# Correlated data generation using GAN and its Application for Skill recommendation

Shreyas Patel[1], Ashutosh Kakadiya[1], Maitrey Mehta[1], Raj Derasari[1], Rahul Patel[1,2], and Ratnik Gandhi[1]

[1]School of Engineering and Applied Science, Ahmedabad University
[2]Logistixian Pvt. Ltd.
{shreyas.p,ashutosh.k,maitrey.m,raj.d}.btechi14@ahduni.edu.in, rpatel.1594@gmail.com,ratnik.gandhi@ahduni.edu.in

**Abstract.** Generative Adversarial Networks (GAN) have shown great promise in tasks like synthetic image generation, image inpainting, style transfer, and anomaly detection. However, generating discrete data is a challenge. This work presents an adversarial training based correlated discrete data (CDD) generation model. It also details an approach for conditional CDD generation. The results of our approach are presented over two datasets; job-seeking candidates skill set (private dataset) and MNIST (public dataset). From quantitative and qualitative analysis of these results, we show that our model performs better as it leverages inherent correlation in the data, than an existing model that overlooks correlation.

**Keywords:** Generative Adversarial Network, Autoencoder, Correlational Neural Networks

## 1 Introduction

Recent advances in computational power have offered a significant boost to learning techniques, fueling applications of Artificial Intelligence(AI) in varied domains – disease diagnoses [1], predicting bank failures [2] and evaluating faults in sports movements [3], etc. Another boost has come from advances due to novel architectures and training techniques: Deep Neural Networks (DNNs), Convolutional Neural Networks (CNNs), Generative Models, Autoencoders amongst many.

One key requirement for supervised learning technique in AI is availability of sufficiently big, labeled, and clean dataset. The availability of such dataset, along with the training technique and computing power, form the basis of an effective AI agent. However, in many real-world scenarios, there is a lack of task-specific datasets (e.g. due to privacy, access restrictions, licenses, copyright, patents and access fees, some of the following datasets are hard to find - medical health-care, criminal background, state policies, employee profiles) or the available data may not be clean (e.g. missing values or corrupted values). Under such scenarios, building human-level AI agents (models) is difficult.

Recent papers on generative models[1] attempt to address one of these issues of lack of data. Generative Adversarial Networks (GANs) [5], a type of generative model, help us mitigate the issue of data unavailability by synthesizing new realistic samples. Following this work, there have been several attempts in the direction of generating image/continuous data [5,6]. Efforts have also been made to make these synthetic samples look more realistic [6]. A similar set of early efforts towards generating text/discrete dataset has been reported in [7,8,9,10]. These efforts can be sub categorized into generating text tokens [7] and generating realistic natural language [8,9,10]. It must be noted here that the latter problem is harder as it requires generative models to capture semantic structures of underlying sentences [11]. In this paper, our focus is on addressing the former problem, i.e., of generating text tokens/discrete data.

We approach this problem considering a scenario in which our aim is to build a text based career counseling AI agent in the domain of employment related services. The most basic ability required by such an agent is as follows:

- Suggest the best profession to an individual based on their existing skills.
- Suggest a new set of skills to acquire, considering career preferences and existing skills.

If we want to accomplish these tasks using an AI agent, then the problem essentially translates to conditional discrete data generation. In other words our aim is to generate skill set conditioned on either profession, or existing skills, or both. However, in absence of open-source candidate profile data, building such models (AI agents) for career progression counseling is difficult. In this case, we can resort to some models such as generative models (specifically, Generative Adversarial Networks (GANs)[5]) or Variational Autoencoder (VAE)[12] that can help generate the required synthetic data.

In this work, we model skills from a candidate's profile as a binary vector (similar to the model presented in [13]), and with the help of GANs, generate synthetic skills. An important difference between our work and [13] is that our model plays the minimax optimization game over conditional input ($D(x|y)$) vs. [13] optimizing only over input ($D(x)$), where $D(\cdot)$ is discriminator output, $x$ is a skill vector, and $y$ is a profession vector. Also, we use Correlational Neural Network [14] to better capture the relation between a profession and skill set, and among skills of a particular skill set. It must be noted that generating realistic sentences of a language is a difficult task, especially due to grammar and semantics of a sentence. This not only focuses on the right set of words but also the right order and context. In this work, we focus on the problem of generating correlated discrete data without worrying about the order between them.

