Sequence graphs realizations and ambiguity in language models

Sammy Khalife, Yann Ponty, Laurent Bulteau

Sequence graphs realizations and ambiguity in language models

Sammy Khalife and Yann Ponty

LIX, CNRS, Ecole Polytechnique, Institut Polytechnique de Paris, 91128 Palaiseau, France
{khalife,yann.ponty}@lix.polytechnique.fr

Abstract. Several natural language models rely on an assumption modeling each word context as a bag of words. We study the combinatorial implications of such assumption for the corresponding word or sentences representations. In particular, we present theoretical results concerning the family of sequence graphs, for which realizations yield equivalent representations given this assumption. 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. Based on these results, we also establish different algorithms, including a dynamic programming formulation, to count and explicit the different realizations of a sequence graph. This allows us to show that the bag of words assumption can induce an important number of sentences to have the same representations, even for relatively short context window sizes.

1 Introduction

Context and motivations. In the fields of Natural Language Processing (NLP) and Information Retrieval (IR), concise representations of words and textual documents are essential for several tasks, including document classification [27], role labelling [21], and named entity recognition [17]. In particular, the Bag-Of-Words (BOW) representations [24,20] encode a text and/or a sentence as a vector of weighted occurrences. Using BOWs, classic tasks such as document similarity computation and indexing can be efficiently computed using sparse linear algebra, leading to their presence at the core of popular software solutions such SMART [23] or Lucene [12].

The invariance of the BOW representation, to permutations of the words in the document, can lead to multiple documents being summarized by the same BOW. Indeed, the number of documents having the same BOW representation grows factorially on the size of the document. Moreover, this high degree of **ambiguity** has practical consequences for fine-grained classification tasks, as empirically shown by [15,27] in the context of multi-label classification. For these reasons, more recent works introduced the **Graph of Words** [9,22,18] (GOW), which supplements the content of BOW with statistics of co-occurrences within a **window** of fixed size w, introduced to mitigate the degree of ambiguity induced by the representation. Several models such as word2vec [16], Glove [19] also use the same type of information and allowed to increase performance for multiple tasks in natural language processing.

While GOW representations are more precise, 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 as an algorithmic problem, coupled with an empirical assessment of the consequences of ambiguity in the context of classification. As a first natural step, we also consider the realizability of a given GOW, *i.e.* the existence of a sequence admitting an input GOW as its representation

After introducing in Section 2 the formal definition of a sequence graph, which is the combinatorial abstraction of a Graph-Of-Word, and descriptions of our main problems, we establish in Section 3 complexity aspects of deciding the existence and counting sequences in GOWs associated with a window size $w = 2$. Then we consider in Section 4 the general case $w \geq 3$, and propose an exponential dynamic programming algorithm to 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 LSTM, a popular neural network.

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*

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

For digraphs, Eq. (1) *is replaced with*

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

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

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

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. A w-sequence weighted digraph can be obtained following algorithm 1; other cases are obtained similarly. The procedure in algorithm 1 defines a correspondence between the sequence set S_X into the graph set $\mathcal{G} : \phi_w : X^\star \to \mathcal{G}, x \mapsto G_w(x)$.

*Problem 1 (*REALIZABLE*).*
Input: Graph G, window size w
Output: True if G is the w-sequence graph of some sequence x, False otherwise.

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

*Problem 2 (*NUMREALIZATIONS*).*
Input: Graph G, window size w
Output: The number of **realizations** of G, *i.e.* preimages of G through ϕ_w such that $|\{x \in X^\star \mid \phi(x) = G\}|$ if finite, or $+\infty$ otherwise.

Note that the last problem strictly generalizes the previous one, as REALIZABLE can be solved by testing the nullity of the number of suitable realization computed by NUMREALIZATIONS. Due to length limitations, full proofs of this paper are left in the appendix.

3 Complexity results over 2-sequence graphs

A graph has a sequential realization with $w = 2$ if and only if it can be drawn without lifting the pen (with possible repetitions). This characterization enables relatively simple algorithmic treatment, leading to the results summarized in Table 1, which we further elaborate in this section.

