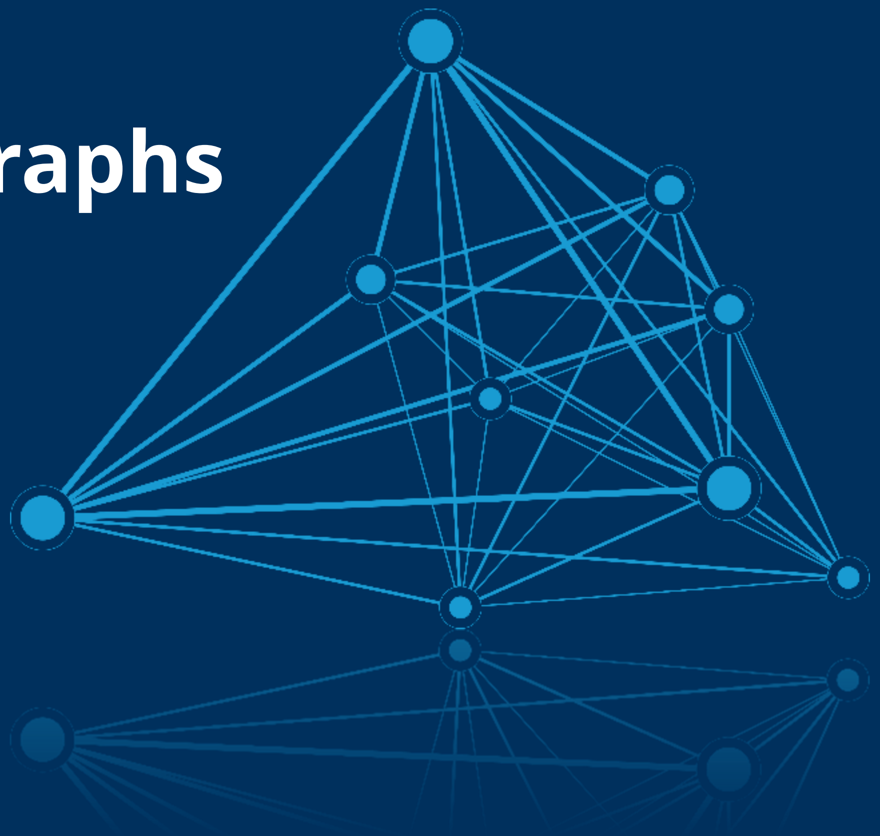
TECHNISCHE UNIVERSITÄT DRESDEN

DRESDEN concept

Blockmodelling Knowledge Graphs



Marcin Pietrasik
University of Alberta

June 2, 2022 // Dresden, Germany

Outline

- Introduction
- Knowledge graph basics
- Stochastic blockmodelling basics
- Past work
- Current and future work