Recently, a combination of Recurrent Neural Netowrks (RNNs) and Deep Q Networks [15] has been used to generate text that use Deep Reinforcement

[1] Generative models [4] are a semi-supervised approach meant for developing models that provide an effective generalization to a large unlabeled data set on the basis of a small labeled dataset. They find use in wide ranges of applications like image search, genomics and natural language parsing, where data is largely unannotated.

Learning [16]. As used in seqGAN [8] and maskGAN [17], policy networks first applies a policy algorithm (such as Monte Carlo search in seqGAN or an actor-critic model in maskGAN) and then offers rewards to players for optimization (instead of penalties). This approach is not very effective when purpose is to generate discrete text tokens - that do not appreciate positional context as set of tokens in a sentence do. In our work, an output is a prediction of discrete data that does not have sequence or positional context. For these reasons, methods based on Deep Q Networks and Policy Networks are not relevant to the problem addressed in this paper.

To the best of our knowledge, this is amongst the early attempts to preserve correlation in generated data and is scalable to datasets which are discrete. The rest of this paper is organized as follows: Section 2 presents our model and algorithms for generating conditional skills for an input candidate profile. Appendix A presents challenges for generating initial data for training our generative model while subsection 3.1 details our experiments. We present our concluding remarks in section 4 with potential line of future work.

## 2 CorrGAN Model

In this section, we present our model and an algorithm for generating correlated discrete (binary) data using adversarial training. The model comprises of primarily two parts; Correlational Neural Network(CorrNN) [14] and Generative Adversarial Network (GAN) [5]. GAN face difficulty in generating discrete data as it is difficult to pass the gradient update from discriminator $D$ to generator $G$. To mitigate this challenge, we make use of CorrNN, that essentially learns to project discrete data into some continuous latent space.

CorrNN is an Autoencoder based approach for Common Representation Learning. It takes into consideration the correlation in input data while regenerating the same. After training CorrNN, its encoder network ($Enc$) learns to map discrete input space $\mathbb{Z}_+^{|c|}$ to continuous latent space $\mathbb{R}^h$ and decoder network ($Dec$) learns to reconstruct the input space from latent space, which is continuous.

Next, we train the GAN, where $G$ learns to map random seed to continuous latent space. The dimensionality of this latent space is same as that of the latent space in CorrNN. Hence, the output produced by the $G$, acts as an input to the Decoder network of CorrNN. The Decoder network then outputs a vector which is passed as a synthetic input to train $D$. Simultaneously, $D$ is also shown real discrete vectors from the input data, which helps it to learn discriminating whether an input sample is real or synthetic. The gradients are then passed back from $D$ to Decoder of CorrNN and from Decoder to $G$.

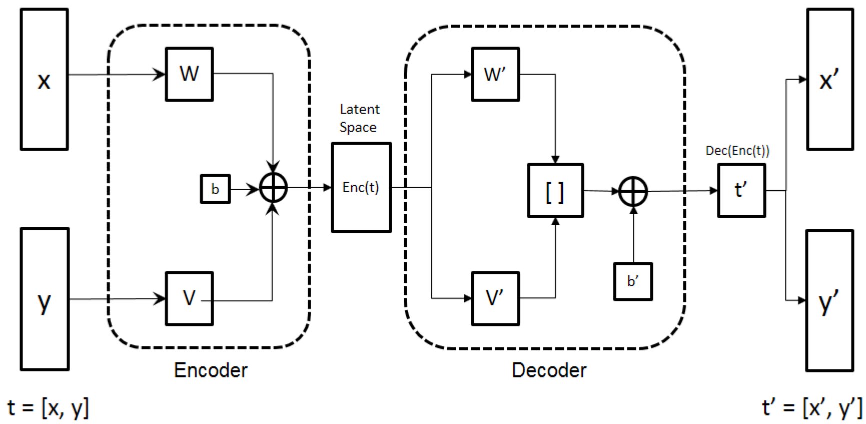


Fig. 1: Block diagram of Correlational Neural Network

Let $T$ be a set of $n$ binary correlated vectors $T = \{t_i \mid 1 < i < n\}, i \in \mathbb{Z}_+$. Input $t_i$ is divided into two mutually exclusive subsets such that,

$$t_i = [x_i, y_i]. \tag{1}$$

The Encoder network takes $x_i$ and $y_i$ as an input and map it to a continuous latent space.

