Resource: A Benchmark Library for Neural Team Formation

Arman Dashti  
University of Windsor  
vaghehd@uwindsor.ca

Dhwani Patel  
University of Windsor  
patel891@uwindsor.ca

Karan Saxena  
University of Windsor  
saxena7@uwindsor.ca

Hossein Fani  
University of Windsor  
hfani@uwindsor.ca

ABSTRACT
We contribute OpeNTF, an open-source python-based benchmark library to support neural team formation research. Team formation falls under social information retrieval (Social IR), where the right group of experts should be retrieved to solve a task, which is intractable due to the vast pool of feasible candidates with diverse skills. Even though neural networks could successfully address efficiency while maintaining efficacy, they lack standard implementation and experimental details, which calls for excessive efforts in repeating or reproducing the results in new domains. OpeNTF provides a standard and reproducible platform for neural team formation. It incorporates a host of canonical neural models along with three large-scale training datasets from varying domains. Leveraging an object-oriented structure, OpeNTF readily accommodates the addition of new neural models and training datasets. The first of its kind in neural team formation, OpeNTF also offers negative sampling heuristics that can be seamlessly integrated during model training to boost efficiency and to improve the effectiveness of inference.

1 INTRODUCTION
Collaborative team formation aims at forming teams of experts whose combined skills, applied in coordinated ways, can accomplish difficult tasks such as a research project on ‘machine learning’ whose success can be measured by publications, or the next blockbuster ‘sci-fi’ movie with a touch of ‘drama’. Team formation can be seen as social information retrieval (Social IR), where the right group of experts, rather than the right documents, are required to accomplish the task at hand [11, 12].

Not unexpectedly, a worldwide network of experts to draw upon, each with their own specific aptitudes, interests, and skills, along with the vast space of possible combinations, can overwhelm the scalability of any algorithmic (rule-based) approach to team formation problem; be it multi-objective optimization in Operation Research [6–8] or subgraph optimization in collaborative social network analysis [9, 13, 15, 26]. To bring efficiency while maintaining efficacy, statistical machine learning approaches have been proposed to learn relationships of experts and skills in the context of teams through an iterative and online learning procedure on all past instances of successful teams [21, 22, 25].

Although there has been an increase in team formation research [3–5, 15], each comes with a domain-specific method and a dataset. Researchers have to spend a substantial amount of time to preprocess the data into a version that could be readily fed into the algorithm of choice. Also, proposed methods are case-specific with no standard implementation and are incapable of accommodating different use-cases, let alone the codebases and details are scarcely publicly available. Specifically, existing systems like the recent PyTFL [23], 1) lack efficient preprocessing of large-scale datasets, 2) cannot be easily customized or extended to new methods, and 3) are not tailored for experiments on new datasets from other domains.

In this paper, we contribute OpeNTF, an open-source, extendable, scalable, and standard benchmark library, to support 1) neural methods in team formation research, 2) that can be trained on large-scale datasets from a variety of domains, and 3) evaluated fairly using information retrieval and classification metrics. For the ease of extensibility, OpeNTF defines an abstract class for teams that can be realized through inheritance, be it a team of researchers in a scientific project, a team of cast and crews in a movie, or a team of inventors in a patent. OpeNTF also defines an abstract class for neural team formation models, built upon pytorch1, that can easily accommodate the addition of new neural models through inheritance. For scalability, OpeNTF employs parallel execution at the data preprocessing step, and gpu-acceleration for model training, validation, and test which is the standard practice in machine learning research yet overlooked in the task of neural team formation to date. For a fair benchmark, OpeNTF provides a one-click pipeline that orchestrates the standard flow of machine learning benchmark with no human in the loop. The pipeline accepts a team formation model and brings it through the cross-fold train-validation stage followed by the test and evaluation on an unseen test set, be it a multilayer feed-forward non-Bayesian model or a variational Bayesian model. More notably, OpeNTF features three negative sampling heuristics that can be plugged in to increase the efficiency of neural models during training while improving inference effectiveness, the first of its kind in the neural team formation research.

Contribution. Like similar efforts such as OpenMatch [17] that offers an extensible platform for the design, comparison and sharing of neural information retrieval models, OpeNTF offers a benchmark platform but for neural team formation. It targets the information retrieval and recommender system research communities to propose new team formation solutions and evaluate their effectiveness in a reproducible benchmark.

1pytorch.org
platform, eschewing the arduous labor in baseline and evaluation pipeline reimplementation. Also, having regard to the comparative results, organizations and practitioners can readily apply OpeNTF’s methods to form collaborative teams of experts whose success is almost surely guaranteed.