Obviously, the simplest case concerns undirected graphs as stated in:

Proposition 1. *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.

The previous characterization is wrong for digraphs, even with strongly connectivity. A counter example is depicted in Fig. 1a. However, strong connectivity remains a sufficient condition:

Proposition 2. *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.*

Proposition 3. *Let $G = (V, E)$ an unweighted digraph. If G is Eulerian or semi-Eulerian, then $G \in \operatorname{Im} \phi_2$.*

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

	NUMREALIZATIONS		REALIZABLE	
Data Instance	Complexity	#Sequences	Complexity	Characterization
Unweighted graph	P	$\{0, +\infty\}$	P	G connected
Weighted graph	#P-hard	???	P	$\psi(G)$ (semi)Eulerian
Unweighted digraph	P	$\{0, 1, +\infty\}$	P	Theorem 1
Weighted digraph	P	BEST Theorem	P	$\psi(G)$ (semi)Eulerian

Again the converse of Prop. 3 does not hold as depicted in Fig. 1b. The characterization of sequence digraph is more subtle. As a start, it is natural to consider directed acyclic graphs (DAGs):

Proposition 4. *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.*

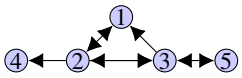
Every directed graph G is a DAG of its strongly connected components. In the following, let $R(G)$ be the DAG obtained by contracting the strongly connected components of G.

Proposition 5. *Let $G = (V, E)$ a digraph. If G is a 2-sequence graph then $R(G)$ is a 2-sequence graph.*

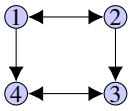
The converse of Prop. 5 does not hold as depicted in Fig. 1c, 1d. However, let us add a weight compatibility in $R(G)$ as follows:

Definition 2. *Let G be a digraph, and $R^+(G)$ be the weighted DAG with the same vertices and edges as $R(G)$, such that the weight of an edge is attributed the number of distinct edges from two strongly connected components in G.*

(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.

Fig. 1: Counter examples for $w = 2$

Theorem 1. *Let $G = (V, E)$ a 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.

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 [26]), and to count the number of distinct edges between these.

Corollary 1. *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.*

3.1 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. 2.

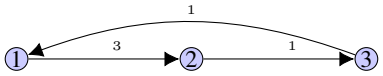


Fig. 2: G is strongly connected but is not a 2-sequence graph

Definition 3. *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 2. *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.*

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. For weighted digraphs, counting Eulerian trails in G has been well studied.

Proposition 6. *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*

$$p_2 = t(\psi(G)) \prod_{v \in V} \big(\deg_{\psi(G)}(\psi(v)) - 1\big)! \tag{4}$$

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

$$t(G) = \prod_{\substack{\lambda_i \in Sp(L) \\ \lambda_i \neq 0}} \lambda_i \tag{5}$$

(a) G is connected (b) G is strongly connected

Fig. 3: Counter-examples for $w = 3$

Proof. Direct consequence of BEST Theorem [5]) and Matrix tree theorem [6]) □

To use formula 4, $\deg_{\psi(G)}(\psi(v))$ can be obtained using the following formula: $deg_{\psi(G)}(\psi(v)) = \sum_{n \in V} \pi_{nv} + \sum_{n \in V} \pi_{vn}$.

4 General sequence graphs

The characterization of 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, which is not the case. Similarly, Fig. 3 depicts a counter example for directed graphs: G does not have self-edges, 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.

4.1 Direct approach

Similarly to the procedure in Sec. 3.1, 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:

$$v_2 = v_3 \;\; \mathsf{and} \;\; (v_1, v_4) \in E \tag{6}$$

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:

$$P = (t_1, t_2), ..., (t_{p-1}, t_p) \;\; \mathsf{s.t} \;\; \forall i \in \{1, ..., p-1\}, \; (t_i, t_{i+1}) \in E \tag{7}$$

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. 6. 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

$$E^{(k)} = \{u_{1:k+1} \in V^{k+1} \mid u_{1:k} \in E^{(k-1)}, u_{2:k+1} \in E^{(k-1)} \wedge (u_1, u_{k+1}) \in E\} \tag{8}$$

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

$$H^{(0)} = G \qquad\qquad \forall k \in \mathbb{N}^*, \;\; H^{(k)} = f(H^{(k-1)}) \tag{9}$$

where f transforms edges into vertices and creates edges between new vertices that verify Eq. 8.

