \tag{12}$$

This problem is equivalent to maximise a product under a budget constraint. When $n^2 \geq w \times p$, which is the case in practice, the maximum is reached for a Boolean matrix $\Pi = (\pi_{i,j}) \in \{0, 1\}^{|V|^2}$, verifying the constraint. This property can be deduced from the inequality:

$$\begin{aligned} 1 \leq a < b - 1 &\implies \log a + \log b < \log(a+1) + \log(b-1) \\ &\implies ab < (a+1)(b-1) \end{aligned}$$

It follows that, in the worst-case scenario, $\prod_{i,j \in V^2}(\pi_{i,j} + 1) \in \mathcal{O}(2^{w\,p})$. Thus, despite the, apparently extreme complexity of our algorithm, it is still possible to compute $N_w[\Pi, p, u_{1:w}]$ for "reasonable" values of p and w.

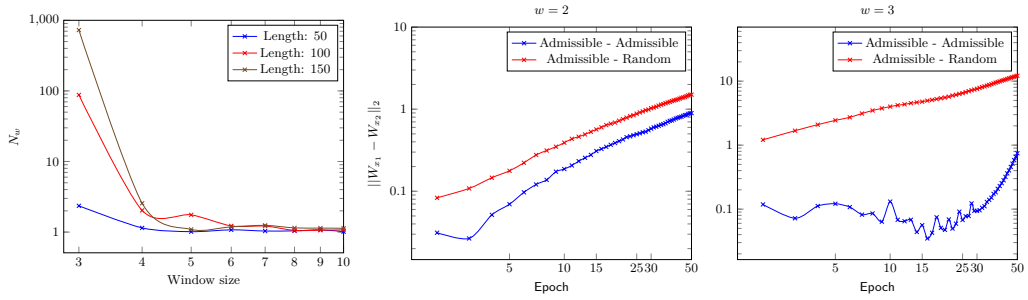


Figure 6 Left plot: Average lower bound (N_w) on the average number of sequences. Two right plots: $||W_{x_1} - W_{x_2}||_2 = f(\text{Epoch})$ - LSTM, $\log - \log$ scale

5 Application to sequential models

5.1 Number of equivalent sequences for weighted sequence digraphs

Since the dynamic programming method in Sec. 4.2 is exponential in the wort case, we provide results for relatively short sequences generated from text data (a dump of English Wikipedia, 2016) of 500 documents, each of them having a length $p \in \{50, 100, 150\}$. Each document contained a minimum of $\frac{3}{4}p$ distinct words. For each $w \in 3 \to 10$, we estimate the number of admissible sequences yielding the same representations for a set of documents and different window size using the procedure described in Sec. 4.2 to compute N_w. Due to memory limitations, it should be noted that N_w is a lower bound of the number of total admissible sequences, since a starting pattern (first w tokens) is fixed.

Results are reported in the left plot of Fig. 6. For $w = 2$, the number of sequences (obtained using Prop. 19) was significantly larger ($> 10^5$), so not reported in the figure for clarity. As expected, the number of distinct admissible sequences tends to 1 when the window size increases. This suggests that window sizes used in skip gram models should be usually larger or equal to 5. In natural language processing, a frequent configuration is $w = 10$ [18, 21]. However, some examples with different realizations exist, even for $w = 10$ and $p = 50$.

5.2 Comparison with a recurrent neural network

The second experiment we consider is to evaluate the difference of the parameters between a sequential model trained on two admissible sequences of a given graph. The sequential model we are considering are a class of recurrent dynamical recurrent models, referred to as long short term memory (LSTM) networks [12]. These models have attracted new interest due to experimental progress for time series prediction [11] and natural language processing [3, 6, 25]. Given a window size w, the task we consider is to predict the next element of the sequence given the $w - 1$ previous ones. If the sequences were equivalent for the sequential model, the weights should numerically converge after training.

