Unexpected attributed subgraphs: a mining algorithm

Simon Delarue [1], Tiphaine Viard [2] and Jean-Louis Dessalles [1]

[1]LTCI, Télécom Paris, Institut Polytechnique de Paris [2]i3, Télécom Paris, Institut Polytechnique de Paris

Key insights

- Pattern mining algorithm on attributed graphs
- Information-theory based filter: Unexpectedness
- Explainable and concise outputs

Context & motivations

- Why graphs? Graphs are everywhere

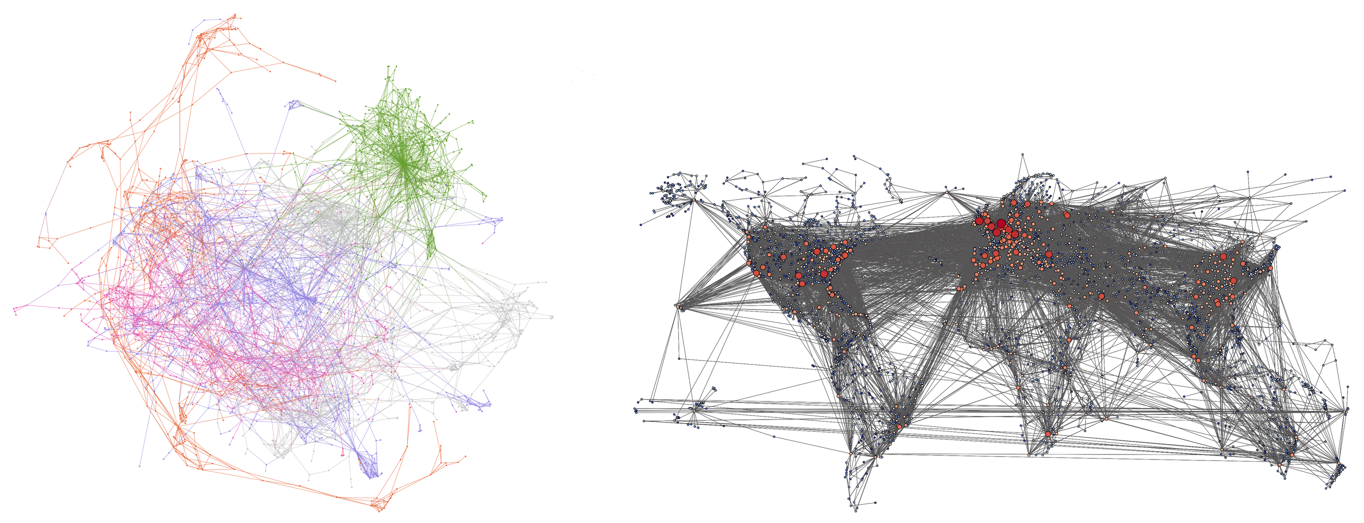


Figure 1: Cora citation network (left) and airplane traffic network (right).

Challenges:

- Real-world networks = **large attributed graphs** $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{A})$. If $u, v \in \mathcal{V}$ are nodes, $(uv) \in \mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ denotes a directed link between u and v. $\mathcal{A}$ is a set of node attributes
- Difficult to **find information**
- Even more difficult to find **"interesting"** information

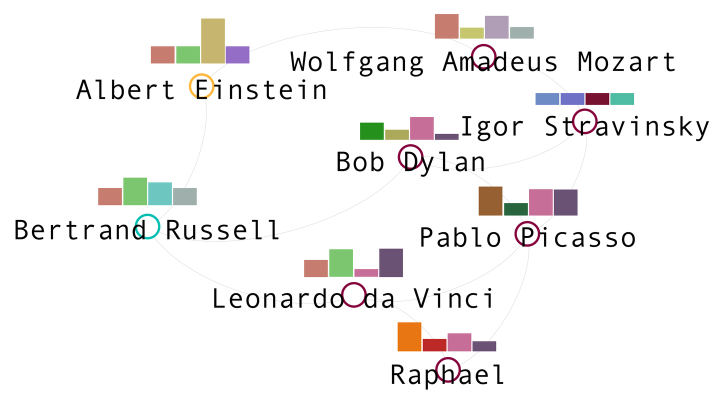


Figure 2: Attributed graph with node labels and features.