$$Enc(t_i; \theta_{enc}) = f(Wx_i \, + \, Vy_i \, + \, b) \tag{2}$$

where, $\theta_{enc} = [W,\, V,\, b]$ are hidden layer learning parameters for $Enc$ and $f$ is a non-linear activation function. The $Dec$ network takes $Enc(t_i; \theta_{enc})$ as an input and regenerates $t_i$ i.e., $t_i^{'}$.

$$Dec(Enc(t_i; \theta_{enc}); \theta_{dec}) = g([W'Enc(t)\ ,\ V'Enc(t)] \, + \, b'), \tag{3}$$

$$t_i^{'} = Dec(Enc(t_i; \theta_{enc}); \theta_{dec}). \tag{4}$$

Similarly, based on equations (1) and (4),

$$x_i^{'} = Dec(Enc([x_i, 0]; \theta_{enc}); \theta_{dec}) \tag{5}$$

$$y_i^{'} = Dec(Enc([0, y_i]; \theta_{enc}); \theta_{dec}) \tag{6}$$

where, $\theta_{dec} = [W',\, V',\, b']$ are hidden layer leaning parameters for $Dec$, and $g$ is non-linear activation function. Subsequently, we define loss function, $\mathcal{J}_T(\theta)$, of the CorrNN as:

$$\mathcal{J}_T(\theta) = \frac{1}{m}\sum_{i=1}^{m}(L(t_i, t_i') \, + \, L(t_i, x_i') \, + \, L(t_i, y_i')) \, - \, corr(Enc(x), Enc(y)). \tag{7}$$

$\mathcal{J}_T(\theta)$ minimizes self-reconstruction error and cross-reconstruction error. It also maximizes correlation between the hidden representations of both the halves of the input data.

$$corr(Enc(x), Enc(y)) = \frac{\sum_1^m (Enc(x_i) - \overline{Enc(x)})(Enc(y_i) - \overline{Enc(y)})}{\sqrt{\sum_1^m (Enc(x_i) - \overline{Enc(x)})^2 \sum_1^m (Enc(y_i) - \overline{Enc(y)})^2}} \tag{8}$$

$L$ is the function for calculating reconstruction error; $m$ is the number of training samples in a batch; $Enc(x_i)$ and $Enc(y_i)$ are hidden layer representation of $x_i$ and $y_i$ respectively; $\overline{Enc(x)}$ and $\overline{Enc(y)}$ are mean vectors of every $Enc(x_i)$ and $Enc(y_i)$ respectively.

The $Dec$ is then used to map the continuous output of $G(z; \theta_g)$ to discrete output $Dec(G(z; \theta_g); \theta_{dec})$, where $z$ is the low dimensional random noise vector and $\theta_g$ are the learning parameters of G. $D(\cdot; \theta_d)$ then predicts whether the generated discrete output is real or synthetic, where $\cdot$ can be any input and $\theta_d$ are the learning parameters of $D$.

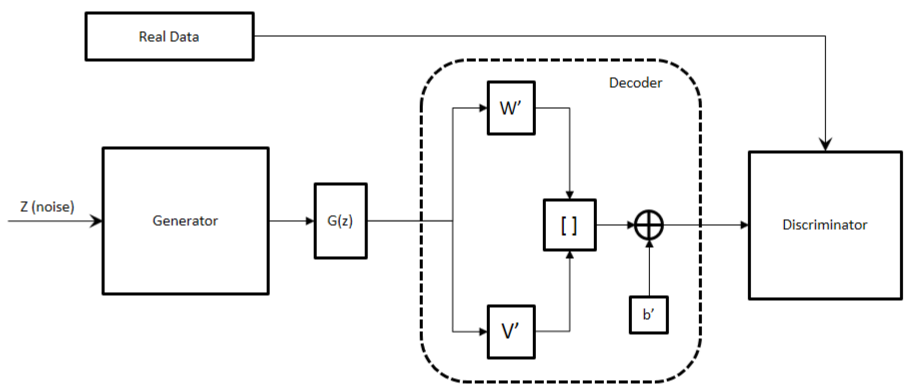


Fig. 2: Block diagram GAN training module.