To generate pairs of admissible sequences encoding for non trivial graphs (i.e not the complete graph), we used algorithm based on Lemma 5. We generate w-admissible sequence (thousand tokens long), for $w \in \{2, 3\}$, but could not provide other pairs for $w > 3$ due to computational time. We compare the pairs of admissible sequences with a pair of one of the sequence, and a sequence generated randomly uniformly on the same vocabulary. We implemented the LSTM network using the Python library Keras [7], a high-level API

running over TensorFlow [1]. In order to remove randomness from the training algorithm, we froze the seed generating initial weights (the optimization directions being fixed by the data). We chose tanh as main activation function, sigmoid for the recurrent activations, with 2 units. The number of units is chosen relatively low in order to obtain a reasonable number of weights (in this case 16).

Two right plots of Fig. 6 reports the results for $w \in \{2, 3\}$, W_x represents all the weights of the network for a sequence x. For $w = 2$, there the difference of the weights for 2 admissible sequences is lower than with one of the sequence and a random one, but this proximity does not appear to be significant compared with a random sequence. For $w = 3$, the recurrent network has relatively close weights for two admissible sequences when compared with a random one.

6 Conclusion

In this study, we revisited a series of problems related to the ambiguity of sequence graphs representations, which are popular in the context of text mining and natural language processing. We derived theoretical properties and practical algorithms for the family of sequence graphs.

This study can be of use used for several sequential models, such as continuous bag of words (CBOW), skip-grams ([10, 14, 24]), pointwise mutual information models [2, 18, 21].

Acknowledgments

The authors wish to express their gratitude to Guillaume Fertin and an anonymous reviewer of an earlier version of this manuscript, for their valuable suggestions and constructive criticisms.

References

1 Martın Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous systems, 2015. *Software available from tensorflow. org*, 1(2), 2015.

2 Sanjeev Arora, Yuanzhi Li, Yingyu Liang, Tengyu Ma, and Andrej Risteski. A latent variable model approach to pmi-based word embeddings. *Transactions of the Association for Computational Linguistics*, 4:385–399, 2016.

3 Sergey Bartunov, Dmitry Kondrashkin, Anton Osokin, and Dmitry Vetrov. Breaking sticks and ambiguities with adaptive skip-gram. In *artificial intelligence and statistics*, pages 130–138, 2016.

4 Graham R Brightwell and Peter Winkler. Counting eulerian circuits is# p-complete. In *ALENEX/ANALCO*, pages 259–262. Citeseer, 2005.

5 Seth Chaiken. A combinatorial proof of the all minors matrix tree theorem. *SIAM Journal on Algebraic Discrete Methods*, 3(3):319–329, 1982.

6 Qian Chen, Xiaodan Zhu, Zhenhua Ling, Si Wei, Hui Jiang, and Diana Inkpen. Enhanced lstm for natural language inference. *arXiv preprint arXiv:1609.06038*, 2016.

7 François Chollet et al. Keras. https://keras.io, 2015.

8 Nicolaas Govert de Bruijn and Tanja van Aardenne-Ehrenfest. Circuits and trees in oriented linear graphs. *Simon Stevin*, 28:203–217, 1951.

9 Jaume Gibert, Ernest Valveny, and Horst Bunke. Dimensionality reduction for graph of words embedding. In *International Workshop on Graph-Based Representations in Pattern Recognition*, pages 22–31. Springer, 2011.

10 Yoav Goldberg and Omer Levy. word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722*, 2014.

11 Klaus Greff, Rupesh K Srivastava, Jan Koutník, Bas R Steunebrink, and Jürgen Schmidhuber. Lstm: A search space odyssey. *IEEE transactions on neural networks and learning systems*, 28(10):2222–2232, 2016.

12 Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

13 Leo Liberti, Carlile Lavor, Nelson Maculan, and Antonio Mucherino. Euclidean distance geometry and applications. *Siam Review*, 56(1):3–69, 2014.

14 Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

15 Tomáš Mikolov, Wen-tau Yih, and Geoffrey 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.

16 David Nadeau and Satoshi Sekine. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26, 2007.

17 Hao Peng, Jianxin Li, Yu He, Yaopeng Liu, Mengjiao Bao, Lihong Wang, Yangqiu Song, and Qiang Yang. Large-scale hierarchical text classification with recursively regularized deep graph-cnn. In *Proceedings of the 2018 World Wide Web Conference*, pages 1063–1072, 2018.