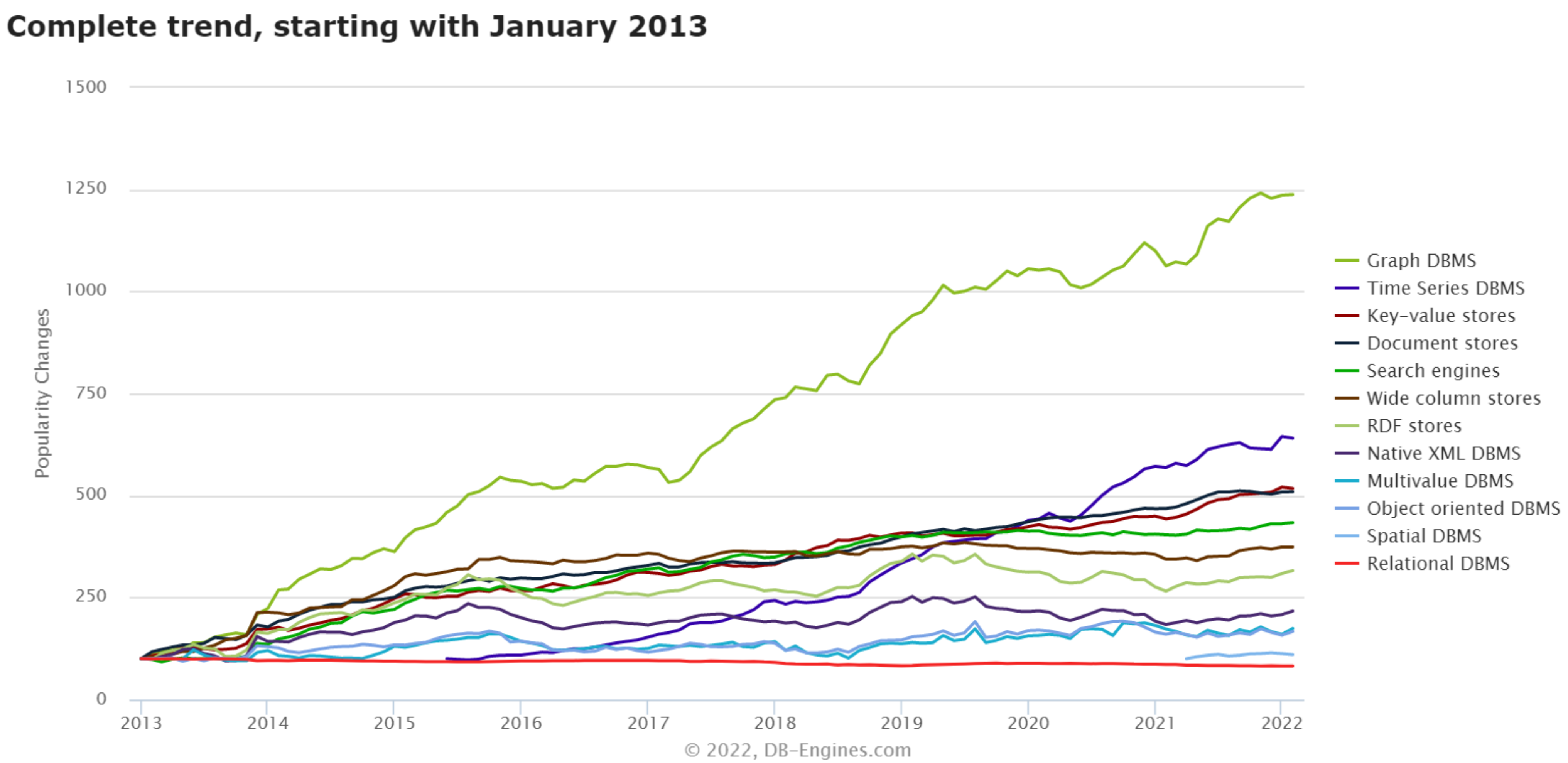
Introduction

- I am a fifth year PhD student in the Department of Electrical and Computer Engineering at the University of Alberta in Edmonton, Canada
- Background is in **Computing Science** (BSc. 2016) with a focus on statistical machine learning
- Guest at **TU Dresden** from May until August

Introduction

- Broadly, my research lies at the intersection of **knowledge graphs** and **artificial intelligence**
- Topics have changed considerably since the starting graduate school
  - Started in modelling dynamic social networks
  - Internships: computer vision, ontology creation, time series forecasting, etc.
- Currently I am working on **learning hierarchies** from knowledge graphs

Knowledge graphs - motivation



Knowledge graphs

- Definition of knowledge graphs varies in the community
  - In this talk, simplest definition
- Knowledge graphs are a method of storing data as a graph structure
- Composed of **two** main **components**:
  - **Entities** (nodes, vertices, points, nouns, etc.)
  - **Predicates** (relations, edges, links, verbs, etc.)