Goals:

- Extract **subgraphs** or **patterns**

$$p = \big(G = (V, E), A\big)$$

with $A = \{a \in \mathcal{A}, \forall v \in V, a(v)\}$ and $G \subseteq \mathcal{G}$ that summarize well the initial information
- Make sure these patterns are "**interesting**" enough
- Remain **computationally efficient**

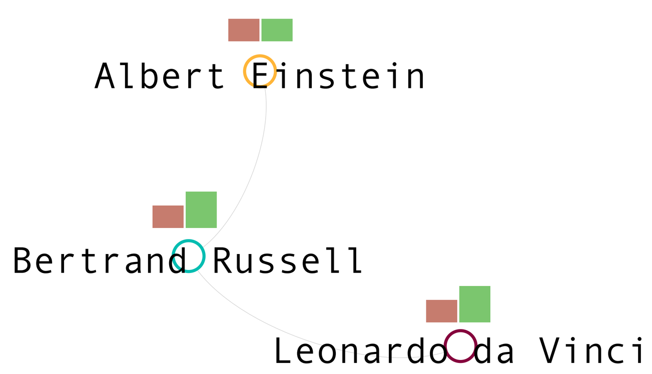


Figure 3: Pattern with nodes sharing common characteristics.

Interestingness measure

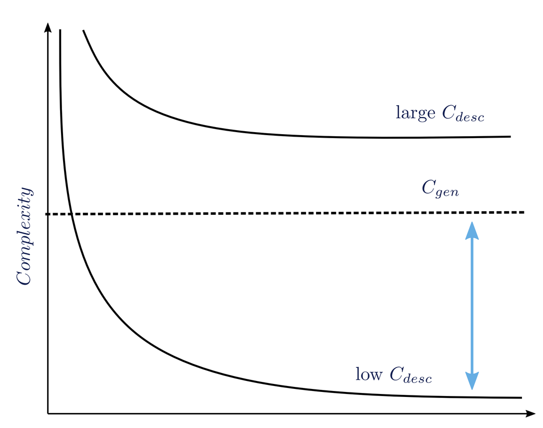
Information-theory based (simplicity theory): **Unexpectedness** [2]

$$U = C_{\text{gen}} - C_{\text{desc}}$$

with C_{gen} the complexity to generate the event and C_{desc} the complexity to describe it

Example: What is the most unexpected lottery draw between the following? `41-34-15-4-28-8` or `1-2-3-4-5-6`

- Unexpectedness = **drop of complexity**
- Simple events appear unexpected



Contact information

- `simon.delarue@telecom-paris.fr`
- `tiphaine.viard@telecom-paris.fr`
- `jean-louis.dessalles@telecom-paris.fr`

Mining algorithm

Main idea: Mine patterns[1] that are unexpected on both **structure** and **attribute** levels

$$U(p) = U(G) + U(A)$$

$$C_{\text{desc}}(G) = \underbrace{\log(|V|)}_{nodes} + \underbrace{\sum_{v \in V} \log(b+1) + \log\left(\binom{|V|}{k_v}\right)}_{edges} \qquad C_{\text{desc}}(A) = \sum_{a \in A} \log(\#a)$$

$$C_{\text{gen}}(\mathcal{G}, m) = C_{\text{desc}}(\tilde{G}), \tilde{G} \sim \mathcal{G}(m),\ 0 \leq m \leq |\mathcal{A}| \qquad C_{\text{gen}}(\mathcal{A}, A) = \log\left(\binom{|\mathcal{A}|}{|A|}\right)$$

with $b = \max_{v \in V}(\deg(v))$, $k = |\mathcal{N}(v)|$ and $\#a$ the number of occurences of a

Results

Real-world network **`Wikischools`**, $|\mathcal{V}| = 4403, |\mathcal{E}| = 112834, |\mathcal{A}| = 20527$