18 Jeffrey Pennington, Richard Socher, and Christopher 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.

19 Michael Roth and Kristian Woodsend. Composition of word representations improves semantic role labelling. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 407–413, 2014.

20 François Rousseau, Emmanouil Kiagias, and Michalis Vazirgiannis. Text categorization as a graph classification problem. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1702–1712, 2015.

21 Arora Sanjeev, Liang Yingyu, and Ma Tengyu. A simple but tough-to-beat baseline for sentence embeddings. *Proceedings of ICLR*, 2017.

22 Micha Sharir. A strong-connectivity algorithm and its applications in data flow analysis. *Computers & Mathematics with Applications*, 7(1):67–72, 1981.

23 Konstantinos Skianis, Fragkiskos Malliaros, and Michalis Vazirgiannis. Fusing document, collection and label graph-based representations with word embeddings for text classification. In *Proceedings of the Twelfth Workshop on Graph-Based Methods for Natural Language Processing (TextGraphs-12)*, pages 49–58, 2018.

24 Yan Song, Shuming Shi, Jing Li, and Haisong Zhang. Directional skip-gram: Explicitly distinguishing left and right context for word embeddings. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 175–180, 2018.

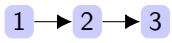
25 Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

A Additional figures

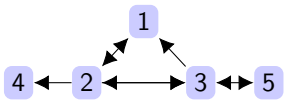
Algorithm 1 Construction of a sequence digraph

Input: Sequence x of length p, window size w, $p \geq w \geq 2$
Parameter: Optional list of parameters
Output: $(G_w(x), \Pi)$

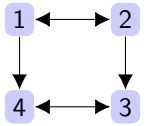
$V \leftarrow \emptyset$
Initiate $\Pi = (\pi_{i,j})$ to $d \times d$ matrix of zeros
for $i = 1 \rightarrow p - 1$ **do**
  $V \leftarrow V \cup \{x_i, x_{i+1}\}$
  for $j = i + 1 \rightarrow \mathsf{min}(i + w - 1, p)$ **do**
    $\pi_{x_i,x_j} \leftarrow \pi_{x_i,x_j} + 1$
  end for
end for
return solution



(a) 1 2 3 is a 2-admissible sequence but is G is not strongly connected

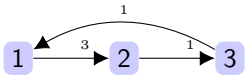


(b) 3 5 3 1 2 1 2 3 2 4 is a 2-admissible sequence but the graph is not Eulerian nor semi-Eulerian



(c) G is not a 2-sequence graph...

(d) ... whereas $R(G)$ is.



(e) G is strongly connected but is not a 2-sequence graph

Figure 7 Counter examples for $w = 2$

B Results and proofs Sec. 3.1 ($w = 2$)

In this section, we present the results for digraphs and $w = 2$. Obviously, the simplest case concerns undirected graphs as stated in:

▶ **Proposition 8.** *If $G = (V, E)$ is unweighted and undirected, with $|V| > 1$, the following are equivalent:*
(i) G is connected
(ii) G has a 2-admissible sequence
(iii) G admits an infinite number of 2-admissible sequences
In these conditions, a 2-admissible sequence can start and end at any vertex.

Proof. Let us suppose G has an admissible sequence u. Let a, b two distinct vertices of G. Then using the definition, a and b must appear at least once in the sequence u, i.e $u_{i_a} = a$ and $u_{i_b} = b$. If $i_a < i_b$, then the sequence $s = (u_i \mid i_a \leq i \leq i_b)$ defines a path from a to b since $\forall i,\ e_{s_i s_{i+1}} \in E$. The case $i_b > i_a$ is dealt similarly. ◀

The previous characterization is wrong for digraphs, even with strongly connectivity. A counter example is depicted in Fig. 7a. However, strong connectivity remains a sufficient condition:

▶ **Proposition 9.** *Let $G = (V, E)$ a unweighted digraph. If G is strongly connected then $G \in \operatorname{Im} \phi_2$. A 2-admissible sequence can start or end at any given vertex of G.*

