Sequence Graphs: Characterization and Couting of Admissible Elements

Sammy Khalife
LIX, Ecole Polytechnique

Cologne-Twente Workshop
on Graphs and Combinatorial Optimization

13th September, 2020

Sequence Graphs: Introduction

- Representations of words and textual documents are essential for several tasks (document classification, translation, role labelling, named entity recognition, ...)

Sequence Graphs: Introduction

- Representations of words and textual documents are essential for several tasks (document classification, translation, role labelling, named entity recognition, ...)
- Bag-Of-Words (BOW) representations:

Sequence Graphs: Introduction

- Representations of words and textual documents are essential for several tasks (document classification, translation, role labelling, named entity recognition, ...)
- Bag-Of-Words (BOW) representations:
  - Document similarity can be efficiently computed using sparse linear algebra.

Sequence Graphs: Introduction

- Representations of words and textual documents are essential for several tasks (document classification, translation, role labelling, named entity recognition, ...)
- Bag-Of-Words (BOW) representations:
  - Document similarity can be efficiently computed using sparse linear algebra.
  - The degree of ambiguity grows factorially on the size of the document.

Sequence Graphs: Introduction

- Representations of words and textual documents are essential for several tasks (document classification, translation, role labelling, named entity recognition, ...)
- Bag-Of-Words (BOW) representations:
  - Document similarity can be efficiently computed using sparse linear algebra.
  - The degree of ambiguity grows factorially on the size of the document.
- This high degree of *ambiguity* has practical consequences for downstream tasks

Sequence Graphs: Introduction

- Representations of words and textual documents are essential for several tasks (document classification, translation, role labelling, named entity recognition, ...)
- Bag-Of-Words (BOW) representations:
  - Document similarity can be efficiently computed using sparse linear algebra.
  - The degree of ambiguity grows factorially on the size of the document.
- This high degree of *ambiguity* has practical consequences for downstream tasks
- Co-occurence models (e.g. Graph of words (GOW)) supplement the content of BOW with statistics of co-occurrences, mitigating the degree of ambiguity.

Sequence Graphs: Introduction

- Representations of words and textual documents are essential for several tasks (document classification, translation, role labelling, named entity recognition, ...)
- Bag-Of-Words (BOW) representations:
  - Document similarity can be efficiently computed using sparse linear algebra.
  - The degree of ambiguity grows factorially on the size of the document.
- This high degree of *ambiguity* has practical consequences for downstream tasks
- Co-occurence models (e.g. Graph of words (GOW)) supplement the content of BOW with statistics of co-occurrences, mitigating the degree of ambiguity.

Linux is not UNIX but Linux

Figure: Sequence graph (or Graph of Words) with $w = 3$

Sequence Graphs: Introduction

- Representations of words and textual documents are essential for several tasks (document classification, translation, role labelling, named entity recognition, ...)
- Bag-Of-Words (BOW) representations:
  - Document similarity can be efficiently computed using sparse linear algebra.
  - The degree of ambiguity grows factorially on the size of the document.
- This high degree of *ambiguity* has practical consequences for downstream tasks
- Co-occurence models (e.g. Graph of words (GOW)) supplement the content of BOW with statistics of co-occurrences, mitigating the degree of ambiguity.

Linux is not UNIX but Linux

Figure: Sequence graph (or Graph of Words) with $w = 3$

Sequence Graphs: Introduction

- Representations of words and textual documents are essential for several tasks (document classification, translation, role labelling, named entity recognition, ...)
- Bag-Of-Words (BOW) representations:
  - Document similarity can be efficiently computed using sparse linear algebra.
  - The degree of ambiguity grows factorially on the size of the document.
- This high degree of *ambiguity* has practical consequences for downstream tasks
- Co-occurence models (e.g. Graph of words (GOW)) supplement the content of BOW with statistics of co-occurrences, mitigating the degree of ambiguity.

Linux is not UNIX but Linux

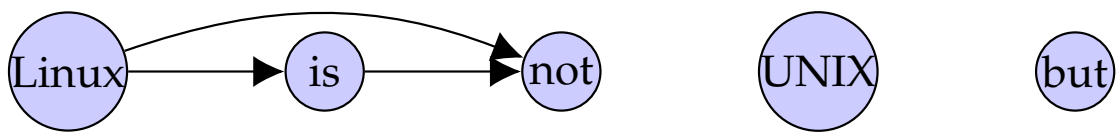


Figure: Sequence graph (or Graph of Words) with $w = 3$

Sequence Graphs: Introduction

- Representations of words and textual documents are essential for several tasks (document classification, translation, role labelling, named entity recognition, ...)
- Bag-Of-Words (BOW) representations:
  - Document similarity can be efficiently computed using sparse linear algebra.
  - The degree of ambiguity grows factorially on the size of the document.
- This high degree of *ambiguity* has practical consequences for downstream tasks
- Co-occurence models (e.g. Graph of words (GOW)) supplement the content of BOW with statistics of co-occurrences, mitigating the degree of ambiguity.

Linux is not UNIX but Linux

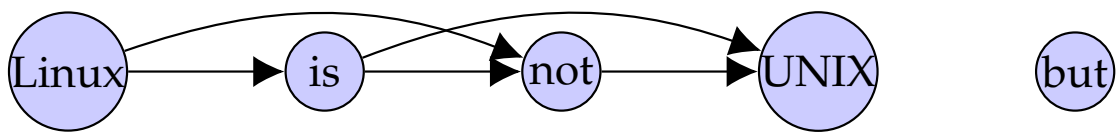


Figure: Sequence graph (or Graph of Words) with $w = 3$

Sequence Graphs: Introduction

- Representations of words and textual documents are essential for several tasks (document classification, translation, role labelling, named entity recognition, ...)
- Bag-Of-Words (BOW) representations:
  - Document similarity can be efficiently computed using sparse linear algebra.
  - The degree of ambiguity grows factorially on the size of the document.
- This high degree of *ambiguity* has practical consequences for downstream tasks
- Co-occurence models (e.g. Graph of words (GOW)) supplement the content of BOW with statistics of co-occurrences, mitigating the degree of ambiguity.

Linux is not UNIX but Linux

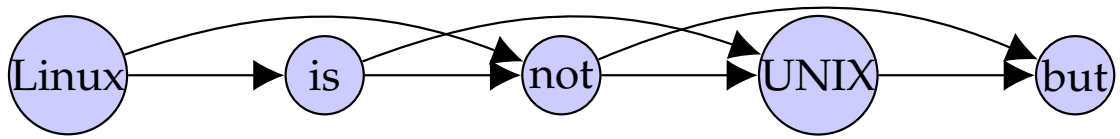


Figure: Sequence graph (or Graph of Words) with $w = 3$

Sequence Graphs: Introduction

- Representations of words and textual documents are essential for several tasks (document classification, translation, role labelling, named entity recognition, ...)
- Bag-Of-Words (BOW) representations:
  - Document similarity can be efficiently computed using sparse linear algebra.
  - The degree of ambiguity grows factorially on the size of the document.
- This high degree of *ambiguity* has practical consequences for downstream tasks
- Co-occurence models (e.g. Graph of words (GOW)) supplement the content of BOW with statistics of co-occurrences, mitigating the degree of ambiguity.

Linux is not UNIX but Linux

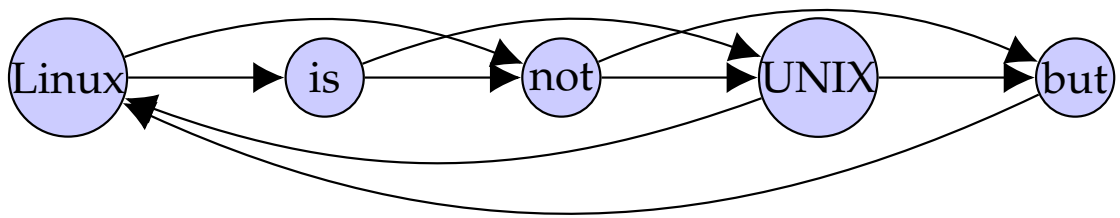


Figure: Sequence graph (or Graph of Words) with $w = 3$

Sequence Graphs: Introduction

- Representations of words and textual documents are essential for several tasks (document classification, translation, role labelling, named entity recognition, ...)
- Bag-Of-Words (BOW) representations:
  - Document similarity can be efficiently computed using sparse linear algebra.
  - The degree of ambiguity grows factorially on the size of the document.
- This high degree of *ambiguity* has practical consequences for downstream tasks
- Co-occurence models (e.g. Graph of words (GOW)) supplement the content of BOW with statistics of co-occurrences, mitigating the degree of ambiguity.

Linux is not UNIX but Linux

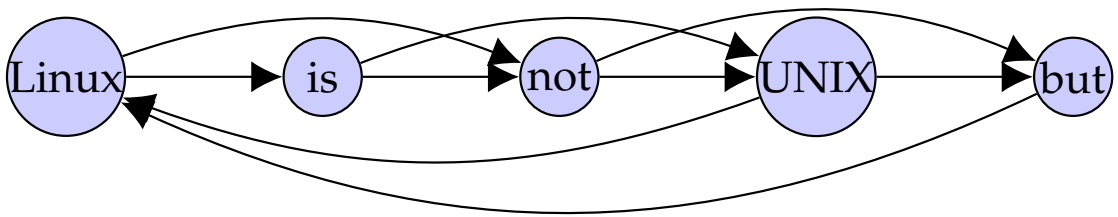
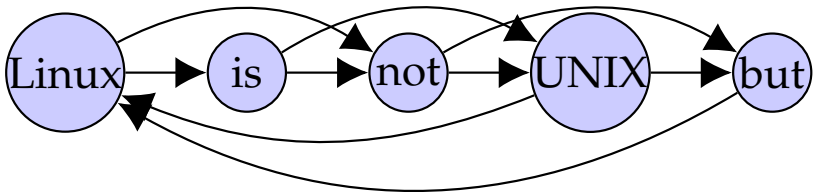


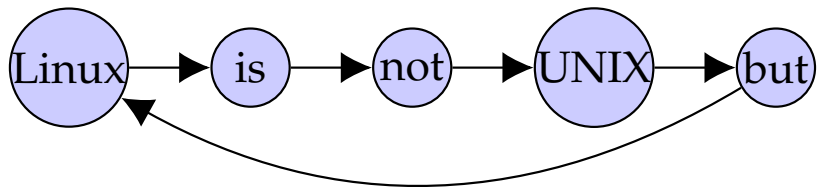
Figure: Sequence graph (or Graph of Words) with $w = 3$

Sequence Graphs: Introduction

- Such models still induce ambiguity due to the fact that several sequences can *a priori* create the same graph.