Triples

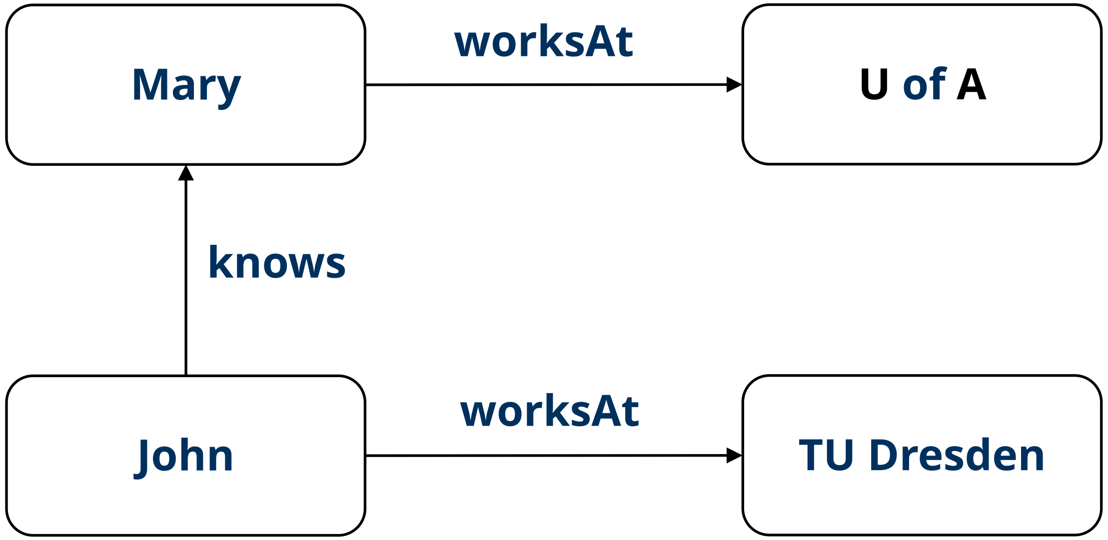
- Triples (facts) are how data is stored in a knowledge graph
  - Links a **subject** (head) to an **object** (tail) via a **predicate**

<John> <worksAt> <TU Dresden>.

Triples

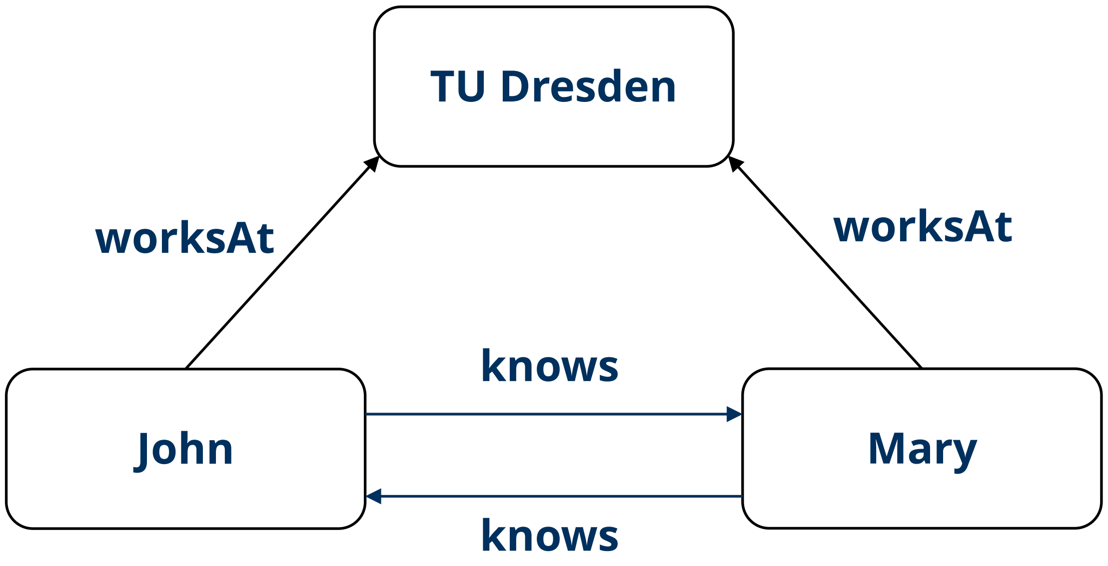
- Put triples together and you geta knowledge graph

<Mary> <worksAt> <U of A>.
<John> <knows> <Mary>.
<John> <worksAt> <TU Dresden>.



Tensor representation

- A knowledge graph can be represented as a three-dimensional binary tensor



knows

	J	M	T
J	0	1	0
M	1	0	0
T	0	0	0

worksAt

	J	M	T
J	0	0	1
M	0	0	1
T	0	0	0

Questions?