The objective function for the GAN model is:

$$\min_G \max_D V(G, D) = \mathbb{E}_{t \sim p_{data}(t)}[\log D(t)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(t_z))] \tag{9}$$

The update functions for $G$ and $D$ are as follows:

$$\theta_d \leftarrow \theta_d + \alpha \nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^m \log D(t_i) + \log(1 - D(t_{z_i})) \tag{10}$$

$$\theta_{g,dec} \leftarrow \theta_{g,dec} + \alpha \nabla_{\theta_{g,dec}} \frac{1}{m} \sum_{i=1}^m \log D(t_{z_i}) \tag{11}$$

where $t_{z_i} = Dec(G(z_i))$, $\theta_d$ and $\theta_g$ are learning parameters for $D$ and $G$ respectively, $\alpha$ is learning rate, $m$ is the number of training samples in the mini-batch, $t_i$ is original training input sample, and $t_{z_i}$ is synthetic sample generated with help of $G$ and $Dec$.

To generate conditional discrete data, we follow the technique in [18] i.e., append the apriori information $y$ to the noise prior $z$, hence $t_z = Dec(G([z, y]))$.

**Algorithm 1:** Algorithm for conditional CorrGAN

```
θ_d, θ_g, θ_enc, θ_dec ← Initialize with random values.
α ← learning rate.
m ← number of training samples.
while pre-training epochs do
    Randomly sample (x_1, y_1), (x_2, y_2), ..., (x_m, y_m).
    Update θ_enc, θ_dec by minimizing Eq. (7) where t_i = [x_i, y_i].
end
while training epochs do
    for d-steps do
        //Update discriminator:
        Randomly sample z_1, z_2, ..., z_m ∈ p_z.
        Randomly sample t_1 = [x_1, y_1], t_2 = [x_2, y_2], ..., t_m = [x_m, y_m].
        t_{z_i} ← Dec(G([z_i, y_i])).
        t̄_z ← (1/m) Σ_1^m t_{z_i}.
        t̄ ← (1/m) Σ_1^m t_i.
        θ_d ← θ_d + αΔ_{θ_d} (1/m) Σ_1^m log D(t_i, t̄) + log(1 − D(t_{z_i}, t̄_z)).
    end
    //Update generator and decoder:
    Randomly sample z_1, z_2, ..., z_m ∈ p_z.
    Randomly sample y_1, y_2, ..., y_m from training data.
    t_{z_i} ← Dec(G([z_i, y_i])).
    t̄_z ← (1/m) Σ_1^m t_{z_i}.
    θ_{g,dec} ← θ_{g,dec} + αΔ_{θ_{g,dec}} (1/m) Σ_1^m log D(t_{z_i}, t̄_z).
end
```

The training process of CorrGAN is demonstrated in Algorithm 1. During training, the first step is pre-training of the CorrNN, where it minimizes $\mathcal{J}_{\mathcal{T}}(\theta)$ according to Equation 7. The second step is training $G$ and $D$, where we use mini-batch averaging and eventually, $G$ and $D$ converge.

## 3 Experimental Setting and Results

In this section, we present results of Algorithm 1. The experiments were carried out on a machine with configuration (Intel i7-4770, 8GB RAM, Intel HD 4600, Ubuntu 64-bit) environment.

### 3.1 Results

As mentioned earlier, we validate our model on two datasets- MNIST [19] and skill dataset. We choose MNIST due to its conformity as a standard dataset for machine learning applications and it provides an inherent correlation which suits our application. The images generated are not conditioned and are generated by giving noise as the input to the generator. The experiment aims to warrant the usage of CorrNN instead of Vanilla Autoencoder for translation of discrete data into continuous space. Figure 3 shows the comparative results between medGAN [13] (which uses Vanilla Autoencoder) and CorrGAN for MNIST dataset on different epochs.