(a) No ambiguity ($w = 3$)



(b) Ambiguity ($w = 2$)

Figure: In case (b), the sequence graph is ambiguous: any circular permutation of the words admits the same representation.

- Popular models (word2vec, Glove, fastText, Latent random walks) use co-occurence statistics
- What is their level of ambiguity?
- Difference with other Language models (e.g. LSTMs)?

Definition of a Sequence Graph

Let $x = x_1, \cdots, x_p$ be a sequence of discrete elements of a vocabulary $\{1, ..., n\}$.

Definition 1.1

Definition of a Sequence Graph

Let $x = x_1, \cdots, x_p$ be a sequence of discrete elements of a vocabulary $\{1, ..., n\}$.

Definition 1.1

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

Definition of a Sequence Graph

Let $x = x_1, \cdots, x_p$ be a sequence of discrete elements of a vocabulary $\{1, ..., n\}$.

Definition 1.1

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

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

Definition of a Sequence Graph

Let $x = x_1, \cdots, x_p$ be a sequence of discrete elements of a vocabulary $\{1, ..., n\}$.

Definition 1.1

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

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

$$(i, j) \in E \iff i \text{ and } j \text{ “appears closer” than } w \text{ in } x$$

Definition of a Sequence Graph

Let $x = x_1, \cdots, x_p$ be a sequence of discrete elements of a vocabulary $\{1, ..., n\}$.

Definition 1.1

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

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

$$(i, j) \in E \iff i \text{ and } j \text{ “appears closer” than } w \text{ in } x$$

$$x \text{ is a } w\text{-realization of } G$$

Sequence Graphs

- A sequence graph can be computed in linear time.

Sequence Graphs

- A sequence graph can be computed in linear time.
- Definition can be generalized to digraphs and weighted graphs

Sequence Graphs

- A sequence graph can be computed in linear time.
- Definition can be generalized to digraphs and weighted graphs
- Correspondence between the monoid $X^\star$ into the graph set $\mathcal{G}$:

$$\phi_w : X^\star \to \mathcal{G}, x \mapsto G_w(x)$$

Sequence Graphs

- A sequence graph can be computed in linear time.
- Definition can be generalized to digraphs and weighted graphs
- Correspondence between the monoid $X^\star$ into the graph set $\mathcal{G}$:

$$\phi_w : X^\star \to \mathcal{G}, x \mapsto G_w(x)$$

- The degree of ambiguity of a representation is given by:

$$\mathsf{Card}\,\mathsf{Im}\,\phi_w^{-1}(G)$$

When G is a given graph.

Sequence Graphs: REALIZABLE$_w$ and NUMREALIZATIONS$_w$

Problem 1 (REALIZABLE$_w$)

Parameters: *Window size w*
Input: *Graph G (and optional matrix weights Π)*
Output: *True if (G, Π) is the w-sequence graph of some sequence x, False otherwise.*

Problem 2 (NUMREALIZATIONS$_w$)

Parameters: *Window size w*
Input: *Graph G (and optional matrix weights Π)*
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.*

2-sequence Graphs: $w = 2$

First results: The case $w = 2$ leads to simple characterization and algorithmic treatment:

2-sequence Graphs: $w = 2$

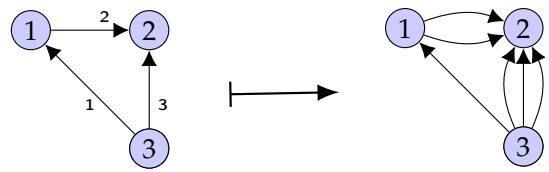
First results: The case $w = 2$ leads to simple characterization and algorithmic treatment:

- Weighted contraction $G \mapsto R^+(G)$ in a DAG of its strongly connected components.

2-sequence Graphs: $w = 2$

First results: The case $w = 2$ leads to simple characterization and algorithmic treatment:

- Weighted contraction $G \mapsto R^+(G)$ in a DAG of its strongly connected components.
- Eulerian reductions.

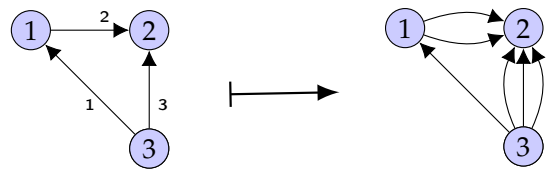


G $\psi(G)$

2-sequence Graphs: $w = 2$

First results: The case $w = 2$ leads to simple characterization and algorithmic treatment:

- Weighted contraction $G \mapsto R^+(G)$ in a DAG of its strongly connected components.
- Eulerian reductions.



G $\qquad$ $\psi(G)$

Table: Complexity for various instances ($w = 2$)

	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	$R^+(G)$ straight line with unit weights
Weighted digraph	P	$\mathbb{N}$ (BEST Theorem)	P	$\psi(G)$ (semi)Eulerian

General Case: $w \geq 3$

No trivial reduction. We propose 3 different approaches:

- a) A direct recursive formulation for REALIZABLE$_w$
  - with polynomial time complexity in the unweighted and undirected case

General Case: $w \geq 3$

No trivial reduction. We propose 3 different approaches:

- a) A direct recursive formulation for REALIZABLE$_w$
  - with polynomial time complexity in the unweighted and undirected case
- b) A linear integer programming formulation for REALIZABLE$_w$
  - Suffers from a large number of constraints.

General Case: $w \geq 3$

No trivial reduction. We propose 3 different approaches:

- a) A direct recursive formulation for REALIZABLE_w
  - with polynomial time complexity in the unweighted and undirected case
- b) A linear integer programming formulation for REALIZABLE_w
  - Suffers from a large number of constraints.
- c) A dynamic programming formulation for NUMREALIZATIONS_w
  - with exponential time average complexity but much better than worst-case scenario.

General Case: Direct Formulation

- If $G = (V, E)$ then $H(G) = (E, H_E)$ such that

General Case: Direct Formulation

- If $G = (V, E)$ then $H(G) = (E, H_E)$ such that

$$\forall e = (v_1, v_2) \in E \quad \text{and} \quad \forall f = (v_3, v_4) \in E$$

General Case: Direct Formulation

- If $G = (V, E)$ then $H(G) = (E, H_E)$ such that

$$\forall e = (v_1, v_2) \in E \quad \text{and} \quad \forall f = (v_3, v_4) \in E$$

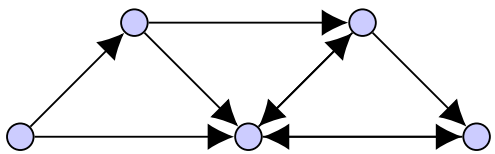
$$(e, f) \in H_E \iff v_2 = v_3 \text{ and } (v_1, v_4) \in E$$

General Case: Direct Formulation

- If $G = (V, E)$ then $H(G) = (E, H_E)$ such that

$$\forall e = (v_1, v_2) \in E \quad \text{and} \quad \forall f = (v_3, v_4) \in E$$

$$(e, f) \in H_E \iff v_2 = v_3 \text{ and } (v_1, v_4) \in E$$



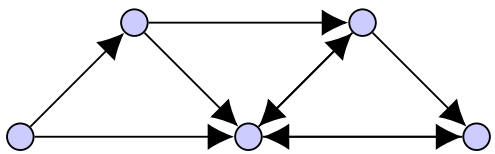
(a) G

General Case: Direct Formulation

- If $G = (V, E)$ then $H(G) = (E, H_E)$ such that

$$\forall e = (v_1, v_2) \in E \quad \text{and} \quad \forall f = (v_3, v_4) \in E$$

$$(e, f) \in H_E \iff v_2 = v_3 \text{ and } (v_1, v_4) \in E$$



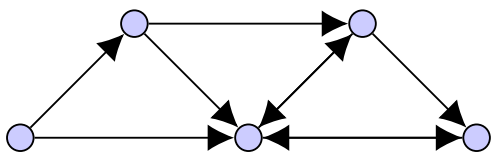
(a) G

General Case: Direct Formulation

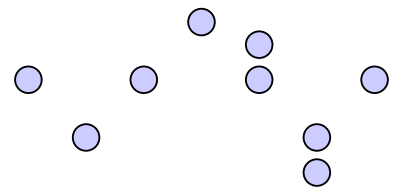
- If $G = (V, E)$ then $H(G) = (E, H_E)$ such that

$$\forall e = (v_1, v_2) \in E \quad \text{and} \quad \forall f = (v_3, v_4) \in E$$

$$(e, f) \in H_E \iff v_2 = v_3 \text{ and } (v_1, v_4) \in E$$



(a) G



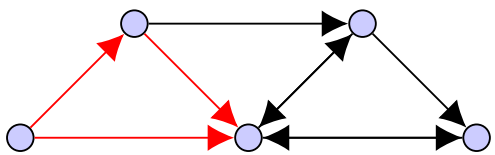
(b) $H(G)$

General Case: Direct Formulation

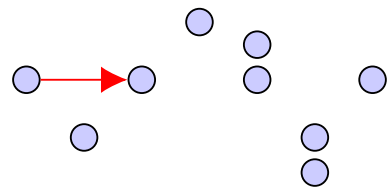
- If $G = (V, E)$ then $H(G) = (E, H_E)$ such that