In contrast to publicly available systems, like the recent PyTFL [23], that faces significant shortcomings especially when scalability, reproducibility, and extensibility are of prime concerns, OpeNTF (1) employs parallel preprocessing of large-scale datasets into a sparse or dense vector representation of skills and experts in teams; (2) embodies built-in neural models that are flexible to customization; (3) smoothly incorporates a new neural model; (4) implements the standard pipeline for training, testing, and evaluating predicted experts for the required input skills for teams in a host of information retrieval and classification metrics, and (5) efficiently generates the statistical characteristics of datasets from different domains, e.g., computer science publications (dblp.v12[27]), movies (imdb[1]), and patents (uspt[2]), that allows to study whether the models’ performances are robust on datasets with a diverse statistical distribution of skills and/or experts in teams. To top it all off, (6) OpeNTF incorporates virtually unsuccessful teams in the absence of explicit unsuccessful teams (e.g., rejected papers) using negative sampling heuristics including uniform, unigram, and smoothed unigram in training minibatches, unigram.b, that can be seamlessly integrated during neural model training to boost efficiency and to improve the effectiveness of inference.

The codebase along with the installation instructions and case studies on dblp.v12, imdb, and uspt on 16+1 neural baselines can be obtained under cc-by-nc-sa-4.0 license at: github.com/fnnl-lab/opentf.

2 NEURAL TEAM FORMATION

Given a set of skills $\mathcal{S}$ and a set of experts $\mathcal{E}$, a team can be abstractly defined as a tuple $(s, e)$ including the non-empty subsets of skills $s \subseteq \mathcal{S}$ and experts $e \subseteq \mathcal{E}$. Examples of successful teams include published research papers consisting of authors as the experts and fields of study (keywords) as

```python
# src/main.py
def create_evaluation_splits(n_sample, n_folds, ...): ...
def run(data_list, domain_list, filter, model_list, output, ...):
    datasets = {}
    models = {}
    if 'dblp' in domain: datasets['dblp'] = Publication
    # other datasets
    if 'fnn' in model: models['fnn'] = Fnn()
    # other models
    for (d_name, d_cls), (m_name, m_obj) in product(datasets.items(), models.items()):
        vecs, ... = d_cls.generate_sparse_vectors(datapath, ...)
        splits = create_evaluation_splits(vecs["id"].shape[0], ...)
        if m_name.find('emb') > 0:
            t2v = Team2Vec(vecs, 'skill', ...)
            t2v.train(emb_setting["id"], emb_setting["w"], ...)
            vecs['skill'] = t2v.dv()
        m_obj.run(splits, vecs, ...)
```

Figure 1: The entry point to OpeNTF’s pipeline.

Figure 2: Data preprocessing speedup by 2.5x using parallel processing on xeon 3.4ghz with 12 cores and 64gb memory.

Figure 3: Efficient calculation of the distribution of experts in teams using teams sparse matrix.

the skills, blockbuster movies consisting of its cast and crew such as actors and directors as the experts and the genres as the skills, or issued patents consisting of its inventors as the experts and categories (classes) as the skills. Nonetheless, what constitutes experts and skills of a team along with its success or failure can be easily overriden in OpeNTF. For instance, success can be redefined based on current number of citations for a research paper, critical acclaims for a movie, and commercialization for a patent.

In neural team formation, given an input subset of skills $s$ we aim at identifying an optimal subset of experts $e$ such that their collaboration is almost surely successful, having regard to the training instances of all previous successful teams. More concretely, a neural model is an estimator for a mapping function $f$ from a subset of skills to a subset of experts, i.e., $f(s) = e$. We refer readers to [22, 23] for further in-depth formal definitions of neural models.

3 SYSTEM OVERVIEW

OpeNTF is designed in a modular way with reproducibility, extensibility, and scalability in mind. It includes a single one-click pipeline that engages three primary components: 1) teams sparse matrix representation, 2) neural model training, and 3) performance evaluation. Figure 1 shows the entry point to the pipeline (.src/main.py). As seen, the pipeline creates a list of dataset classes (e.g., Publication for dblp) and a list of neural models (e.g., Fnn()) for the feed-forward non-Bayesian model) and executes a benchmark on all neural models over all datasets by calling overridden methods to form collaborative teams of experts whose success is almost surely guaranteed.