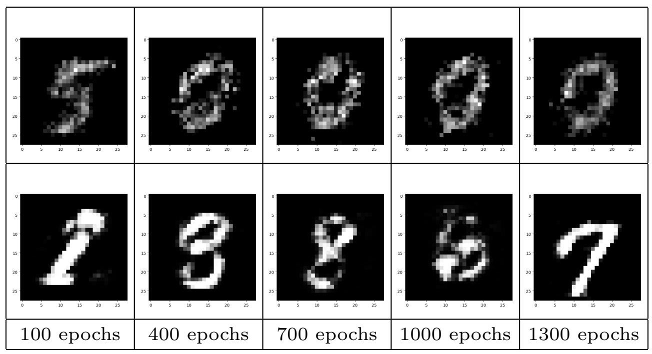


Fig. 3: Output samples - medGAN(top) and CorrGAN(bottom) for given epochs

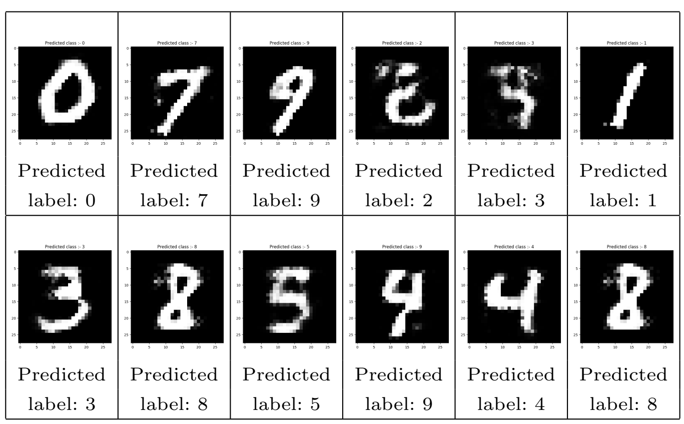


Fig. 4: CorrGAN-MNIST samples and predicted classes

In our proposed model, we exploit the inherent correlation between pixels of an image, to generate conditional data. Considering the dimensions of an image, we append the bottom half of an image to a noise vector of low dimensionality and reconstruct the upper half. A latent space is generated which is passed as input to the Decoder, which produces a complete image. Some results are shown in Figure 4. We next perform the experiment on candidate skill dataset. Due to scarcity of clean data, appropriate pre-processing was undertaken [Appendix A]. Table 1 gives details pertaining to the skill data at hand. Generated results conditioned on a given profession are described in Table 2.

| Description | Statistic |
|---|---|
| # of job titles | 6 |
| # of candidates | 9686 |
| # of skills | 7055 |
| | {Avg, Max, Min} |
| Profiles per Job Title | {1614, 2248, 460} |
| Skills per Candidate | {6, 108, 1} |

(a) Dataset statistics (Avg, Max, Min)

| Description | Statistic |
|---|---|
| Training Time | 15 Hrs. |
| # Pre-Training Epochs | 150 |
| # Epochs | 1000 |
| Batch Size | 100 |

(b) Model training statistics

Table 1: Training data statistics for input skill-set data, after data preprocessing

| A prior input: Profession | Generated Skills |
|---|---|
| Web designer | ms sql server, **oracle**, database, **sql, java, .net** |
| | **mysql, javascript, c++, jira, html** |
| | **javascript**, android, **php, jquery, java** |
| Net developer | **xml**, illustrator, **css, html** |
| | graphic design, illustrator, photoshop, **html** |
| | hibernate, **xml**, spring, jdbc, servlets, **jquery**, struts, **ajax**, j2ee, servlet, **html**, jsf, jsp, **java** |
| Java developer | **oracle, eclipse**, jdbc, j2ee, **sql**, jsp, **java** |
| | **javascript**, asp.net, **c#**, asp, **jquery**, ms asp, **.net** |
| | **javascript**, j2ee, **css, html,** jsp, **java** |
| Application developer | asp.net, **c#**, asp, ms asp, **.net** |
| | **unix, mysql**, wordpress, vb.net, programming, **xml**, rest, **javascript**, linq, scrum, **python**, soap, **c#, php,** css3, **eclipse**, visual studio, ibm rad, sublime, scss, **jquery, agile, ajax, pl/sql,** iis, **jira, json, sql,** angular js, **html**, windows, **java** |
| | **mysql, xml,** ms sql server, **javascript, php,** sql server, **c++, json, sql, html, java** |
| Application support analyst | asp.net, **c#,** asp, database, ms asp, iis, **sql, .net** |

Table 2: We train CorrGAN once, on (profession,skill) tuples. Given a one-hot profession input, we can generate skill profiles for the given profession. Some skills are common across all professions in the dataset, which are shown in bold.