$$\forall e = (v_1, v_2) \in E \quad \text{and} \quad \forall f = (v_3, v_4) \in E$$

$$(e, f) \in H_E \iff v_2 = v_3 \text{ and } (v_1, v_4) \in E$$



(a) G



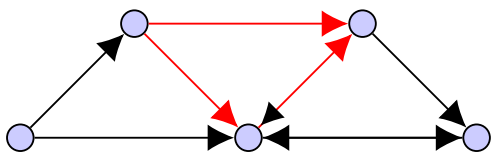
(b) $H(G)$

General Case: Direct Formulation

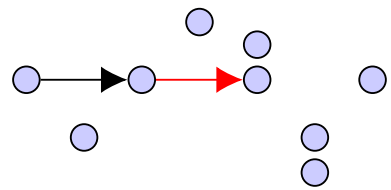
- If $G = (V, E)$ then $H(G) = (E, H_E)$ such that

$$\forall e = (v_1, v_2) \in E \quad \text{and} \quad \forall f = (v_3, v_4) \in E$$

$$(e, f) \in H_E \iff v_2 = v_3 \text{ and } (v_1, v_4) \in E$$



(a) G



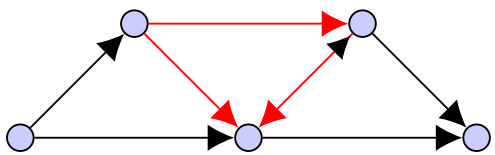
(b) $H(G)$

General Case: Direct Formulation

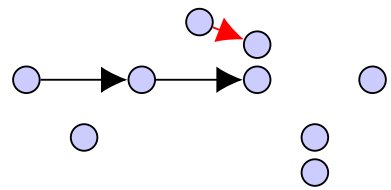
- If $G = (V, E)$ then $H(G) = (E, H_E)$ such that

$$\forall e = (v_1, v_2) \in E \quad \text{and} \quad \forall f = (v_3, v_4) \in E$$

$$(e, f) \in H_E \iff v_2 = v_3 \text{ and } (v_1, v_4) \in E$$



(a) G



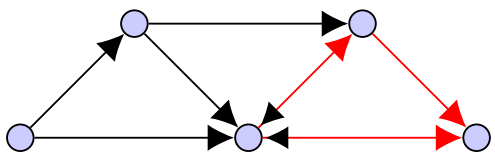
(b) $H(G)$

General Case: Direct Formulation

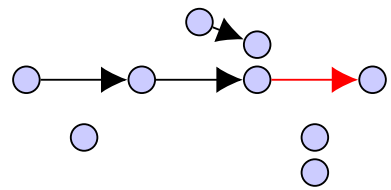
- If $G = (V, E)$ then $H(G) = (E, H_E)$ such that

$$\forall e = (v_1, v_2) \in E \quad \text{and} \quad \forall f = (v_3, v_4) \in E$$

$$(e, f) \in H_E \iff v_2 = v_3 \text{ and } (v_1, v_4) \in E$$



(a) G



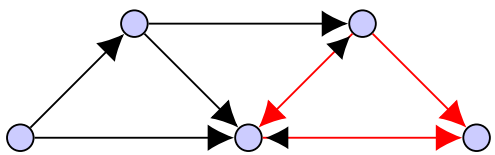
(b) $H(G)$

General Case: Direct Formulation

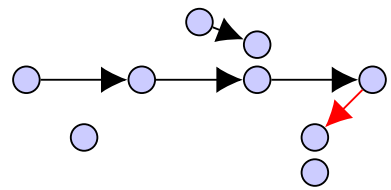
- If $G = (V, E)$ then $H(G) = (E, H_E)$ such that

$$\forall e = (v_1, v_2) \in E \quad \text{and} \quad \forall f = (v_3, v_4) \in E$$

$$(e, f) \in H_E \iff v_2 = v_3 \text{ and } (v_1, v_4) \in E$$



(a) G



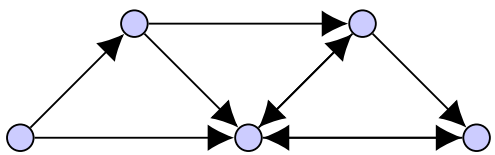
(b) $H(G)$

General Case: Direct Formulation

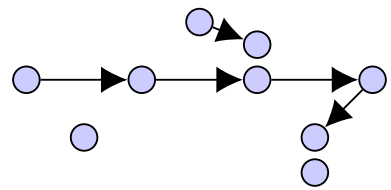
- If $G = (V, E)$ then $H(G) = (E, H_E)$ such that

$$\forall e = (v_1, v_2) \in E \quad \text{and} \quad \forall f = (v_3, v_4) \in E$$

$$(e, f) \in H_E \iff v_2 = v_3 \text{ and } (v_1, v_4) \in E$$



(a) G



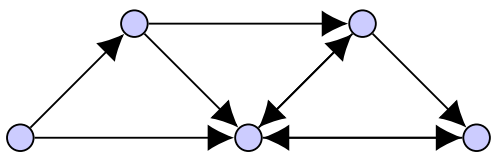
(b) $H(G)$

General Case: Direct Formulation

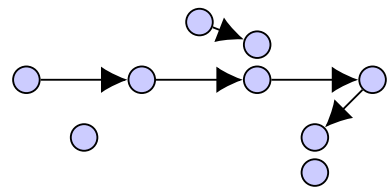
- If $G = (V, E)$ then $H(G) = (E, H_E)$ such that

$$\forall e = (v_1, v_2) \in E \quad \text{and} \quad \forall f = (v_3, v_4) \in E$$

$$(e, f) \in H_E \iff v_2 = v_3 \ \text{and} \ (v_1, v_4) \in E$$



(a) G



(b) $H(G)$

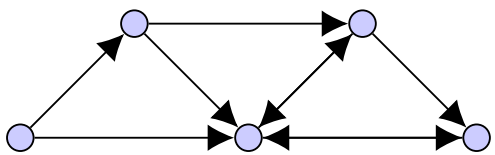
$H(G)$ is the ***adjoint graph*** of G ("line graph") for which some edges have been deleted.

General Case: Direct Formulation

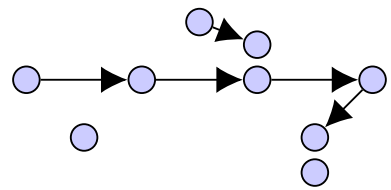
- If $G = (V, E)$ then $H(G) = (E, H_E)$ such that

$$\forall e = (v_1, v_2) \in E \quad \text{and} \quad \forall f = (v_3, v_4) \in E$$

$$(e, f) \in H_E \iff v_2 = v_3 \text{ and } (v_1, v_4) \in E$$



(a) G



(b) $H(G)$

$H(G)$ is the ***adjoint graph*** of G ("line graph") for which some edges have been deleted.

$\rightarrow$ If $w = 3$, a realization of G is a walk in $H(G)$

General Case: Direct Formulation

Idea: For $w > 3$, recursively iterate the process for $k \leq w - 2$: and define a sequence of auxiliary graphs $H^{(k)}$.

General Case: Direct Formulation

Idea: For $w > 3$, recursively iterate the process for $k \leq w - 2$: and define a sequence of auxiliary graphs $H^{(k)}$.

Proposition 3.1 (Khalife, 2020)

Let $x = x_1, \ldots, x_p$ be a w-realization of a graph $G = (V, E)$. If $w \leq p$, x, then x corresponds to a walk of length $p - w + 1$ on $H^{(w-2)}$.

General Case: Direct Formulation

Idea: For $w > 3$, recursively iterate the process for $k \leq w - 2$: and define a sequence of auxiliary graphs $H^{(k)}$.

Proposition 3.1 (Khalife, 2020)

Let $x = x_1, \ldots, x_p$ be a w-realization of a graph $G = (V, E)$. If $w \leq p$, x, then x corresponds to a walk of length $p - w + 1$ on $H^{(w-2)}$.

Theorem 3.2 (Khalife, 2020)

Let G a graph and $w \in \mathbb{N}^* - \{1, 2\}$. If G is undirected then $\textsc{Realizable}_w$ *is in P.*

General Case: Direct Formulation

Idea: For $w > 3$, recursively iterate the process for $k \leq w - 2$: and define a sequence of auxiliary graphs $H^{(k)}$.

Proposition 3.1 (Khalife, 2020)

Let $x = x_1, \ldots, x_p$ be a w-realization of a graph $G = (V, E)$. If $w \leq p$, x, then x corresponds to a walk of length $p - w + 1$ on $H^{(w-2)}$.

Theorem 3.2 (Khalife, 2020)

Let G a graph and $w \in \mathbb{N}^* - \{1, 2\}$. If G is undirected then $\textsc{Realizable}_w$ *is in P.*

Proof. Extract connected components of $H^{(w-2)}$. For each of them, there exists a walk covering all vertices at least once. A potential realization cannot generate more edges than these walks. Use the definition to compute the realizations from the walks and compare.

General Case: Direct Formulation and Digraphs

- Digraphs: the analogue procedure enumerates all paths in the DAG $R(H^{(w-2)})$, where the operator R is the contraction in strongly connected components.

General Case: Direct Formulation and Digraphs

- Digraphs: the analogue procedure enumerates all paths in the DAG $R(H^{(w-2)})$, where the operator R is the contraction in strongly connected components.
- The number of paths can be exponential in the auxiliary graph $R(H^{(w-2)})$ obtained from a sequence graph.

General Case: Direct Formulation and Digraphs

- Digraphs: the analogue procedure enumerates all paths in the DAG $R(H^{(w-2)})$, where the operator R is the contraction in strongly connected components.
- The number of paths can be exponential in the auxiliary graph $R(H^{(w-2)})$ obtained from a sequence graph.