Proof. This can be proved similarly to (i) $\implies$ (ii) for proposition 9 by replacing connectivity with strong connectivity. ◀

▶ **Proposition 10.** *Let $G = (V, E)$ an unweighted digraph. If G is Eulerian or semi-Eulerian, then $G \in \operatorname{Im} \phi_2$.*

Proof. If G is Eulerian or semi-Eulerian, there exists a walk going through all edges, this walk defines a 2-admissible sequence. ◀

Again the converse of Prop. 10 does not hold as depicted in Fig. 7b. The characterization of sequence digraph is more subtle. As a start, it is natural to consider directed acyclic graphs (DAGs):

▶ **Proposition 11.** *Let $G = (V, E)$ a DAG. G is a 2-sequence graph if and only if it is a directed path, i.e G is a directed tree where each node has at most one child and at most one parent. In this case, G has a unique 2-admissible sequence.*

Proof. If G is directed path, since G is finite, it admits a source node. Therefore a 2-admissible sequence is obtained by simply going through all vertices from the source node. This is obviously the only one.

Conversely, let us suppose G is a DAG and a 2-sequence graph. If G is not a directed path, there are two cases: either there exists a vertex having two children, or two parents. Let s be a vertex having 2 distinct children c_1 and c_2. This is not possible since there cannot be a walk going through (s, c_1) and (s, c_2): G would have a cycle otherwise. Finally a vertex v cannot have two parents p_1 and p_2: if a 2-admissible sequence existed, it would have to go through (p_1, v) and (p_2, v), creating a cycle, hence the contradiction. ◀

Every directed graph G is a DAG of its strongly connected components. In the following, let $R(G)$ be the DAG obtained by contracting the strongly connected components of G.

▶ **Proposition 12.** *Let $G = (V, E)$ a digraph. If G is a 2-sequence graph then $R(G)$ is a 2-sequence graph.*

Proof. Let G be a 2-sequence graph, and let us suppose that $R(G)$ is not a 2-sequence graph. Since $R(G)$ is a (weakly) connected DAG, then using Prop. 11, it cannot be a directed path, so $R(G)$ has either a node having two children or two parents. Let S be a node of $R(G)$ having at least 2 distinct children C_1 and C_2. This means that there exist three distinct corresponding nodes in V, s, v_1 and v_2 such that $(s, v_1) \in E$ and $(s, v_2) \in E$. Since G is a 2-sequence graph, there exists a walk covering (s, v_1) and (s, v_2), such walk would make S, C_1 and C_2 the same node in $H(G)$, hence the contradiction. The case for which a vertex has two parents is dealt with similarly. ◀

The converse of Prop. 12 does not hold as depicted in Fig. 7c, 2. However, let us add a weight compatibility in $R(G)$ as follows:

▶ **Definition 13.** *Let G be a digraph, and $R^+(G)$ be the weighted DAG obtained from $R(G)$, such that the weight of an edge is attributed the number of distinct arcs from two strongly connected components in G.*

▶ **Theorem 14.** *Let $G = (V, E)$ be an unweighted digraph.*

G is a 2-sequence graph if and only if $R^+(G)$ is a directed path and its weights are all equal to 1.

Proof. If G is a 2-sequence graph, $R(G)$ is a 2-sequence graph using Prop. 12. Also Prop. 11 implies that $R(G)$ and $R^+(G)$ are directed paths. Moreover, if $R^+(G)$ had a weight strictly greater that 1, then there would be strictly more than one edge between two connected components C_1 and C_2. All these edges go in the same direction otherwise $C_1 \cup C_2$ would form a strongly connected component. This is a contradiction since any 2-admissible sequence would have to go from C_1 to C_2 and then come back to C_1 (or conversely) which would would make $C_1 \cup C_2$ a strongly connected component.