Definition 4. *Let u be a vertex of $H^{(k)}$ for $k \in \mathbb{N}$, $u = (u_1, ..., u_k, u_{k+1})$. By definition, 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 7. *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: 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 3. *Let G a graph and $w \in \mathbb{N}^* - \{1, 2\}$. If G is undirected then* REALIZABLE *is in P.*

Proof. An algorithm is obtained by going through all the connected components of $H^{(w-2)}$. Let $C_1, ..., C_m$ the connected components of $H^{(w-2)}$. It is possible to compute them in polynomial time. For each $i \in \{1, ..., m\}$, and for each of them to construct walks covering all 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. 7, 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:

$$\exists i \in \{1, ..., m\},\ \phi_w(X_i) = G \tag{10}$$

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. Here we do not adress the question if this problem

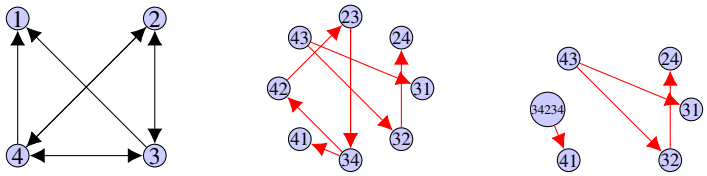
is NP-hard. Finally, if $x_1, ..., x_c$ are vertices of a strongly component of $H^{(w-2)}$, which order should be considered to form a new vertex attribute x_C? The following lemma shows that this order is not important, as long as it represents a walk in the component. Moreover, it is possible to reconstruct all admissible sequences from walks on $R(H^{w-2})$. With the same notations:

Lemma 1. *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 $R(H^{(w-2)})$.*

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. □



(a) (b) (c)

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

4.2 Dynamic programming formulation

In the previous subsection, we characterized w-sequence graphs but did not present a way to count admissible sequences in the general case. Although the tractability of our problems (NP-hardness of REALIZABLE, #P-hardness of NUMREALIZATIONS) currently remains open, we present in this subsection a method based on dynamic programming.

The recursion proceeds by extending a partial sequence, initially set to be empty, keeping track of for represented edges along the way. Namely, consider $N_w[\Pi, p, u_{1:w}]$ 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 subword $u_1, ..., u_{w-1}$. Then, with the notations of Section 4, we have $\forall (u_1, u_2, ..., u_{w-1}) \in V^{w-1}$, $\forall p \geq 0$:

$$N_w[\Pi, p, (u_1, ..., u_{w-1})] = \sum_{u_w \in V} N_w[\Pi', p-1, (u_2, ..., u_w)] \tag{11}$$

with

$$\Pi' := (\pi_{ij} - |\{k \in [1, w_1] \mid (u_k, u_w) = (i, j)\}|)_{(i,j)\in V^2}. \tag{12}$$

The base case of this recurrence corresponds to $p = 0$, and is defined as

$$\forall\ \Pi,\ N_w[\Pi, 0, u_{1:w-1}] = \begin{cases} 1 & \text{if } \Pi = (0)_{(i,j)\in V^2} \\ 0 & \text{otherwise.} \end{cases} \tag{13}$$

Finally, this algorithm only computes extensions of a prefix of length w, so one must marginalize on all possible prefixes to obtain the number $N_w[\Pi, p]$ of realizations:

$$N_w[\Pi, p] = \sum_{u_1,...,u_{w-1}\in V^{w-1}} N_w[\Pi', p-(w-1), (u_1, ..., u_{w-1})] \tag{14}$$

where Π' is the update of Π, computed similarly as in Equation (12).