Example

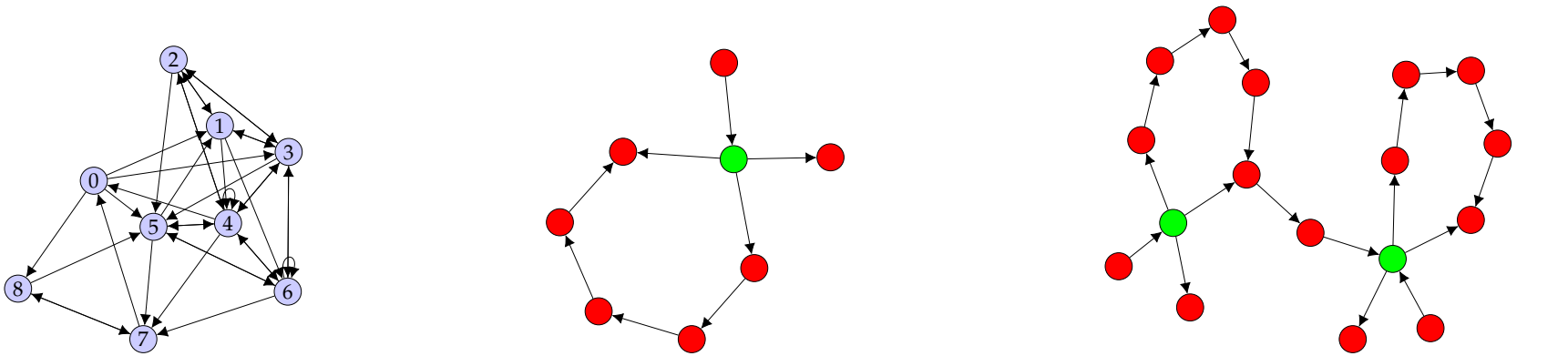
$$x_{\text{init}} = 0, 3, 1, 2, 3, 4, 4, 2, 5, 1, 4, 6, 3, 5, 6, 6, 4, 7, 0, 8, 5, 7$$

For $P \in \mathbb{N}^*$, let $x \in \mathbb{N}^{22P}$:

$$\forall i \in \{0, \ldots, 22P - 1\} \qquad x_i = x_{\text{init}}[i] + 10 \left\lfloor \frac{i}{22} \right\rfloor$$

Claim: With $w = 3$, $R(H(x))$ has at least 2^P paths.

General Case: Direct Formulation and Digraphs



(a) $G = \phi_3(x_{\text{init}})$ (b) $R(H(x)) \quad (P = 1)$ (c) $R(H(x)) \quad (P = 2)$

Figure: Direct approach has exponential complexity in the worst case

- For digraphs and weighted graphs, the complexity classes characterizations of NUMREALIZATIONS$_w$ and REALIZABLE$_w$ remain opened ($w \geq 3$).

General Case: Integer Programming Formulation

Khalife S. & Ponty Y. (2020). Sequence graphs realizations and ambiguity in language models (hal-02495333, v2)

REALIZABLE$_w$ can be formulated as an *Linear integer program*.

Decision variable $X \in \mathbb{M}_{n,p}(\{0,1\})$ where $X_{i,j} = 1 \iff x_j = i$

General Case: Integer Programming Formulation

Khalife S. & Ponty Y. (2020). Sequence graphs realizations and ambiguity in language models (hal-02495333, v2)

REALIZABLE$_w$ can be formulated as an *Linear integer program*.

Decision variable $X \in \mathbb{M}_{n,p}(\{0,1\})$ where $X_{i,j} = 1 \iff x_j = i$

- $n^2(w-1)(p - \frac{w-2}{2})$ constraints

  $\rightarrow w = 3$ (window size), $n = 20$ (vocabulary size), $p = 10$ (sequence length): 7600 constraints. Not solvable in practice with Branch and Bound.

General Case: Integer Programming Formulation

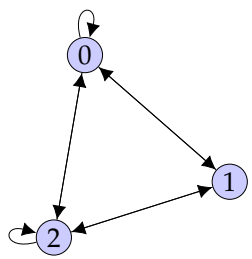
Khalife S. & Ponty Y. (2020). Sequence graphs realizations and ambiguity in language models (hal-02495333, v2)

REALIZABLE$_w$ can be formulated as an *Linear integer program*.

Decision variable $X \in \mathbb{M}_{n,p}(\{0,1\})$ where $X_{i,j} = 1 \iff x_j = i$

- $n^2(w-1)(p - \frac{w-2}{2})$ constraints

  $\rightarrow w = 3$ (window size), $n = 20$ (vocabulary size), $p = 10$ (sequence length): 7600 constraints. Not solvable in practice with Branch and Bound.



(a) Graph G

$$\begin{pmatrix} 2 & 3 & 1 \\ 2 & 0 & 2 \\ 2 & 3 & 2 \end{pmatrix}$$

(b) Matrix Π

2 2 1 2 0 0 1 0 2 1

(c) A solution

Figure: Example of instance solved using the Linear Integer Programming formulation ($n = 3, p = 10, w = 3$).

General Case: Dynamic Programming Formulation

Khalife S. & Ponty Y. (2020). Sequence graphs realizations and ambiguity in language models (hal-02495333, v2)

$N_w\,[\Pi, p, \mathbf{u}]$ number of realizations:

- of the graph with weights matrix Π
- of length p with starting pattern $\mathbf{u}$

General Case: Dynamic Programming Formulation

Khalife S. & Ponty Y. (2020). Sequence graphs realizations and ambiguity in language models (hal-02495333, v2)

$N_w\,[\Pi, p, \mathbf{u}]$ number of realizations:

- of the graph with weights matrix Π
- of length p with starting pattern $\mathbf{u}$

Recursion

Idea: Decompose the problem recursively with respect to p by summation.

General Case: Dynamic Programming Formulation

Khalife S. & Ponty Y. (2020). Sequence graphs realizations and ambiguity in language models (hal-02495333, v2)

$N_w\,[\Pi, p, \mathbf{u}]$ number of realizations:

- of the graph with weights matrix Π
- of length p with starting pattern $\mathbf{u}$

Recursion

Idea: Decompose the problem recursively with respect to p by summation.

General Case: Dynamic Programming Formulation

Khalife S. & Ponty Y. (2020). Sequence graphs realizations and ambiguity in language models (hal-02495333, v2)

$N_w\,[\Pi, p, \mathbf{u}]$ number of realizations:

- of the graph with weights matrix Π
- of length p with starting pattern $\mathbf{u}$

Recursion

Idea: Decompose the problem recursively with respect to p by summation.

$$N_w\,[\Pi, p, \mathbf{u}] = \sum_{v \in V} \Phi\left(\Pi'_{(\mathbf{u},v)}, p-1, \mathbf{u}\right)$$

General Case: Dynamic Programming Formulation

Khalife S. & Ponty Y. (2020). Sequence graphs realizations and ambiguity in language models (hal-02495333, v2)

$N_w\left[\Pi, p, \mathbf{u}\right]$ number of realizations:

- of the graph with weights matrix Π
- of length p with starting pattern $\mathbf{u}$

Recursion

Idea: Decompose the problem recursively with respect to p by summation.

$$N_w\left[\Pi, p, \mathbf{u}\right] = \sum_{v \in V} \Phi\left(\Pi'_{(\mathbf{u},v)}, p-1, \mathbf{u}\right)$$

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

General Case: Dynamic Programming Formulation

Khalife S. & Ponty Y. (2020). Sequence graphs realizations and ambiguity in language models (hal-02495333, v2)

$N_w\,[\Pi, p, \mathbf{u}]$ number of realizations:

- of the graph with weights matrix Π
- of length p with starting pattern $\mathbf{u}$

Recursion

Idea: Decompose the problem recursively with respect to p by summation.

$$N_w\,[\Pi, p, \mathbf{u}] = \sum_{v \in V} \Phi\,(\,\Pi'_{(\mathbf{u},v)}, p-1, \mathbf{u}\,)$$

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

Π' is an update of Π:

$$\Pi'_{(\mathbf{u},v)} := \left(\,\pi_{ij} - \#\left\{\,k \in \{1,\ldots,|\mathbf{u}|\} \mid (u_k, v) = (i,j)\,\right\}\,\right)_{(i,j)\in V^2}$$

General Case: Dynamic Programming Formulation

Recursion

$$N_w\left[\Pi, p, \mathbf{u}\right] = \sum_{v \in V} \Phi\left(\Pi'_{(\mathbf{u},v)}, p-1, \mathbf{u}\right)$$

General Case: Dynamic Programming Formulation

Recursion

$$N_w\left[\Pi, p, \mathbf{u}\right] = \sum_{v \in V} \Phi\left(\Pi'_{(\mathbf{u},v)}, p-1, \mathbf{u}\right)$$

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

Π' is an update of Π:

$$\Pi'_{(\mathbf{u},v)} := \left(\pi_{ij} - \#\left\{k \in \{1, ..., |\mathbf{u}|\} \mid (u_k, v) = (i,j)\right\}\right)_{(i,j) \in V^2}$$

General Case: Dynamic Programming Formulation

Recursion

$$N_w\,[\Pi, p, \mathbf{u}] = \sum_{v \in V} \Phi\,(\,\Pi'_{(\mathbf{u},v)}, p-1, \mathbf{u}\,)$$

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

Π' is an update of Π:

$$\Pi'_{(\mathbf{u},v)} := (\,\pi_{ij} - \#\{\,k \in \{\,1, ..., |\mathbf{u}|\} \mid (u_k, v) = (i,j)\,\}\,)_{(i,j)\in V^2}$$

Initialisation

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

The total number of admissible sequences is $N_w[\Pi, p, \varepsilon]$, ε being the empty prefix.