Conversely, let us suppose $R^+(G)$ is a a directed path and its weights are equal to one. First, there exists a walk $x_1, ..., x_p$ covering all edges of $R^+(G)$ verifying: (i) $\forall i$, $x_i \in V$ or x_i represents a strongly connected component of G, (ii) there is only one edge in G between from x_i to x_{i+1} and (iii) x has no repetition, i.e there is no common vertex in G between x_i and x_{i+1}. We construct a 2-admissible sequence y for G by means of the following procedure.

Initialisation: If $x_1 \in V$, we simply set $y \leftarrow x_1$. Otherwise, x_1 corresponds to a strongly connected component C_1 of G and we add to y any 2-admissible sequence of C_1.

For $i \in \{1, .., p-1\}$:

- If $(x_i, x_{i+1}) \in E$: we add x_{i+1} to the sequence y.
- If $x_i \in V$ and x_{i+1} is a strongly connected component C_i of G: By assumption, there exists only one edge of G from x_i to a vertex of C_i, say c_0^i. Since C_i is strongly connected, using Prop. 9, C_i has a walk going through all of its edges and starting in c_0^i, say $c_0^i, ..., c_p^i$. We add $c_0^i, ..., c_p^i$ to y.
- If x_i corresponds to a strongly connected component C_i and $x_{i+1} \in V$: we perform similar operations by stopping on the single node of C_i that has a edge to x_{i+1} (this is possible thanks to Prop. 9).
- x_i and x_{i+1} both correspond to strongly connected components C_i and C_{i+1} , there exists only one edge between in E between C_i and C_{i+1}, say $e_i = (v_i, v_{i+1})$. We can complete y by a walk from the last vertex visited which belong to C_i and v_i, and then by a 2-admissible sequence through C_{i+1} starting in v_i and ending in v_{i+1}.

End For

The process stops when $i = p - 1$, and all edges are covered by the sequence y. ◀

Therefore, an algorithm to decide if a digraph is a 2-sequence is obtained by extract its connected components (there exist linear time algorithms e.g [22]), and to count the number of distinct edges between these.

▶ **Corollary 15.** *Let G an unweighted digraph. The possible numbers of 2-admissible sequences for G is exactly $\{0, 1, +\infty\}$. Moreover, G admits a unique 2-admissible sequence if and only if G is a directed path.*

Proof. Let G a 2-sequence graph and let us show that G has either a unique or an infinite number of 2-admissible sequence. G verifies characterization of Theorem 14. If $R(G)$ has

a vertex representing a strongly connected component of G (or a vertex with a self loop), then by adding an arbitrary number of cycles to y, the obtained walk is still admissible. Otherwise, if every vertex of $R(G)$ is in V without self-loops in E, then G is a DAG. Using Prop. 11, y is the unique 2-admissible sequence. ◀

Weighted 2-sequence graphs

The weighted case cannot be treated similarly due to the weight constraints implying that a weighted graph has a finite number of admissible sequences. A counter example is depicted in Fig. 7e.

▶ **Definition 16.** *Let $\psi(G)$ be the multigraph with the same vertices as $G = (V, E)$ and with π_{ij} edges between $(i, j) \in V^2$.*

Due to the previous study, the characterization of weighted 2-sequence graphs using $\psi(G)$ is immediate.

▶ **Theorem 17.** *If G is a weighted graph (directed or not), with $\Pi(G) \in \mathcal{M}_d(\mathbb{N})$, then: $G \in \operatorname{Im}\phi_2 \iff \psi(G)$ is connected and semi-Eulerian.*

Proof. $G \in \operatorname{Im}\phi_2$ means that there is a trail going through each edge $(i, j) \in E$ exactly $\pi_{i,j}$ times. This trail corresponds to a semi-Eulerian path in $\psi(G)$. ◀