As a simple use case of synthetic results for a career counseling AI agent, we can consider a candidate's current profession and compute a set difference between his/her current skills and identify skills from synthetic data to suggest to the candidate as skills that they should acquire for career progression. Note that we have tested the model for two given data sets, i.e, user skill vectors and MNIST. However, this model can be used for inherently correlated data.

### 3.2 Evaluation Metrics

To evaluate images generated by our model, we pass them through the CNN MNIST classifier using TensorFlow [20] which is known to have 99.51% accuracy in classifying the MNIST dataset. We find 69% of SVM labeled images to be matching perceptual estimates [2]. Some of the results are shown in Figure 4. For the skill dataset, we use scatter plots to compare probabilities of occurrence of skills in the input and generated (synthetic) data. We calculate the mean-squared error of scatter points with respect to the line $y$=$x$ to outline a comparison between CorrGAN and medGAN. Furthermore, we define a correlation metric, to evaluate the joint occurrences of skills in the generated data as compared to that of the input data.

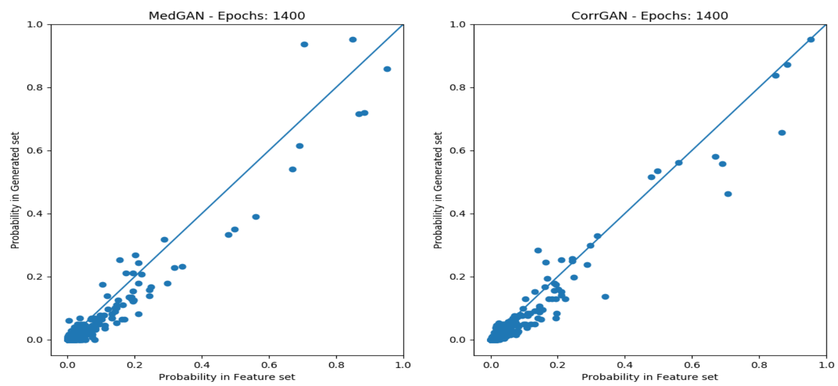


Fig. 5: The graphs capture the probabilities of occurrence of each skill in the training set (for a given profession, Application Developer) versus that in the generated set, for the two models (medGAN, CorrGAN). Each point corresponds to a skill in our database. Ideally, in order to replicate the characteristic of the feature set, the points should fall on the line $y$=$x$ implying equal probability in either set.

In Figure 5, the probability measure of skills' occurrence in the input set to the generated set is plotted. The mean square-error(MSE) is calculated to account the variation of scatter points from the line $y$=$x$. Consequently, the MSE estimate for medGAN is $7.567 \times 10^{-5}$ and corrGAN is $5.049 \times 10^{-5}$ (at 1400 epochs). However, Figure 5 demonstrates quantification of the occurrence of a

[2] Majority vote by authors on 100 SVM samples

skill, rather than considering correlation among two skill points, which motivates us to use a different evaluation metric, to quantify the joint occurrence of skills.

**Algorithm 2:** Build correlation matrix of given skill vector

```
Let vectors ← no of samples/profiles
Let dim ← no of skills
inputMatrix[vectors][dim] ← Data Matrix
Initialize corMatrix[dim][dim] ← Zero Matrix
Initialize threshold α ← Preset value ∈ [0, 1]
for i=1:dim-1 do
    for j=i+1:dim do
        for k=1:vectors do
            if inputMatrix[k][i]> α and inputMatrix[k][j]> α then
                corMatrix[i][j]+=1.
            else
            end
        end
        corMatrix[j][i]:=corMatrix[i][j].
    end
end
corMatrix:=corMatrix/vectors.
```

To measure the skill coherence, we calculate the correlation between generated skills for the medGAN and CorrGAN models (every 100 epochs), by computing the joint occurrence probability of two skills. Cooccurrence matrices are calculated for both input data and generated data, by Algorithm 2 using respective data matrix.