Dynamic Programming Formulation: Complexity Study

Proposition 4.1 (*Khalife, 2020*)

Comparison of the worst case complexity $\mathcal{W}_{\mathcal{C}}$ with Average complexity $\mathcal{A}_{\mathcal{C}}$:

$$\frac{2^{w\,p}}{\mathcal{U}(w\,p, \alpha^2 p^2)} \leq \frac{\mathcal{W}_{\mathcal{C}}}{\mathcal{A}_{\mathcal{C}}}$$

$$\text{where } \mathcal{U}(m, r) = \frac{1}{\binom{m+r-1}{r-1}} \sum_{m_z=1}^{\min(r,m)} \binom{r}{m_z} \binom{m-1}{m_z-1} \left(\frac{m}{m_z} + 1\right)^{m_z}$$

Dynamic Programming Formulation: Complexity Study

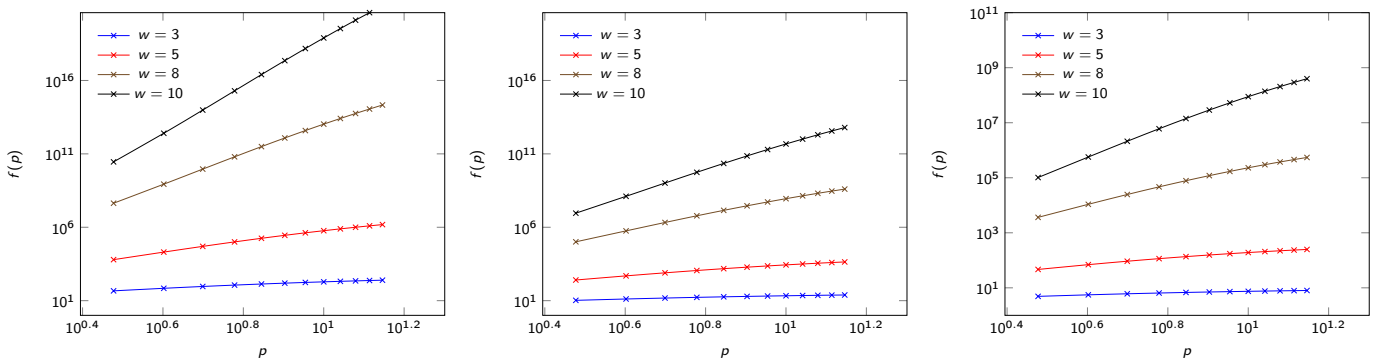
Proposition 4.1 (*Khalife, 2020*)

Comparison of the worst case complexity $\mathcal{W}_{\mathcal{C}}$ with Average complexity $\mathcal{A}_{\mathcal{C}}$:

$$\frac{2^{w\,p}}{\mathcal{U}(w\,p, \alpha^2 p^2)} \leq \frac{\mathcal{W}_{\mathcal{C}}}{\mathcal{A}_{\mathcal{C}}}$$

$$\text{where } \mathcal{U}(m, r) = \frac{1}{\binom{m+r-1}{r-1}} \sum_{m_z=1}^{\min(r,m)} \binom{r}{m_z} \binom{m-1}{m_z-1} \left(\frac{m}{m_z} + 1\right)^{m_z}$$

Figure: Lower bound of $\frac{\mathcal{W}_{\mathcal{C}}}{\mathcal{A}_{\mathcal{C}}}$ w.r.t. w and α (fraction of distinct words) in log $-$ log scale.



(a) $\alpha = 0.5$ (b) $\alpha = 0.66$ (c) $\alpha = 0.83$

Dynamic Programming Formulation: Complexity Study

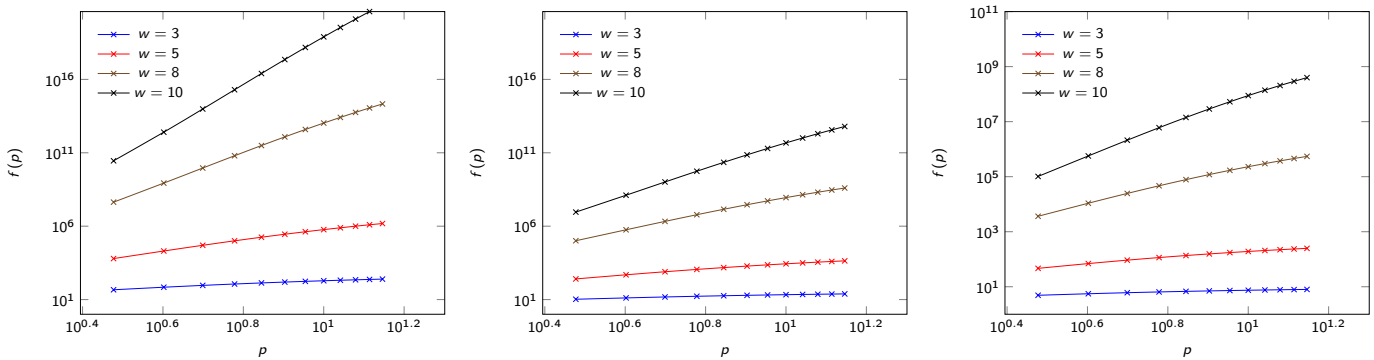
Proposition 4.1 (*Khalife, 2020*)

Comparison of the worst case complexity $\mathcal{W}_{\mathcal{C}}$ with Average complexity $\mathcal{A}_{\mathcal{C}}$:

$$\frac{2^{w\,p}}{\mathcal{U}(w\,p, \alpha^2 p^2)} \leq \frac{\mathcal{W}_{\mathcal{C}}}{\mathcal{A}_{\mathcal{C}}}$$

$$\text{where } \mathcal{U}(m, r) = \frac{1}{\binom{m+r-1}{r-1}} \sum_{m_z=1}^{\min(r,m)} \binom{r}{m_z} \binom{m-1}{m_z-1} \left(\frac{m}{m_z} + 1\right)^{m_z}$$

Figure: Lower bound of $\frac{\mathcal{W}_{\mathcal{C}}}{\mathcal{A}_{\mathcal{C}}}$ w.r.t. w and α (fraction of distinct words) in $\log - \log$ scale.



(a) $\alpha = 0.5$ (b) $\alpha = 0.66$ (c) $\alpha = 0.83$

- Good average complexity compared to worst-case
- Works on medium-size instances ($p \leq 500$)

Relation with Language Models: *Experiment 1*

Experiment 1: Ambiguity w.r.t window size

Relation with Language Models: *Experiment 1*

Experiment 1: Ambiguity w.r.t window size

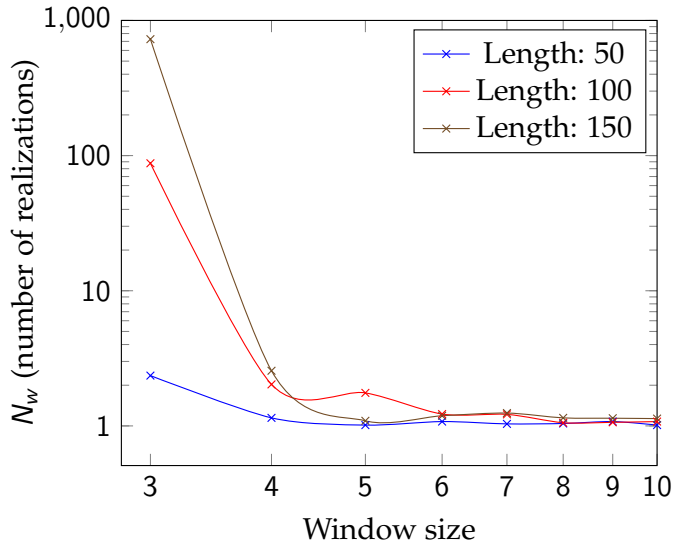


Figure: Estimation of the average number of realizations N_w.
500 documents of English Wikipedia, 2016.

Relation with Language Models: *Experiment 1*

Experiment 1: Ambiguity w.r.t window size

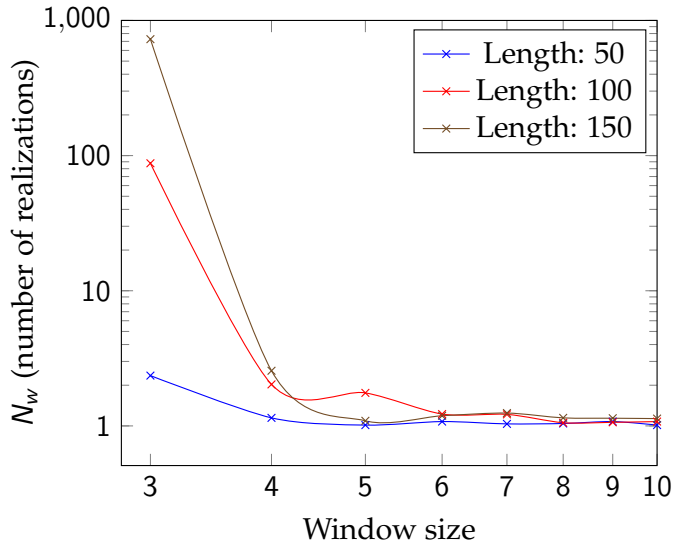


Figure: Estimation of the average number of realizations N_w.
500 documents of English Wikipedia, 2016.

- Average number of distinct realizations tends to 1 w.r.t window size.

Relation with Language Models: *Experiment 1*

Experiment 1: Ambiguity w.r.t window size

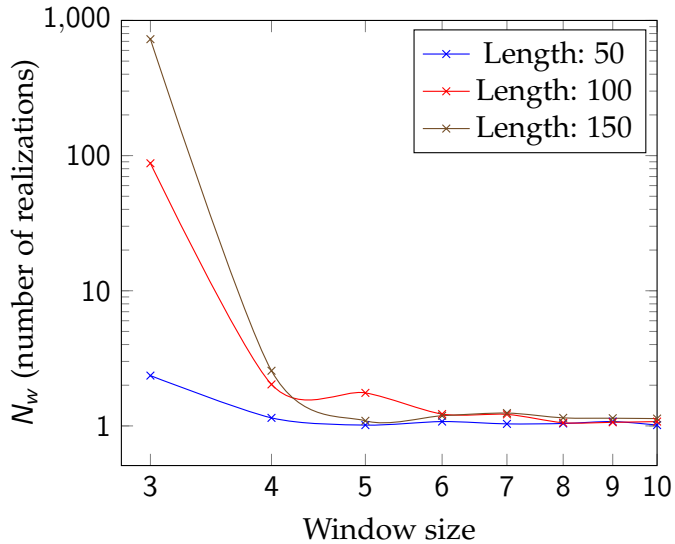


Figure: Estimation of the average number of realizations N_w.
500 documents of English Wikipedia, 2016.