Stochastic blockmodels

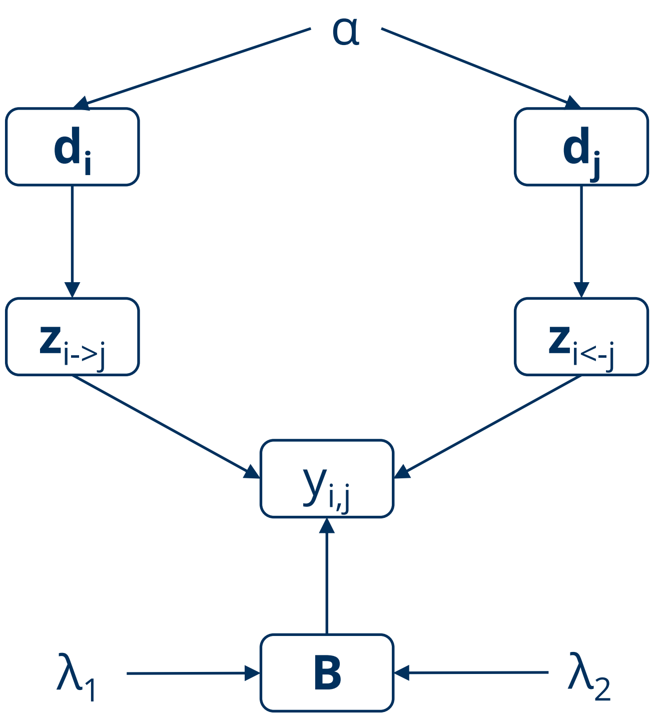
- Stochastic blockmodels are **generative models** for graphs
- Decompose a graph into **probability distributions**, for **blocks** of the model
- When **sampled** from, the blocks generate the graph
- Th learning process is then to infer the parameters of these distributions

Stochastic blockmodels

- Lego analogy

Stochastic blockmodels - example

- For each entity *i* in graph
  - $\mathbf{d}_i \sim \text{Dirichlet}(\alpha)$
- For each community *(p,q)*
  - $b_{p,q} \sim \text{Beta}(\lambda_1, \lambda_2)$
- For each interaction *(i,j)* in graph
  - $\mathbf{z}_{i\text{->}j} \sim \text{Multinomial}(\mathbf{d}_i)$
  - $\mathbf{z}_{i\text{<-}j} \sim \text{Multinomial}(\mathbf{d}_j)$
  - $y_{i,j} \sim \text{Bernoulli}(\mathbf{z}_{i\text{->}j}\, \mathbf{B}\, \mathbf{z}_{i\text{<-}j})$



How can we learn the parameters of the model?

Statistical inference

- Process by which parameters of a model are **learned** from the data
- Analogous to the training process in other machine/deep learning methods
- In my work, usually use **Gibbs sampling**

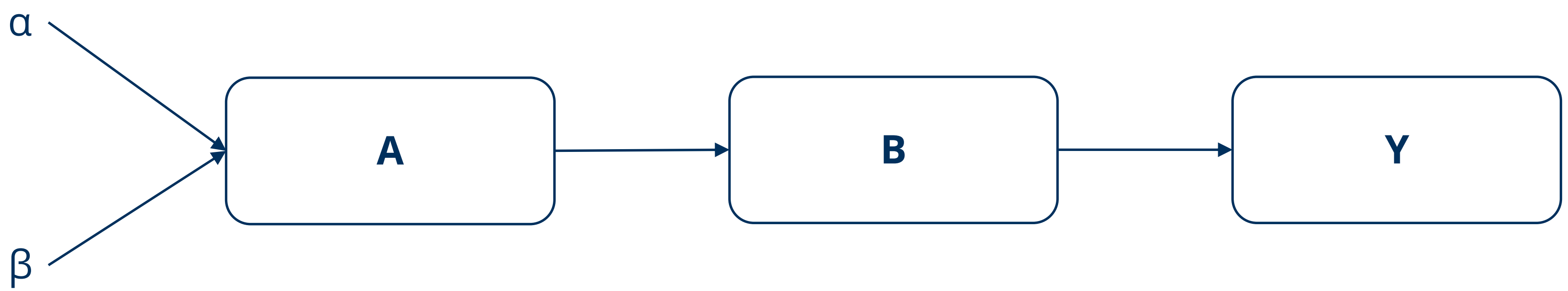
Gibbs sampling

- **Markov Chain Monte Carlo** method for approximating probability distributions
- Used in stochastic blockmodelling to approximate the joint distribution of the model
- **Condition**: it is easier to sample from the conditional distributions of the model than the joint distribution