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

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

and it follows that $\prod_{i,j\in V^2} \pi_{i,j} \in \mathcal{O}(2^{w\,p})$. 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.

5 Application to sequential models

5.1 Number of equivalent sequences for weigthed 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 $w \in 3 \rightarrow 10$, 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. It should be noted that N_w is a lower bound of the number of total admissible sequences, since a starting pattern is required (first w tokens).

Results are reported in the left plot of Fig. 5. For $w = 2$, the number of sequences (obtained using Prop. 6) 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 $w = 10$ [19,25]. 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 [13]. These models have attracted new interest due to experimental progress for time series prediction [11] and natural language processing [3,7,29]. They do not share the bag-of-words assumption by construction. 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 1. 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 [8], 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. 5 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.

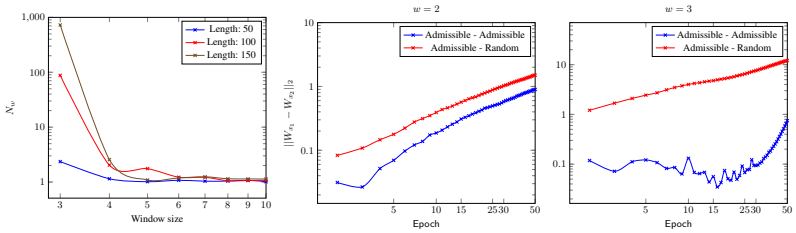


Fig. 5: Left plot: Average lower bound (N_w) on the average number of sequences. Two right plots: $||W_{x_1} - W_{x_2}||_2 = f(\text{Epoch})$ - LSTM, $\log - \log$ scale

6 Conclusion

In this preliminary study, we presented some theoretical results and practical algorithms for the family of sequence graphs. Graphs having this structure have been partially studied in the Distance Geometry (DG) literature before, mostly to do with proteins, where

an "atom window" can be defined by using the protein backbone [14], by determining symmetry structure and the complexity of finding orders to the case where every element in a sequence is assigned an individual node in the graph (an element repeating in a sequence is attributed different vertices), which is not the model we are investigating in this paper.

We applied these results to experiments for sequential models, with a focus on natural language modeling. This study can be of use used for several sequential models, such as continuous bag of words (CBOW), skip-grams ([16,10,28]), pointwise mutual information models [19] and generative probabilistic models [2,25]. These experiments suggests that ambiguity induced by graphical representations are not present with recurrent neural networks, suggesting a semantic difference for the initial considered sequences.

The hardness of the considered problems remains open for some instances, and we plan to address their complexity in future work, for instance by exploiting the structural properties of sequence graphs (*e.g.* existence of forbidden patterns).

References

1. Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., et al.: Tensorflow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow. org **1**(2) (2015)
2. Arora, S., Li, Y., Liang, Y., Ma, T., Risteski, A.: A latent variable model approach to pmi-based word embeddings. Transactions of the Association for Computational Linguistics **4**, 385–399 (2016)
3. Bartunov, S., Kondrashkin, D., Osokin, A., Vetrov, D.: Breaking sticks and ambiguities with adaptive skip-gram. In: artificial intelligence and statistics. pp. 130–138 (2016)
4. Brightwell, G.R., Winkler, P.: Counting eulerian circuits is# p-complete. In: ALENEX/ANALCO. pp. 259–262. Citeseer (2005)
5. de Bruijn, N.G., van Aardenne-Ehrenfest, T.: Circuits and trees in oriented linear graphs. Simon Stevin **28**, 203–217 (1951)
6. Chaiken, S.: A combinatorial proof of the all minors matrix tree theorem. SIAM Journal on Algebraic Discrete Methods **3**(3), 319–329 (1982)
7. Chen, Q., Zhu, X., Ling, Z., Wei, S., Jiang, H., Inkpen, D.: Enhanced lstm for natural language inference. arXiv preprint arXiv:1609.06038 (2016)
8. Chollet, F., et al.: Keras. https://keras.io (2015)
9. Gibert, J., Valveny, E., Bunke, H.: Dimensionality reduction for graph of words embedding. In: International Workshop on Graph-Based Representations in Pattern Recognition. pp. 22–31. Springer (2011)
10. Goldberg, Y., Levy, O.: word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method. arXiv preprint arXiv:1402.3722 (2014)
11. Greff, K., Srivastava, R.K., Koutník, J., Steunebrink, B.R., Schmidhuber, J.: Lstm: A search space odyssey. IEEE transactions on neural networks and learning systems **28**(10), 2222–2232 (2016)
12. Hatcher, E., Gospodnetic, O.: Lucene in action. Manning Publications (2004)
13. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8), 1735–1780 (1997)
14. Liberti, L., Lavor, C., Maculan, N., Mucherino, A.: Euclidean distance geometry and applications. Siam Review **56**(1), 3–69 (2014)