- Average number of distinct realizations tends to 1 w.r.t window size.
- Different realizations exist, even for $w = 10$

Relation with Language Models: *Experiment 1*

Experiment 1: Ambiguity w.r.t window size

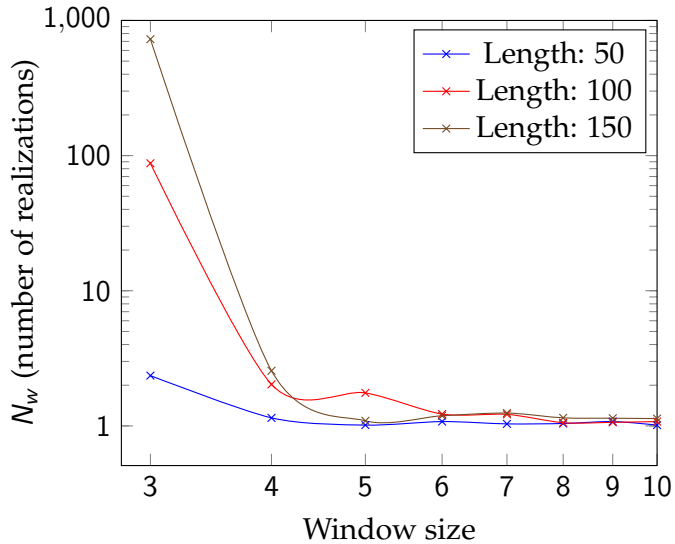


Figure: Estimation of the average number of realizations N_w.
500 documents of English Wikipedia, 2016.

- Average number of distinct realizations tends to 1 w.r.t window size.
- Different realizations exist, even for $w = 10$
- Recommendation of window size w.r.t document length.

Relation with Language Models: *Experiment 2*

Experiment 2: comparison with a LSTM

Relation with Language Models: *Experiment 2*

Experiment 2: comparison with a LSTM

- Generate distinct pairs of realizations of sequence graphs (can be done with similar dynamic approach)

Relation with Language Models: *Experiment 2*

Experiment 2: comparison with a LSTM

- Generate distinct pairs of realizations of sequence graphs (can be done with similar dynamic approach)
- Train a sequential model (Long Short Term Memory Network, LSTM) on two realizations of a given graph to predict the next element of the sequence given the $w-1$ previous ones.

Relation with Language Models: *Experiment 2*

Experiment 2: comparison with a LSTM

- Generate distinct pairs of realizations of sequence graphs (can be done with similar dynamic approach)
- Train a sequential model (Long Short Term Memory Network, LSTM) on two realizations of a given graph to predict the next element of the sequence given the $w-1$ previous ones.
- Evaluate the normed difference of the parameters between pairs of realizations.

Relation with Language Models: *Experiment 2*

Experiment 2: comparison with a LSTM

- Generate distinct pairs of realizations of sequence graphs (can be done with similar dynamic approach)
- Train a sequential model (Long Short Term Memory Network, LSTM) on two realizations of a given graph to predict the next element of the sequence given the $w-1$ previous ones.
- Evaluate the normed difference of the parameters between pairs of realizations.
- Compare the weights of the pair of realizations and with a sequence generated randomly uniformly on the same vocabulary.

Relation with Language Models: *Experiment 2*

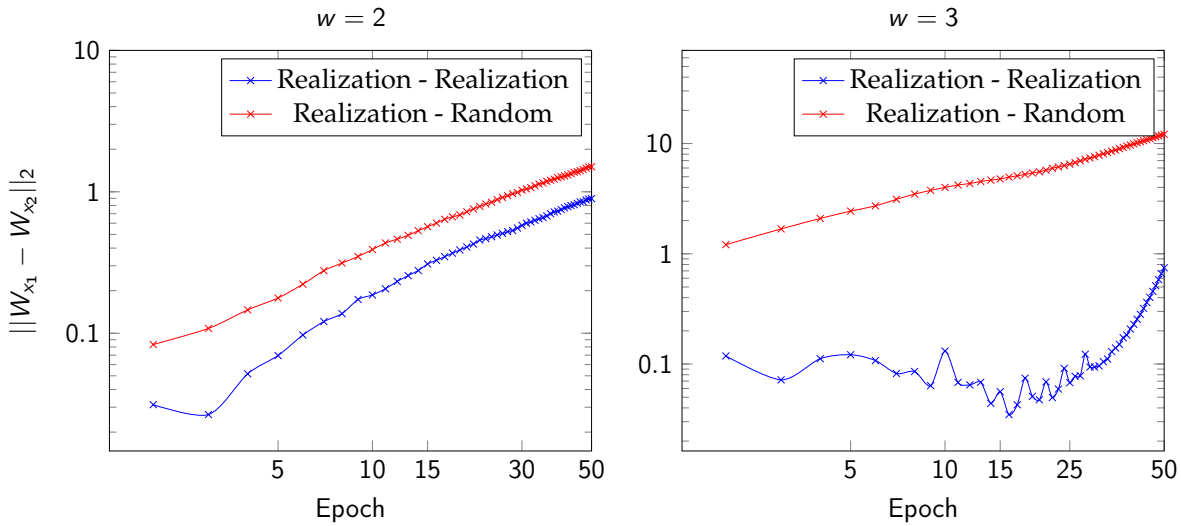


Figure: $||W_{x_1} - W_{x_2}||_2 = f(\text{Epoch})$ - LSTM, log − log scale

Relation with Language Models: *Experiment 2*

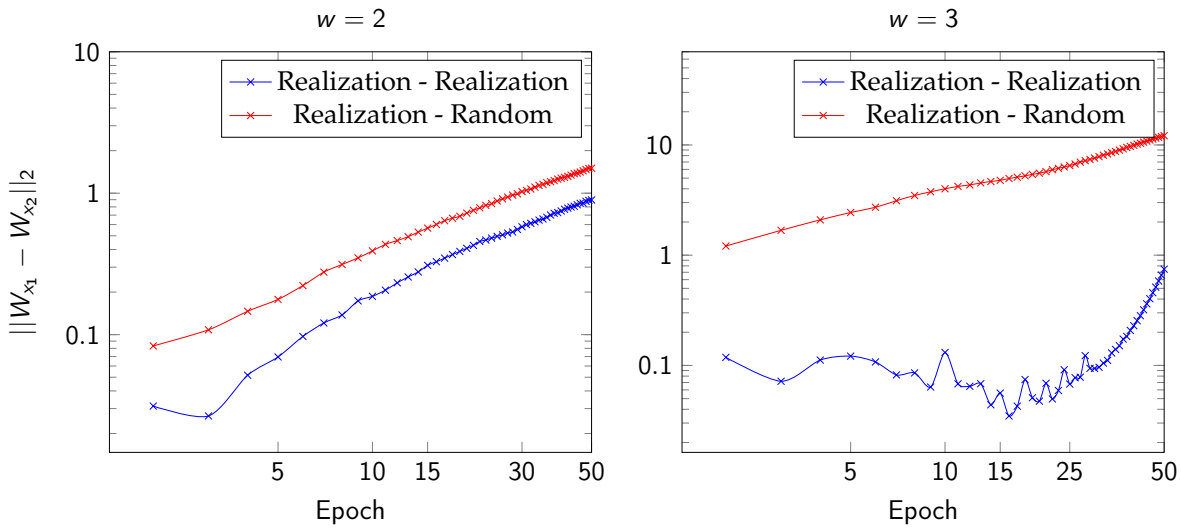


Figure: $||W_{x_1} - W_{x_2}||_2 = f(\text{Epoch})$ - LSTM, log − log scale

- If the sequences were equivalent for the sequential model, the model weights should be similar after training.

Relation with Language Models: *Experiment 2*

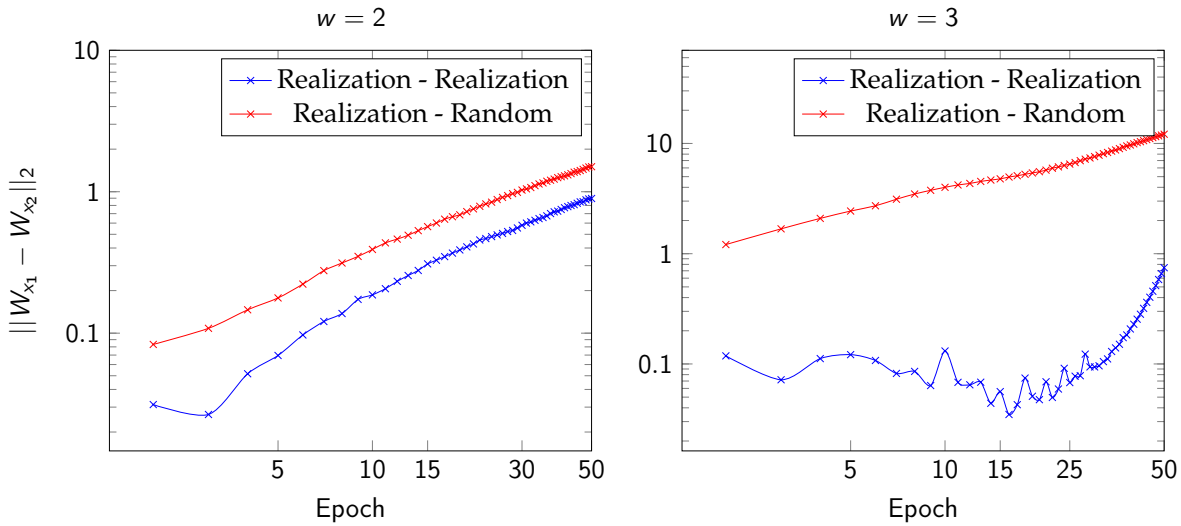


Figure: $||W_{x_1} - W_{x_2}||_2 = f(\text{Epoch})$ - LSTM, log − log scale

- If the sequences were equivalent for the sequential model, the model weights should be similar after training.
- No apparent similarity between different realizations for a LSTM.