▶ **Lemma 18.** *Let $G = (V, E)$ a weighted 2-sequence graph (possibly oriented). Let $\mathcal{E}$ be the set of Eulerian paths of $\psi(G)$ and $\mathcal{S}$ be the set of 2-realizations of G. Then*

$$\mathcal{E} = (\#\mathcal{S}) \prod_{e \in E} \pi_e! \tag{13}$$

Proof. We will first prove it for digraphs. If $e = (v_1, v_2)$ is an edge of a digraph, we will represent the source and target vertex of e as $e(s)$ and $e(t)$. Let $(e_1, e_2, ..., e_h)$ be a Eulerian path of $\psi(G)$ defined as a sequence of its edges. Then $\forall (i, j) \in \{1, ..., h\}^2$, $e_i \neq e_j$ and $\forall i \in \{1, ..., h-1\}$, $e_i(t) = e_{i+1}(s)$. Let us consider the transformation:

$$\begin{gathered} \mathcal{E} \longrightarrow \mathcal{S} \\ (e_1, e_2, ..., e_h) \mapsto (e_1(s), e_2(s), ..., e_{p-1}(s), e_h(t)) \end{gathered} \tag{14}$$

We have already shown this transformation is surjective: any 2-sequence of G can be obtained with a Eulerian path of $\psi(G)$. We will now consider the action of $\mathfrak{S}_h$ on $\mathcal{E}$. For a Eulerian path, let us suppose that two edges of $\psi(G)$ have been permuted, say e_1 and e_{i_0} without loss of generality. If the two corresponding sequences are the same: $(e_{i_0}(s), e_2(s), ..., e_h(t)) = (e_1(s), e_2(s), ..., e_{i_0}(s), ..., e_h(t))$. Obviously, $e_{i_0}(s) = e_1(s)$. Also $e_1(t) = e_2(s)$ implies $e_{i_0}(t) = e_1(t)$. This shows that e_{i_0} and e_1 are associated to the same edge in E. Therefore, given a 2-sequence, the choice of a corresponding Eulerian path correspond to the choice of $\sigma = (\tau_1, ..., \tau_{|E|})$ where τ_e is a permutation of $\{1, ..., \pi_e\}$. Therefore $\#\mathcal{E} = (\#\mathcal{S}) \prod_{e \in E} \pi_e!$.

If G is undirected, the proof is still valid, but the operators $e \mapsto e(s)$ and $e \mapsto e(t)$ are now induced the natural direction of the considered Eulerian path. ◀

Counting the number of Eulerian paths in a undirected graph has been proven to be a $\#P$-complete problem [4]. Since $G \mapsto \psi(G)$ is bijective, counting the number of 2-admissible sequences is also $\#P$-complete. Finally, counting Eulerian trails of weighted digraphs has been well studied, hence the following proposition:

▶ **Proposition 19.** *If $G = (V, E)$ is a weighted digraph, with $\Pi(G) \in \mathcal{M}_d(\mathbb{N})$. Then, if $deg(v)$ is the indegree of a vertex v, the number p_2 of 2-admissible sequences is*

$$p_2 = \frac{t(\psi(G))}{\prod_{e\in E}\pi_e!}\prod_{v\in V}\big(\deg_{\psi(G)}(\psi(v)) - 1\big)! \tag{15}$$

where $t(G)$ is the number of spanning trees of a graph G. If L is the Laplacian matrix of G, then $t(G)$ is given by

$$t(G) = \prod_{\substack{\lambda_i\in Sp(L)\\ \lambda_i\neq 0}} \lambda_i \tag{16}$$

Proof. Direct consequence of BEST Theorem [8]) and Matrix tree theorem [5]) ◀

To use formula 15, $\deg_{\psi(G)}(\psi(v))$ can be obtained using the following formula: $deg_{\psi(G)}(\psi(v)) = \sum_{n\in V}\pi_{nv} + \sum_{n\in V}\pi_{vn}$.

C Proofs section 3.2 ($w \geq 3$)

Proof of Proposition 3. Let $x = x_1, ..., x_p$ be a w-admissible sequence of G. Let P be a walk on $H^{(w-2)}$, and $P[i]$ be the i-th element of P, $P[i] \in H^{(w-2)}$: $P[i] = (P[i]_1, ..., P[i]_{w-1})$.

