**MULTI-DOMAIN SUPPORT**

Example: Entrepreneurial start-up founding. Such a setting entails diverse personalization scenarios. A recommender system needs to suggest (i) relevant experts that can provide feedback to an innovation idea, (ii) support potential co-founder and team member matching, (iii) allow accelerators, incubators, and innovation hubs to discover these innovations as well as (iv) continuously provide relevant education materials until the innovation idea has become mature enough in order to form a start-up.

Objective: The utilized recommender algorithms need to be customized with respect to the underlying data structures.

**Co-Founder Matching:**
- Increase skill diversity
- Maintain personal compatibility

**Production Setup.** Different challenges need to be addressed in order to continuously maintain the stability and health of a recommender system. A distributed architecture should be guided by design principles that:
  - Provide service isolation
  - Support data heterogeneity
  - Allow for algorithmic customization
  - Ensure fault tolerance

**FRAMEWORK FOR A MULTI-DOMAIN ENVIRONMENT**

**REAL-TIME PERFORMANCE**

Adapting for real-time recommendations. Response times in most cases need to be below 100-200 milliseconds. Embeddings can be stored as payloads in Lucene and Cosine similarity can be used for retrieval.

Example: Job Marketplace. Stored embeddings can be combined in real-time in order to optimize between favoring frequency or recency of consumed content.

BLL. Accounts simultaneously for both frequency and recency. Browsing behaviour for a reference vector is modelled by:

\[ BLL_{u,j} = \ln \left( \sum_{i=2}^{\infty} (TS_{re,f} - TS_{j,i})^{-d} \right) \]

where \( d \) represents the time-dependent decay parameter.

**CUSTOMIZATION**

There is no one-size-fits-all solution. Domain-specific algorithm configurations need to be set up and evaluated accordingly.

**SCALABILITY**

Load Test. A threadpool simultaneously requests recommendations. If we scale-up the number of processing nodes (e.g., from 1 to 4), we can guarantee a much higher number of recommendation requests per second while maintaining the desired runtime performance.