Gibbs sampling

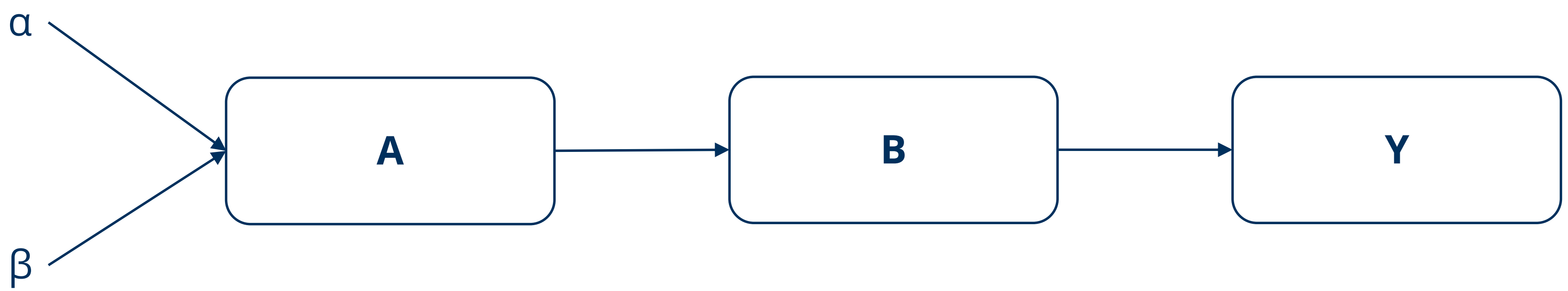
- **Basic idea**: iteratively sample from full conditional distributions of model parameters
  - **Note**: the parameters that samples are conditioned on are always changing, allowing the sampling to converge
- Early samples do not reflect the desired distribution and must be **discarded** (burned in)
  - Burning in a Gibbs sampler is analogous to the training step of machine/deep learning methods

Gibbs sampling - example



- **A** is a random variable from some probability distribution with hyperparameters α and β
- **B** is some probability distribution
- **Y** is the output of the model

Gibbs sampling - example



1. Initialize **A** and **B** with some values in the support
2. For *i* iterations
    1. Obtain new **A** from p(**A**|**Y**,**B**,α,β)
    2. Obtain new **B** from p(**B**|**Y**,**A**,α,β)

Gibbs sampling pros/cons

- **Pros**
  - Sampling from the conditionals is easy (well… easier)
  - You can set up the model in a way to integrate out parameters (collapsing Gibbs sampler)
- **Cons**
  - When parameters are strongly correlated, sampler can get stuck
  - Analytical integration can be difficult on complex models
  - SLOW!

Collapsing Gibbs sampler

- Collapsing a Gibbs sampler means **analytically integrating** out parameters of the model
- Parameters that are integrated out **do not** need to be sampled
- Less parameters to sample = **faster** sampling process

Collapsing Gibbs sampler - example

$$\mathbb{P}(c_{pqr} \mid \mathbf{C}_{-(pqr)}, \mathbf{G}, \mathbf{P}, \mathbf{Z}, \lambda, \eta) = \frac{\mathbb{P}(\mathbf{G} \mid \mathbf{C}, \mathbf{P}, \mathbf{Z}, \lambda, \eta)\mathbb{P}(c_{pqr} \mid \mathbf{C}_{-(pqr)}, \lambda, \eta)}{\int_{c_{pqr}} \mathbb{P}(\mathbf{G} \mid \mathbf{C}, \mathbf{P}, \mathbf{Z}, \lambda, \eta)\mathbb{P}(c_{pqr} \mid \mathbf{C}_{-(pqr)}, \lambda, \eta)\, \mathrm{d}c_{pqr}}$$

HARD!

$$a = \left|\left\{g_{xyz} \in \mathbf{G} \;:\; (x,y) \neq (i,j) \wedge \Psi(x,y) = \Psi(i,j) \wedge g_{xyz} = 1\right\}\right|$$

$$b = \left|\left\{g_{xyz} \in \mathbf{G} \;:\; (x,y) \neq (i,j) \wedge \Psi(x,y) = \Psi(i,j) \wedge g_{xyz} = 0\right\}\right|$$