<table>
<thead>
<tr>
<th>Team</th>
<th>#teams (papers in dblp.v12)</th>
<th>#teams</th>
<th>#members</th>
<th>#members</th>
</tr>
</thead>
<tbody>
<tr>
<td>dblp.v12</td>
<td>10^5</td>
<td>10^3</td>
<td>10^2</td>
<td>10^1</td>
</tr>
<tr>
<td>imdb</td>
<td>10^4</td>
<td>10^3</td>
<td>10^2</td>
<td>10^1</td>
</tr>
</tbody>
</table>

2 NEURAL TEAM FORMATION

Given a set of skills $\mathcal{S}$ and a set of experts $\mathcal{E}$, a team can be abstractly defined as a tuple $(s, e)$ including the non-empty subsets of skills $s \subseteq \mathcal{S}$ and experts $e \subseteq \mathcal{E}$. Examples of successful teams include published research papers consisting of authors as the experts and fields of study (keywords) as
expert members are represented in sparse occurrence vectors (one-hot encoded). Teams’ skills and members can be also represented in dense vectors through OpenNTF’s adoption of paragraph vector [16] (Team2Vec). Preprocessed data, as either sparse or dense matrix, is randomly split into a train-validation set and a test set. The train-validation set is further split into folds (create_evaluation_splits()). Next, a neural model is trained and validated on each fold followed by the evaluation on the test set using information retrieval and classification metrics (run()). In the following subsections, we detail each component.

### 3.1 Teams Sparse Matrix Representation

OpenNTF transforms training sets from different domains into a uniform data structure that neural models can easily consume. As per Figure 4, it defines an abstract class Team that builds the sparse vector representation of a team instance having regard to two main properties of a team: skills and expert members, alleviating the discrepancies in the underlying datasets from different domains. Each instance of a team is transformed into an occurrence vector of skills and members independently in parallel(get_one_hot()). OpenNTF uses bucketing to trade-off the performance gain by parallel stacking of vectors and multiprocessing overhead. Figure 2 shows the speedup when using parallel preprocessing for dblp.v12. From the figure, while elapsed time for loading the raw data remains relatively low for an increasing number of teams, it linearly grows for creating the sparse vector representation which is reduced via parallel processes.

Teams sparse matrix representation further facilitates efficient analysis of statistical characteristics of the datasets by calculating the distributions on the matrix using linear algebra, as opposed to enumerating raw data, followed by visualization. Table 1 shows that our library efficiently calculates point statistics and distributions with visualization. Knowing the underlying statistics of the training datasets is key to the choice of the neural model, as highlighted in [22]. For example, Figure 3 depicts the distribution of teams over experts for dblp.v12 and imdb. Both datasets suffer from long tail distributions; a few experts have participated in many teams, whereas the majority have participated sparingly.

<table>
<thead>
<tr>
<th>statistics</th>
<th>distributions</th>
</tr>
</thead>
<tbody>
<tr>
<td>#teams</td>
<td>#teams per each skill</td>
</tr>
<tr>
<td>#experts</td>
<td>#teams per each expert</td>
</tr>
</tbody>
</table>
| #skills | #skills having 1 team, 2 teams ...
| #teams w/ one skill | #skills having 1 team, 2 teams ...
| #teams w/ one expert | #skills having 1 skill, 2 skills ...
| average #skills per team | #teams having 1 expert, 2 experts ...
| average #skills per expert | #teams time (seconds) |
| average #teams per expert | uspt 7,068,508 157.1706 |
|                  | dblp.v12 4,877,383 177.8351 |
|                  | imdb 507,034 28.3453 |

#### 3.2 Neural Model Training

OpenNTF provides an abstraction on neural team formation models (Ntf), inherited from pytorch’s nn.Module. From Figure 5, Ntf provides a single interface for major steps of neural model benchmark including training, test, and evaluation. Researchers and practitioners can instantiate the built-in neural models or seamlessly plug in custom ones, being liberated from rebuilding the benchmark stages.

Currently, OpenNTF includes two reference neural architectures: *i*) feed-forward neural network (Fnn) and *ii*) the state-of-the-art variational Bayesian neural network (Bnn) in neural team formation [22]. The variational Bayesian model inherits from Fnn and reuses the implementation of the test stage and the negative sampling heuristics, e.g., ns_uniform(), while overriding the learning stage. Both neural networks are designed to accept sparse or dense vector representations of skills in the input layer and encode them into subsets of experts through one or several hidden layers of different sizes. The neural model’s architecture and hyperparameters can be dynamically set (.//src/param.py) to perform a wide range of ablation studies programatically. For the minimum level of comparison baseline, OpenNTF also includes a random model (Rnd) that blindly assigns a random subset of experts to the input subset of skills. Neural models benefit from two additional features, as explained hereafter.