Relation with Language Models: *Experiment 2*

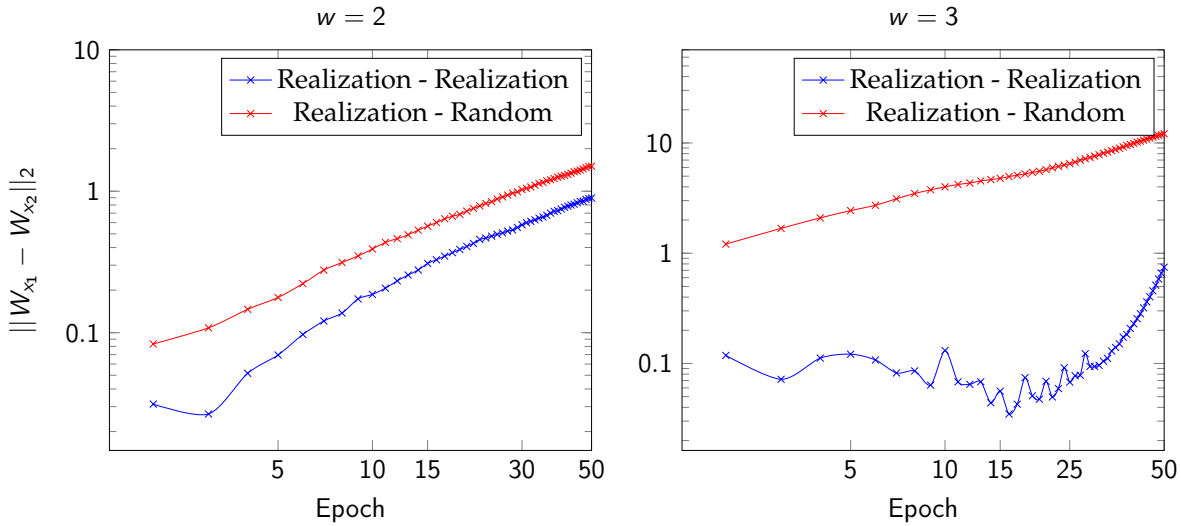


Figure: $||W_{x_1} - W_{x_2}||_2 = f(\text{Epoch})$ - LSTM, log − log scale

- If the sequences were equivalent for the sequential model, the model weights should be similar after training.
- No apparent similarity between different realizations for a LSTM.
- Suggests ambiguity in co-occurrence language models.

Conclusion

- REALIZABLE_w and NUMREALIZATIONS_w seem to be devoid of reductions

Conclusion

- $\textsc{Realizable}_w$ and $\textsc{NumRealizations}_w$ seem to be devoid of reductions
- Partial theoretical results and practical algorithms

Conclusion

- $\textsc{Realizable}_w$ and $\textsc{NumRealizations}_w$ seem to be devoid of reductions
- Partial theoretical results and practical algorithms
- Full complexity characterization is left opened

Conclusion

- $\textsc{Realizable}_w$ and $\textsc{NumRealizations}_w$ seem to be devoid of reductions
- Partial theoretical results and practical algorithms
- Full complexity characterization is left opened
- Allows ambiguity measurement in co-occurrence based models

Conclusion

- $\textsc{Realizable}_w$ and $\textsc{NumRealizations}_w$ seem to be devoid of reductions
- Partial theoretical results and practical algorithms
- Full complexity characterization is left opened
- Allows ambiguity measurement in co-occurrence based models
- Ambiguity is not shared with a recurrent neural network (LSTMs)

2-sequence Graphs

Proposition 6.1

Let $G = (V, E)$ a directed acyclic graph (DAG). G is a 2-sequence graph if and only if it is a straight line (directed tree where each node has at most one child). In this case, G has a unique realization.

Theorem 6.2

Let $G = (V, E)$ a unweighted digraph.
G is a 2-sequence graph $\iff$ $R^+(G)$ is a 2-sequence graph $\iff$ $R^+(G)$ is a straight line and its weights are all equal to 1.

where $R^+(G)$ is the DAG obtained by weighted contraction of G in strongly connected components. Proofs by contradiction.

General Case: Direct Formulation and Digraphs

- Digraphs: the analogue procedure enumerates all paths in the DAG $R(H^{(w-2)})$, where the operator R is the contraction in strongly connected components.

General Case: Direct Formulation and Digraphs

- Digraphs: the analogue procedure enumerates all paths in the DAG $R(H^{(w-2)})$, where the operator R is the contraction in strongly connected components.
- The number of paths can be exponential, even for the auxiliary graph $R(H^{(w-2)})$ obtained from a sequence graph.

Example

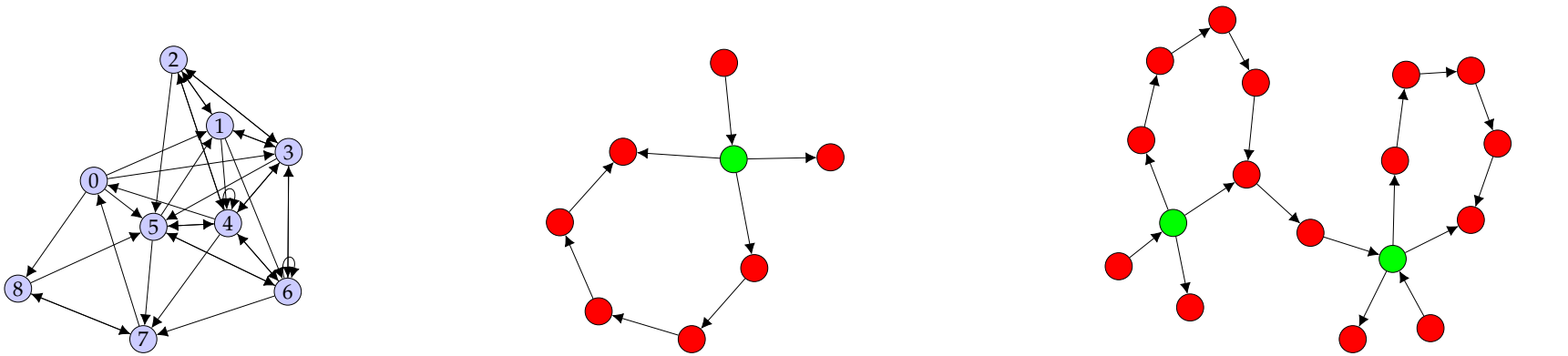
$$x_{\text{init}} = 0, 3, 1, 2, 3, 4, 4, 2, 5, 1, 4, 6, 3, 5, 6, 6, 4, 7, 0, 8, 5, 7 \tag{1}$$

For $P \in \mathbb{N}^*$, let $x \in \mathbb{N}^{22P}$:

$$\forall i \in \{0, \ldots, 22P - 1\} \qquad x_i = x_{\text{init}}[i] + 10 \left\lfloor \frac{i}{22} \right\rfloor \tag{2}$$

Claim: With $w = 3$, $R(H(x))$ has at least 2^P paths.

General Case: Direct Formulation and Digraphs



(a) $G = \phi_3(x_{\text{init}})$ (b) $R(H(x)) \quad (P = 1)$ (c) $R(H(x)) \quad (P = 2)$

Figure: Direct approach has exponential complexity in the worst case

- For digraphs and weighted graphs, the complexity classes characterizations of NUMREALIZATIONS$_w$ and REALIZABLE$_w$ remain opened ($w \geq 3$).

General Case: Integer Programming Formulation

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

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

$$\forall e = (v_1, v_2) \in E$$
$$\forall e' = (v_1', v_2') \notin E$$
$$\forall i \in \{1,\dots,w-1\}$$

$$\begin{cases}
-X_{v_1,1} + y_1^e(i) \le 0 & \\
-X_{v_2,1+i} + y_1^e(i) \le 0 & \\
X_{v_1,1} + X_{v_2,1+i} - y_1^e(i) \le 1 & X_{v_1',1} + X_{v_2',1+i} \le 1 \\
\vdots & \vdots \\
-X_{v_1,p-i} + y_{p-i}^e(i) \le 0 & X_{v_1',p-i} + X_{v_2',p} \le 1 \\
-X_{v_2,p} + y_{p-i}^e(i) \le 0 & \\
X_{v_1,p-i} + X_{v_2,p} - y_{p-i}^e(i) \le 1 &
\end{cases}$$

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

General Case: Integer Programming Formulation

If the minimum obtained is $\sum_{e \in E} \pi_e$ then the output of $\text{REALIZABLE}_w(G, \Pi, w)$ is True, and False otherwise.

- Branch and bound methods available

General Case: Integer Programming Formulation

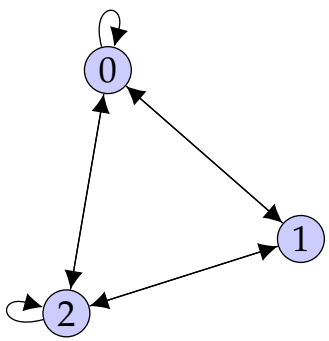
If the minimum obtained is $\sum_{e \in E} \pi_e$ then the output of $\textsc{Realizable}_w(G, \Pi, w)$ is True, and False otherwise.

- Branch and bound methods available
- However $n^2(w-1)(p - \frac{w-2}{2})$ constraints
  $\rightarrow w = 3, n = 20, p = 10$: 7600 constraints. Not solvable in practice with Branch and Bound.