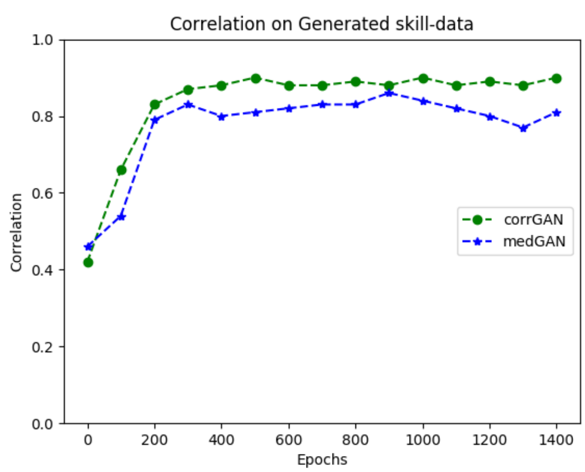


Fig. 6: Correlation of generated data vs epochs. Note that we calculate the correlation matrix for medGAN, CorrGAN and original data by Algorithm 2.

Note that the matrices are symmetric and are *skill vs skill*; with every cell as the intersection of row skill and column skill, being the total co-occurrences of the two skills. The matrices are then normalized so that each value lies in [0,1]. The original data's co-occurrence matrix is subtracted from that of the generated data to calculate the error matrix and the determinant is computed. We plot the correlation in Figure 6. We infer from the plot, that correlation improves in our model over the number of epochs.

## 4 Future work

Candidate data is mostly proprietary content and thus its availability is scarce. The dataset consisted of 24,934 candidate profiles, out of which 6,762 profiles had an empty skill section and hence had to be removed from the dataset. Thus, a module which can extract skills from other sections of profiles like projects and experience can be developed. Further, certain imperfections in cleaning the data - creating valid skill tokens from text in skill section - affect the experiments. An interesting future improvement can be to map the skills to a predefined dictionary of skill by calculating the edit distance between them or using a more robust algorithm such as Fuzzy Matching [21].

As a non-trivial extension, our model can be extended to condition on other parameters, like geographical regions, education background, organization, in a candidate profile.

## 5 Acknowledgement

The authors of this work would like to thank HireValley Pvt. Ltd. for providing candidate profile data, which was pivotal to the experiments conducted.

## References

1. J. H. Holland, *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press, 1992.
2. M. D. Fethi and F. Pasiouras, "Assessing bank efficiency and performance with operational research and artificial intelligence techniques: A survey," *European Journal of Operational Research*, vol. 204, no. 2, pp. 189 – 198, 2010.
3. R. Bartlett, "Artificial intelligence in sports biomechanics: New dawn or false hope?," *Journal of sports science & medicine*, vol. 5, no. 4, p. 474, 2006.
4. D. P. Kingma, S. Mohamed, D. J. Rezende, and M. Welling, "Semi-supervised learning with deep generative models," in *Advances in Neural Information Processing Systems*, pp. 3581–3589, 2014.
5. I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, pp. 2672–2680, 2014.
6. A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb, "Learning from simulated and unsupervised images through adversarial training," *arXiv preprint arXiv:1612.07828*, 2016.