#### 3.2.1 Dense Vector Representation

Following Rad et al. [22], OpenNTF incorporates learning dense vector representations for the input subsets of skills and output subsets of experts. Inspired by paragraph vectors of Le and Mikolov [16], we consider a team as a document and its skills and expert members as the document’s words. We developed Team2Vec class (.//src/ml/team2vec.py) to learn dense vector representations of subsets of skills and experts in the same or disjoint embedding spaces (embtypes='skill', 'member', 'joint'). We employ the distributed memory model to generate the real-valued embeddings using gensim[24]. Figure 1 shows that
neural models are able to easily employ dense vectors for skill in the input and expert members in the output layers.

3.2.2 Negative Sampling Heuristics. Leveraging negative samples conveys complementary signals to a neural model and improves accuracy, best shown in social network analysis, language modeling, and recommender systems. [14, 18, 20, 28]. However, most real-world training datasets in the team formation domain do not have unsuccessful teams explicitly (e.g., collection of rejected papers). In the absence of unsuccessful training samples, OpeNTF incorporates three negative sampling heuristics based on the closed-world assumption where no currently known successful subset of experts for the input skills is assumed unsuccessful:

- **uniform**, where subsets of experts are randomly chosen from the uniform distribution over all subsets of experts as unsuccessful teams.
- **unigram**, where subsets of experts are chosen regarding their frequency in the training set. Intuitively, teams of experts that have collaborated more often will be given higher probabilities and chosen more frequently as negative samples to dampen the effect of popularity bias.
- **unigram_b**, where we employed the Laplace smoothing when computing the unigram distribution of the experts but in each training minibatch.

In sum, OpeNTF is ready to benchmark 16+1 baselines: \{Fnn, Bnn\} × \{sparse, dense\} × \{none, uniform, unigram, unigram_b\} + random.

3.3 Performance Evaluation

The evaluation methodology of OpeNTF is based on n-fold cross-validation at the model training-validation stage followed by a test stage. The set of teams (technically, the rows) of the teams sparse matrix is randomly split into a test set (15% by default) and a train-validation set. The train-validation set is further split into n-folds for model training and validation that results in one trained model per fold. The train-validation folds and test set are generated by the OpeNTF’s pipeline and equally fed into all baseline models for a fair evaluation comparison. Given a team from the test set, OpeNTF compares the ranked list of a predicted subset of experts by the model of each fold with the observed subset of experts and reports the performance of the trained model on each fold as well as the average in all folds. It reports the information retrieval metrics including normalized discounted cumulative gain (ndcg) and mean average precision (map) at top-k as well as classification metrics including precision and recall at top-k and area under the receiver operating characteristic (rocauc) using pytrec_eval [10] and scikit-learn [19] libraries.

In order to evaluate the efficiency of neural models at the training phase (i.e., whether the model converges sooner to the minimum loss) versus its inference efficacy, OpeNTF also evaluates the models’ effectiveness on the test set at each training epoch and report the metrics within the increasing number of epochs. The complete results of neural baselines on dblp.v12 and imdb are accessible at OpeNTF’s codebase.

4 QUICK START

OpeNTF can be obtained by:

```bash
git clone https://github.com/fani-lab/opentf.git
```

It accepts paths to raw data, domain names, and neural model names and benchmarks all the models on all the datasets based on hyperparameters in ./src/param.py without human in the loop until final delivery of trained models and evaluation metrics in the output path:

```
$ python -u main.py -data toy.dblp.v12.json \
        -domain dblp imdb uspt \n        -model random fnn fnn_emb bnn bnn_emb
```

An example run of our library is:

```bash
cd ./opentf/src
python -u main.py -data toy.dblp.v12.json \n    -domain dblp imdb uspt \n    -model random fnn fnn_emb bnn bnn_emb
```

5 CONCLUSION AND FUTURE WORK

We presented OpeNTF, the first open-source python-based benchmark library for neural team formation research. OpeNTF features i) end-to-end reproducible pipeline with standard experimental methodology, ii) abstractions for training datasets and neural models for ease of extensibility along with reference implementation of the state-of-the-art neural models, iii) scalable preprocessing and efficient statistical analysis of large-scale datasets, and notably iv) three negative sampling heuristics to boost neural models’ efficiency and efficacy at training and testing stages, respectively. Currently, OpeNTF is being extended to incorporate human and non-human factors in team formation, such as scheduling preferences (temporal team formation), social aspects (‘social fit’ for a team), diversity (not only varying skills, but different institutions, countries, and education), and fairness in team formation [5].