General Case: Integer Programming Formulation

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

- Branch and bound methods available
- However $n^2(w-1)(p - \frac{w-2}{2})$ constraints
  $\rightarrow w = 3, n = 20, p = 10$: 7600 constraints. Not solvable in practice with Branch and Bound.



(a) Graph G

$$\begin{pmatrix} 2 & 3 & 1 \\ 2 & 0 & 2 \\ 2 & 3 & 2 \end{pmatrix}$$

(b) Matrix Π

2 2 1 2 0 0 1 0 2 1

(c) A solution

Figure: Example of instance solved using the Linear Integer Programming formulation ($w = 3$).

General Case: Direct Formulation

Idea: For $w > 3$, recursively iterate the process for $k \leq w - 2$:

$$E^{(k)} = \{u_{1:(k+1)} \in V^{k+1} \mid u_{1:k} \in E^{(k-1)},\ u_{2:(k+1)} \in E^{(k-1)} \\ \text{s.t}(u_1, u_{k+1}) \in E\}$$

$$H^{(k)} \triangleq (E^{(k)}, E^{(k+1)})$$

Proposition 6.3

Let $x = x_1, \ldots, x_p$ be a w-realization of a graph $G = (V, E)$. If $w \leq p$, x, then x corresponds to a walk of length $p - w + 1$ on $H^{(w-2)}$.

$$\rightarrow \text{Proof by induction on } k \in \{1, \ldots p\}$$

General Case: Dynamic Programming Formulation

$N_w[\Pi, p, \mathbf{u}]$ number of realizations:

- of the graph with weights matrix Π
- of length p with starting pattern $\mathbf{u}$

General Case: Dynamic Programming Formulation

$N_w[\Pi, p, \mathbf{u}]$ number of realizations:

- of the graph with weights matrix Π
- of length p with starting pattern $\mathbf{u}$

Recursion

General Case: Dynamic Programming Formulation

$N_w[\Pi, p, \mathbf{u}]$ number of realizations:

- of the graph with weights matrix Π
- of length p with starting pattern $\mathbf{u}$

Recursion

$$N_w[\Pi, p, \mathbf{u}] = \sum_{v \in V} \Phi\left(\Pi'_{(\mathbf{u},v)}, p-1, \mathbf{u}\right)$$

General Case: Dynamic Programming Formulation

$N_w[\Pi, p, \mathbf{u}]$ number of realizations:

- of the graph with weights matrix Π
- of length p with starting pattern $\mathbf{u}$

Recursion

$$N_w[\Pi, p, \mathbf{u}] = \sum_{v \in V} \Phi\left(\Pi'_{(\mathbf{u},v)}, p-1, \mathbf{u}\right)$$

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

General Case: Dynamic Programming Formulation

$N_w\,[\Pi, p, \mathbf{u}]$ number of realizations:

- of the graph with weights matrix Π
- of length p with starting pattern $\mathbf{u}$

Recursion

$$N_w\,[\Pi, p, \mathbf{u}] = \sum_{v \in V} \Phi\,(\,\Pi'_{(\mathbf{u},v)}, p-1, \mathbf{u}\,)$$

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

Π' is an update of Π following:

$$\Pi'_{(\mathbf{u},v)} := (\,\pi_{ij} - \#\{\,k \in \{1, ..., |\mathbf{u}|\} \mid (u_k, v) = (i,j)\,\}\,)_{(i,j)\in V^2}$$

General Case: Dynamic Programming Formulation

Initialisation

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

The total number of admissible sequences is $N_w[\Pi, p, \varepsilon]$, ε being the empty prefix.

Dynamic Programming Formulation: Complexity Study

Proposition 6.4

Dynamic programming formulation: Estimation of the Worst case complexity $\mathcal{W}_C$ and Average complexity $\mathcal{A}_C$.

$$\mathcal{W}_C = n^w 2^{w\,p} \qquad \mathcal{A}_C = n^w \mathcal{E}_C$$

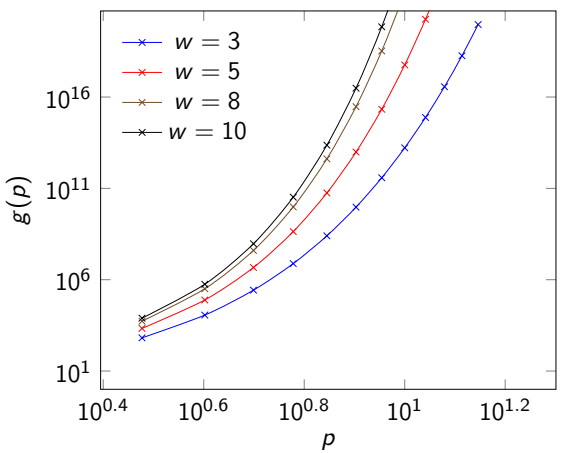
$$\mathcal{L}(wp, n^2) \leq \mathcal{E}_C \leq \mathcal{U}(wp, n^2)$$

- n *number of words*, w *window size*, p *sequence length*
- $\mathcal{L}(m,r) = \frac{1}{\binom{m+r-1}{r-1}} \sum_{m_z=1}^{\min(r,m)} \binom{r}{m_z}\binom{m-1}{m_z-1} 2^{m_z-1}(m-m_z+2)$
- $\mathcal{U}(m,r) = \frac{1}{\binom{m+r-1}{r-1}} \sum_{m_z=1}^{\min(r,m)} \binom{r}{m_z}\binom{m-1}{m_z-1}(\frac{m}{m_z}+1)^{m_z}$

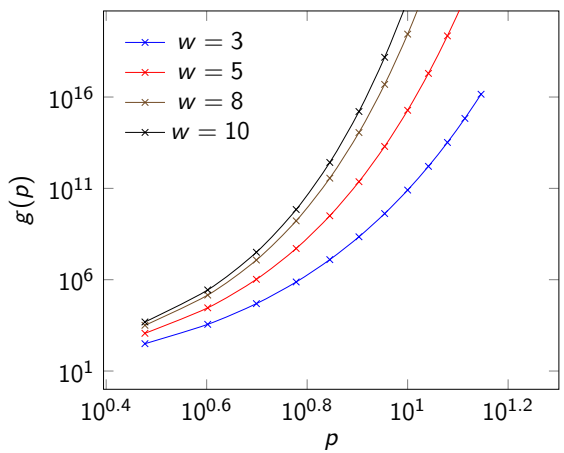
A lower bound of the ratio $\frac{\mathcal{W}_C}{\mathcal{A}_C}$ *is given by* $\frac{2^{w\,p}}{\mathcal{U}(w\,p, \alpha^2 p^2)}$.

Dynamic Programming Formulation: Complexity Study

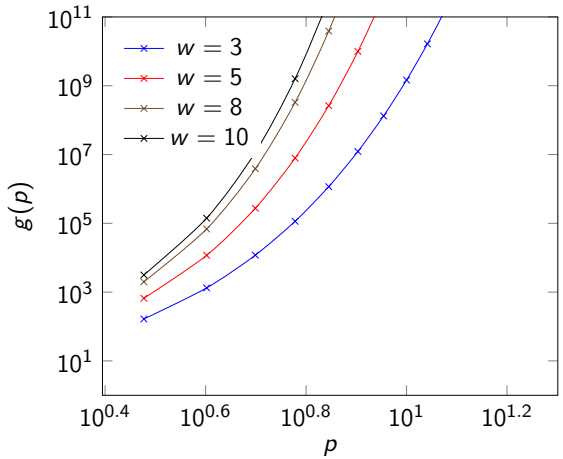
Lower bound of average complexity $\mathcal{A}_{\mathcal{C}}$ of the dynamic programming formulation, for several values of w and α (ratio of distinct words over p) in log $-$ log scale.



(a) $\alpha = 0.5$



(b) $\alpha = 0.66$



(c) $\alpha = 0.83$

- Still suffers from exponential complexity

Identification: use the Knowledge Graph

Algorithm 1: Feature extraction with knowledge graph and ontology

Data: Knowledge Graph $\boldsymbol{G}$, Query $\boldsymbol{q}$, Entity candidate $\boldsymbol{e}$ with initial score s^0, Types $(\boldsymbol{t}_j)_{1 \leq j \leq \boldsymbol{T}}$

Result: Entity candidate representation

$\boldsymbol{X}^{q,e} = [s^0]$;

Get neighbor nodes of $\boldsymbol{e}$;

for $j = 1$ *to* $\boldsymbol{T}$ **do**

  Aggregate text description of neighbors of type t_j;

  Compute score s_{t_j} between $\mathcal{N}_{t_j}(\boldsymbol{e})$ and the query $\boldsymbol{q}$;

  Append s_{t_j} to $X^{q,e}$;

end

Return Score vectors $(\boldsymbol{X}^{q,e})_{1 \leq j \leq \boldsymbol{T}+1}$

Sum up: Graph-based Comparison

Algorithm 2: Named entity identification (Inference)

Data: Knowledge base $\boldsymbol{B}$ and its graph $\boldsymbol{G_B}$, queries $(\boldsymbol{q}_i)_{1\leq i\leq M}$, scoring threshold $\boldsymbol{K}$, trained predictor $\hat{\boldsymbol{F}}$

Result: Entity candidate representation

1 **for** $i = 1$ *to* M **do**

2 Use filtering with query $\boldsymbol{q}_i$, get list of $\boldsymbol{K}$ top ranked entities $(\boldsymbol{e}_h^1)_{1\leq h\leq K}$;

3 Use algorithm 1 to return $\boldsymbol{K}$ entity candidates new representations;

4 Evaluate $\hat{\boldsymbol{F}}$ on each vector score and use maximum a posteriori to infer estimated gold entity $\hat{\boldsymbol{g}}_i$;

5 **end**

6 Return $(\hat{\boldsymbol{g}}_i)_{1\leq i\leq M}$ (list of estimated gold entities);