7. T. Che, Y. Li, R. Zhang, R. D. Hjelm, W. Li, Y. Song, and Y. Bengio, "Maximum-likelihood augmented discrete generative adversarial networks," *arXiv preprint arXiv:1702.07983*, 2017.
8. L. Yu, W. Zhang, J. Wang, and Y. Yu, "Seqgan: sequence generative adversarial nets with policy gradient," in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.
9. Y. Kim, K. Zhang, A. M. Rush, Y. LeCun, *et al.*, "Adversarially regularized autoencoders for generating discrete structures," *arXiv preprint arXiv:1706.04223*, 2017.
10. S. Subramanian, S. Rajeswar, F. Dutil, C. Pal, and A. Courville, "Adversarial generation of natural language," *ACL 2017*, p. 241, 2017.
11. I. Goodfellow, "Generative adversarial networks for text," 2016.
12. D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.
13. E. Choi, S. Biswal, B. Malin, J. Duke, W. F. Stewart, and J. Sun, "Generating multi-label discrete electronic health records using generative adversarial networks," *Machine Learning in Health Care (MLHC) 2017*, 2017.
14. S. Chandar, M. M. Khapra, H. Larochelle, and B. Ravindran, "Correlational neural networks," *CoRR*, vol. abs/1504.07225, 2015.
15. H. Guo, "Generating text with deep reinforcement learning," *CoRR*, vol. abs/1510.09202, 2015.
16. K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "A brief survey of deep reinforcement learning," *CoRR*, vol. abs/1708.05866, 2017.
17. W. Fedus, I. Goodfellow, and A. M. Dai, "Maskgan: Better text generation via filling in the _," *arXiv preprint arXiv:1801.07736*, 2018.
18. M. Mirza and S. Osindero, "Conditional generative adversarial nets," *CoRR*, vol. abs/1411.1784, 2014.
19. Y. LeCun and C. Cortes, "MNIST handwritten digit database," 2010.
20. R. Malm, "Mnist image class. tensorflow cnn 99.51% test acc. on kaggle," 2017.
21. S. Chaudhuri, K. Ganjam, V. Ganti, and R. Motwani, "Robust and efficient fuzzy match for online data cleaning," in *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*, SIGMOD '03, (New York, NY, USA), pp. 313–324, ACM, 2003.

## A Data Pre-Processing

We obtained 24,934 candidate profiles in JSON format. A typical candidate profile contains work experience, educational qualification, projects, additional information apart from skill data. However, the experiments presented in this work only require use of skill data and the candidate's current profession. An example of input data containing skills as discrete tokens and profession is given in Table 3.

It must be noted that 8486 candidate profiles (out of 24934) contained narrative in the skill section. Table 4 lists one such profile. Deriving skills from such narrative requires specific data mining approaches. We also note that, as far as skills in the IT sector are concerned, most skills are of length 15 characters or less[3]. Thus, in this work we do not focus on data points with comprehension in skill section and ignore profiles with skill lengths exceeding 15 characters. Note that the threshold 15 is an empirically derived constant, and can be changed without affecting performance of the algorithm presented.

| Profession | Skills |
|---|---|
| Java Developer | Java, J2EE, Servlets, Jsp, JQuery, Spring 2.5, Spring MVC |
| Application Developer | .NET, SQL, ASP .Net, VB.NET, C#, Oracle, WCF |
| Applications Engineer | Java, J2EE, Servlets, Jsp, JQuery, Spring 2.5, Spring MVC, SOAP |
| Application Support Analyst | UNIX, AIX, Solaris, Sun Storage, Sun SPARC, Sun Ultra, HP, VNC |
| Net Developer | C#, SQL, ASP, ASP.NET, MS ASP, MS SQL SERVER, SQL SERVER |
| Java Developer | JAVA, JAVASCRIPT, JSP, JUNIT, HTML, SOAP, XML |

Table 3: Snippet of Input data

| | |
|---|---|
| Net Developer | Gathering and analysis of requirements and delivery of solutions (1 year). High experience level in computer repair, assemble and analysis of requirements for a specific computer (3 years). Operating Windows XP and newer versions, Installation and configuration (3 years), Management of the Linux Operating System, installation, basic configuration and installation of basic programs(Less than 1 year). Basic knowledge of Java, HTML, CSS, C#, JavaScript. (Less than 1 year), Basic knowledge of MySQL DataBase (Less than 1 year), Basic management of Netbeans IDE, VS.NET, Photoshop, Microsoft Office (1 year), Basic knowledge of SQL Server. (1 year) |

Table 4: Candidate profile with comprehension in skill section.

We maintain a superset of skills and professions called *skillDictionary* and *professionDictionary*, created by taking the union of all skills and professions in the input profession-skills pairs respectively. These pairs of profession-skills is converted to pairs of *professionVector*-*skillVector*. The *skillVector* is binary, containing ones and zeros for the skills present and absent respectively in a candidate profile, whereas *professionVector* is one-hot, containing a 1 in an appropriate position, corresponding to a candidate's current profession.

[3] The average length of skills in our data is 11.4173 characters with a standard deviation of ± 4.7876. Hence, we infer that our threshold of 15 is suitable for good results.