$$\begin{aligned}
\mathbb{P}(c_{pqr} \mid \mathbf{C}_{-(pqr)}, \mathbf{G}, \mathbf{P}, \mathbf{Z}, \lambda, \eta) &= \frac{\left(\binom{a+b}{a} c_{pqr}^{a}(1-c_{pqr})^{b}\right)\left(\frac{c_{pqr}^{\lambda-1}(1-c_{pqr})^{\eta-1}}{\mathrm{B}(\lambda,\eta)}\right)}{\int_{c_{pqr}} \left(\binom{a+b}{a} c_{pqr}^{a}(1-c_{pqr})^{b}\right)\left(\frac{c_{pqr}^{\lambda-1}(1-c_{pqr})^{\eta-1}}{\mathrm{B}(\lambda,\eta)}\right) \mathrm{d}c_{pqr}} \\
&= \frac{\left(\frac{\binom{a+b}{a}}{\mathrm{B}(\lambda,\eta)}\right)\left(c_{pqr}^{a+\lambda-1}(1-c_{pqr})^{b+\eta-1}\right)}{\frac{\binom{a+b}{a}}{\mathrm{B}(\lambda,\eta)} \int_{c_{pqr}} c_{pqr}^{a+\lambda-1}(1-c_{pqr})^{b+\eta-1}\, \mathrm{d}c_{pqr}} \\
&= \frac{c_{pqr}^{a+\lambda-1}(1-c_{pqr})^{b+\eta-1}}{\int_{c_{pqr}} c_{pqr}^{a+\lambda-1}(1-c_{pqr})^{b+\eta-1}\, \mathrm{d}c_{pqr}} \\
&= \frac{c_{pqr}^{a+\lambda-1}(1-c_{pqr})^{b+\eta-1}}{\mathrm{B}(a+\lambda, b+\eta)} \\
&= \mathrm{Beta}(a+\lambda, b+\eta)
\end{aligned}$$

EASY! We know how to do this

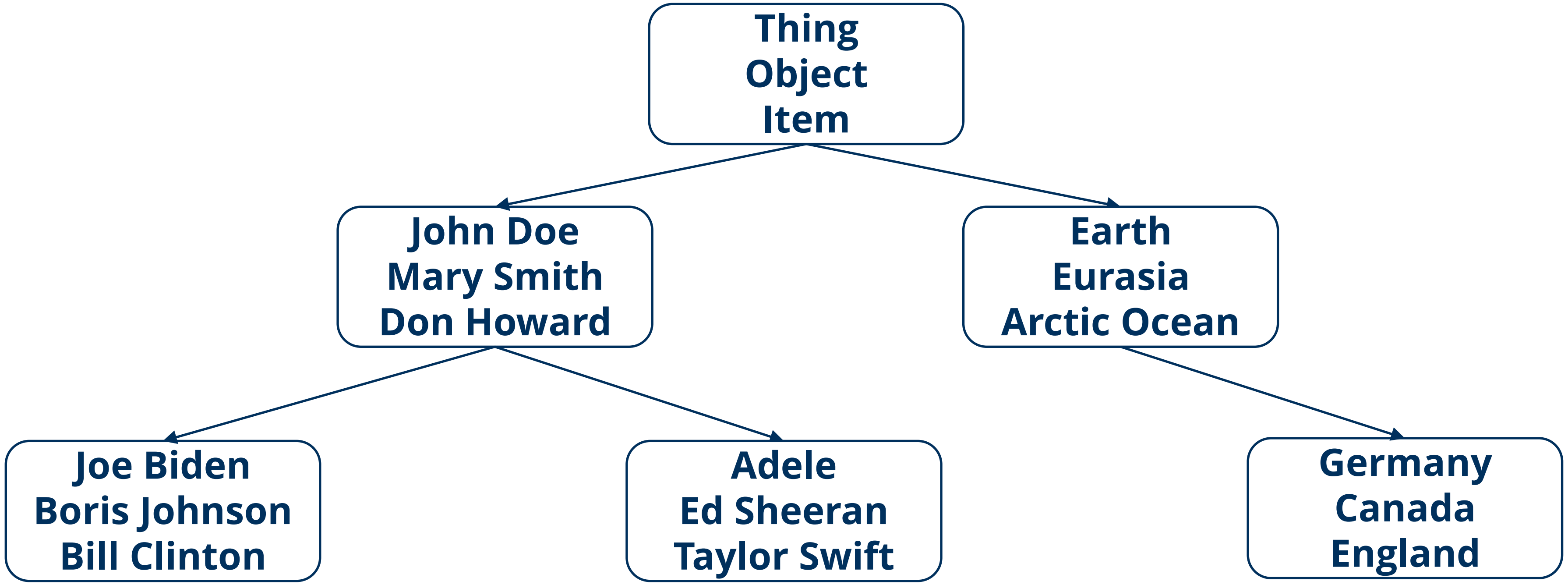
Questions?