15. Malliaros, F.D., Skianis, K.: Graph-based term weighting for text categorization. In: Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015. pp. 1473–1479 (2015)
16. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)
17. Nadeau, D., Sekine, S.: A survey of named entity recognition and classification. Lingvisticae Investigationes **30**(1), 3–26 (2007)
18. Peng, H., Li, J., He, Y., Liu, Y., Bao, M., Wang, L., Song, Y., Yang, Q.: Large-scale hierarchical text classification with recursively regularized deep graph-cnn. In: Proceedings of the 2018 World Wide Web Conference. pp. 1063–1072 (2018)
19. Pennington, J., Socher, R., Manning, C.: Glove: Global vectors for word representation. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). pp. 1532–1543 (2014)
20. Ramos, J., et al.: Using tf-idf to determine word relevance in document queries. In: Proceedings of the first instructional conference on machine learning. vol. 242, pp. 133–142. Piscataway, NJ (2003)
21. Roth, M., Woodsend, K.: Composition of word representations improves semantic role labelling. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). pp. 407–413 (2014)
22. Rousseau, F., Kiagias, E., Vazirgiannis, M.: 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). pp. 1702–1712 (2015)
23. Salton, G.: The smart system. Retrieval Results and Future Plans (1971)
24. Salton, G., Wong, A., Yang, C.S.: A vector space model for automatic indexing. Communications of the ACM **18**(11), 613–620 (1975)
25. Sanjeev, A., Yingyu, L., Tengyu, M.: A simple but tough-to-beat baseline for sentence embeddings. Proceedings of ICLR (2017)
26. Sharir, M.: A strong-connectivity algorithm and its applications in data flow analysis. Computers & Mathematics with Applications **7**(1), 67–72 (1981)
27. Skianis, K., Malliaros, F., Vazirgiannis, M.: 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). pp. 49–58 (2018)
28. Song, Y., Shi, S., Li, J., Zhang, H.: 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). pp. 175–180 (2018)
29. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: Advances in neural information processing systems. pp. 5998–6008 (2017)

A Proofs section 3 ($w = 2$)

Proof (Prop. 1). 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. This can be proved similarly to (i) $\implies$ (ii) for proposition 2 by replacing connectivity with strong connectivity. □

Proof (Prop. 3). This can be proved similarly to (i) $\implies$ (ii) for proposition 2 by replacing connectivity with strong connectivity. If G is Eulerian or semi-Eulerian, there exists a walk going through all edges, this walk defines a 2-admissible sequence. □

Proof (Prop. 4). 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. □

Proof (Prop. 5). 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. 4, 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. □

Proof (Th. 1). If G is a 2-sequence graph, $R(G)$ is a 2-sequence graph using Prop. 5. Also Prop. 4 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. 2, 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. 2).

• 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. □

Proof (Corollary 1). 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 1. 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. 4, y is the unique 2-admissible sequence.

Proof (Prop. 6). Direct consequence of BEST Theorem [5]) and Matrix tree theorem [6]).

B Proofs section 4 ($w \geq 3$)

Proof (Prop. 7). 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 (Lemma 1). 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. 2). 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. □