Let us suppose that $w \leq p$ (which we can always do), and let us show the following property by induction on k:

$$\forall k \in \{w-1, ..., p\},\ \exists \text{ walk } P \text{ on } H^{(w-2)} \text{ such that :}$$

$$x_{1:k} = P[1]_1, P[2]_1, ..., P[k-(w-1)]_1, P[k+1-(w-1)]_{1:(w-1)}$$

- Initialisation: $k = w-1$. By construction of $H^{(w-2)}$, $x_{1:w-1}$ is the authentic sequence of "static walk": $P = P[1] = x_{1:w-1} \in H^{(w-2)}$.
- Induction: let us suppose the property is verified for $k \in \{w-1, ..., p-1\}$, i.e there exists a walk P on $H^{(w-2)}$ such that:

$$x_{1:k} = P[1]_1, P[2]_2, ..., P[k-(w-1)]_1, P[k+1-(w-1)]_{1:(w-1)}$$

Since x is w-admissible, then by definition:

$$\forall i \in \{k+1-(w-1), ..., k\},\ \forall j \in \{i+1, ..., \mathsf{min}\{k+1, i+w-1\}\} :\ (x_i, x_j) \in E$$

Therefore, by definition of $H^{(w-2)}$, $\xi^{k+1} = x_{k+1-(w-1)}, ..., x_{k+1} \in H^{(w-2)}$.

Let $P[k+2-(w-1)] \mathrel{\widehat{=}} \xi^{k+1}$, then

$$P[k+2-(w-1)]_{1:(w-1)} = x_{k+1-(w-1)}, ..., x_{k+1}$$

Besides, from the induction assumption: $\forall i \in \{1, ..., k-(w-1)\},\ P[i]_1 = x_i$. This ensures that:

$$x_{1:(k+1)} = P[1]_1, P[2]_1, ..., P[k+1-(w-1)]_1, P[k+2-(w-1)]_{1:(w-1)}$$

which ends the induction and the proof. ◀

Proof of Lemma 5. Let $P = P[1], , ..., P[r]$ a walk on $H^{(w-2)}$ going through a strongly connected component C, with an arbitrary ordering of its vertices, i.e $C = \{c_1, ..., c_m\}$. This means $\exists (m_0, i_0) \in \{1, ..., m\} \times \{1, ..., r-1\}$ s.t $P[i_0] = c_{m_0}$ and $(c_{m_0}, P[i_0+1]) \in E^{(w-2)}$. Let $\mathcal{P}_C = c_{m_0}, c_{j_1}, ..., c_{j_v}$ be a path in C with $(c_{j_v}, P[i_0+1]) \in E^{(w-2)}$. Let Q be the new

path: $Q = P[1], ..., P[i_0], c_{j_1}, ..., c_{j_v}, P[i_0 + 1], ..., P[r]$. By construction of $H^{(w-2)}$, the edges (between elements of V) created by any walk on $H^{(w-2)}$ are in E, so Q is still admissible.

Let us label every node of $R(H^{(w-2)})$ representing a strongly connected component of $H^{(w-2)}$ by any 2−admissible sequence (one exists thanks to Prop. 9). A walk on $H^{(w-2)}$: $x_1, ..., x_p$ can be met by a walk on $R(H^{(w-2)})$ using the following procedure:

For $i \in \{1, ..., p-1\}$:

- if $x_i, x_{i+1} \in E^{(w-2)}$, we keep x_i and x_{i+1}
- if x_i is a vertex of $H^{(w-2)}$ and x_{i+1} is in a strongly connected component of $H^{(w-2)}$ (but a node of $R(H^{(w-2)})$), represented by $c_1, ..., c_{C_i}$, then a path from x_{i+1} to c_1 exists since the component is strongly connected: $x_{i+1}, p_1, ..., p_m, c_1$. We keep

$$x_i, x_{i+1}, p_1, ..., p_m, c_1, ..., c_{C_i}$$

Using the aforementioned result, this does not perturb admissibility.

- if x_{i+1} is a vertex of $H^{(w-2)}$ and x_i is in a strongly connected component of $H^{(w-2)}$, we proceed similarly (x_i and x_{i+1} are swapped).
- if both x_{i+1} and x_i are strongly connected components of $H^{(w-2)}$, we add intermediary nodes to both components similarly.

◀