Past work

- Deep Dynamic Mixed Membership Stochastic Blockmodel
- Fragmentation Coagulation Mixed Membership Stochastic Blockmodel
- Neural Blockmodeling for Multilayer Networks
- Hierarchical Topic Modelling for Knowledge Graphs

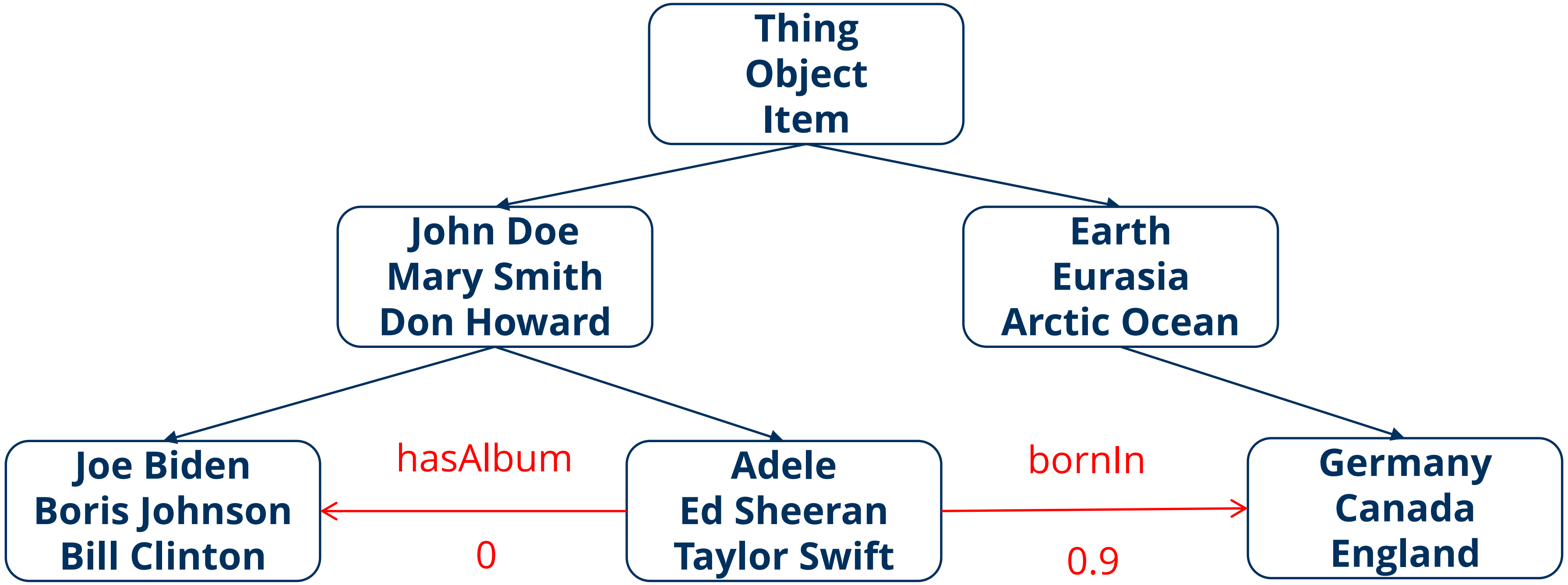
Current work

- Stochastic blockmodel to learn **hierarchies** from knowledge graphs



Current work

- Stochastic blockmodel to learn **hierarchies** from knowledge graphs



Current work

- **Problem**: how can we model a hierarchy in the statistical framework
- **Solution**: Dirichlet processes
  - Nested Chinese Restaurant Process: for generating paths for entities in infinite tree
  - Stick Breaking Process: assigning entities a level in infinite depth

Current work – next steps

- Make it faster!
  - Parallelization?
  - Sparse sampling?
- Apply to more real-world data
- Make changes to model? Depends on performance on large datasets

Questions?