- **396 unexpected subgraphs**

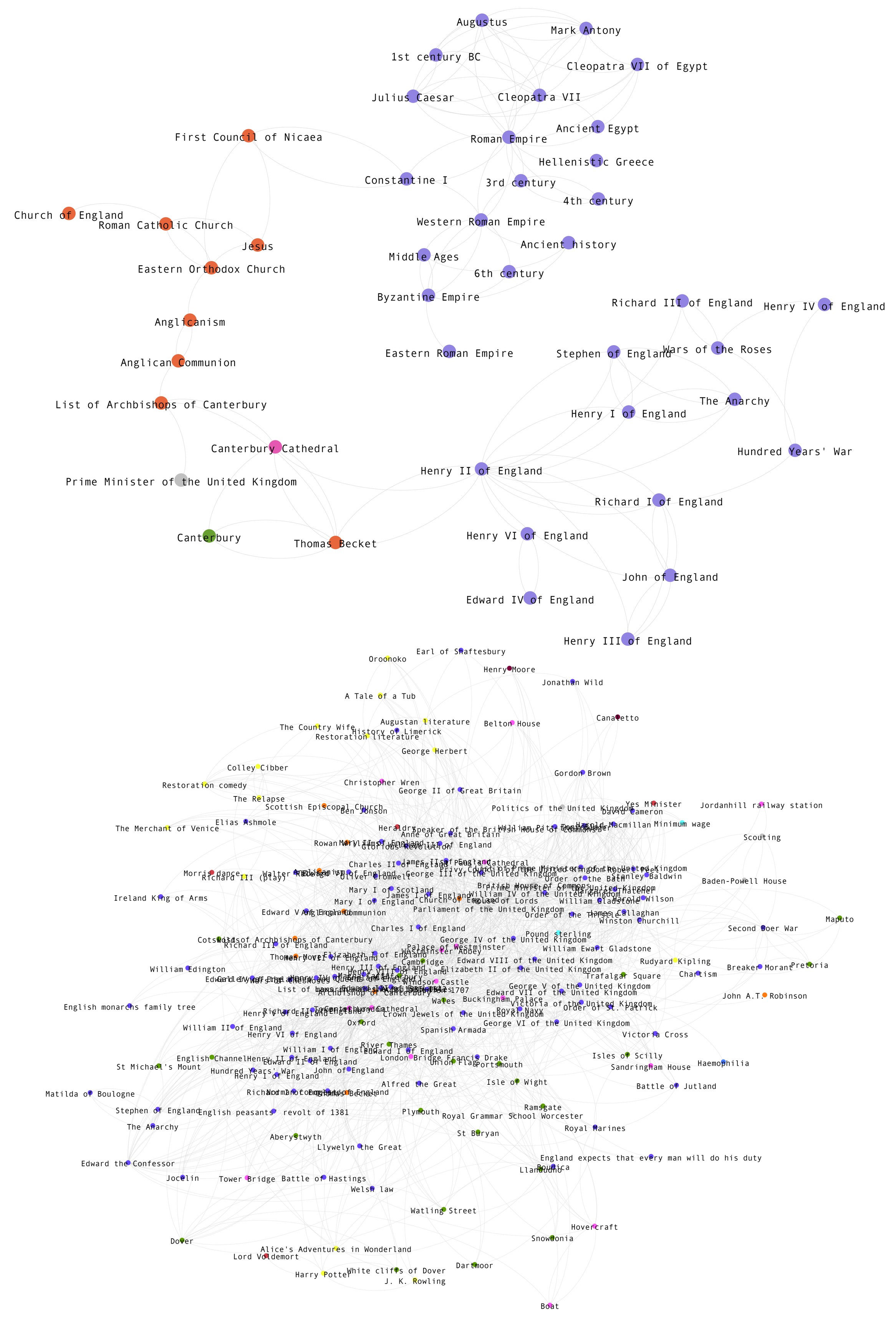


Figure 4: Unexpected subgraph: 43 nodes, 31 words `['justinian', 'antique', 'heir', 'richard', 'rome', 'throne', ...]` (top). Louvain cluster sharing the most nodes with pattern: 191 nodes, 3979 words (bottom).

Conclusion & future work

+	−
Explainable outputs: easy-to-read subgraphs with concise summaries	May miss patterns
Hierarchy between patterns	Sensitive to parameters

Applications

- Ad-hoc explanations for Machine Learning outputs
- Query the graph with nodes/keywords/structures

References

[1] S. Andrews. In-Close, a Fast Algorithm for Computing Formal Concepts. In *International Conference on Conceptual Structures (ICCS)*, page 15, 2009.

[2] J.-L. Dessalles. Coincidences and the encounter problem: A formal account. *arXiv preprint arXiv:1106.3932*, 2011.