Prot-A-GAN: Automatic Protein Function Annotation using GAN-inspired Knowledge Graph Embedding

Bishnu Sarker [1 2] **Marie-Dominique Devignes** [2] **Guy Wolf** [3] **Sabeur Aridhi** [2]

Abstract

Proteins perform various functions in living organisms. Automatic protein function annotation is defined as finding appropriate association between proteins and functional labels like Gene Ontology (GO) terms. n this paper, we present a preliminary exploration of the potential of generative adversarial networks (GAN) for protein function annotation. The Prot-A-GAN aproach uses GAN-like adversarial training for learning embedding of nodes and relation in an heterogenous knowledge graph. Following the terminologies of GAN, we firstly train a discriminator using domain-adaptive negative sampling to discriminate positive and negative triples, and then, we train a generator to guide a random walk over the knowledge graph that identify paths between proteins and GO annotations. We evaluate the method by performing protein function annotation using GO terms on human disease proteins from UniProtKB/SwissProt. As a proof-of-concept, the conducted experiments show promising outcome and open up new avenue for further exploration for protein function annotation.

1. Introduction

The recent advances in Artificial Intelligence (AI) have proved its tremendous potential to revolutionize discoveries in the field of biology and health. Along with the progress in Next Generation Sequencing (NGS) technologies, affordable genome sequencing has made it possible for AI technologies to find use cases in genomics and health. A plethora of sequences are already available in public databases. For example, UniProt Knowledge Base[1] the largest and the most comprehensible public database for storing protein sequences contains more than 208 million sequences according to 2020_09 release. This large volume of protein sequences opens up opportunities to perform analyses beneficial to answering long-hold questions in biology. On the other hand, it poses challenges due to the fact that this huge base of data is nearly impossible to annotate by manual effort. In practice, UniProtKB is divided into two parts : UniProtKB/SwissProt and UniProtKB/TrEMBL. In UniprotKB/SwissProt, the protein sequences are manually annotated or manually reviewed. This is a tremendous job for human annotators. It requires significant amount of time to read publications, to find the information regarding a particular protein, to identify functional properties and finally, to annotate it. The total process is costly as well. These are the primary reasons of UniProtKB/SwissProt having very slow growth over time. According to 2020_09 release, there are roughly 564 thousands of protein sequences which are manually reviewed.

On the contrary, in UniProtKB/TrEMBL, the protein sequences do not have proper annotation or possibly they have annotation from automatic tools but, are not manually reviewed. When UniProtKB receives a new protein sequence, it puts it into UniProtKB/TrEMBL with minimum processing and the sequence is available online for further investigation. This is one of the reasons why UniProtKB/TrEMBL has very sharp growth over the years. According to the release from September, 2020, there are 208 million protein sequences in TrEMBL. However, without proper functional annotation, the use of the protein sequences is very limited.

To enrich and exploit this immensely valuable data, it is essential to annotate these sequences with functional properties such as Enzyme Commission (EC) numbers or Gene Ontology (GO) terms, for example. To reduce the gap between the annotated and unannotated protein sequences, it is essential to develop accurate automatic protein function annotation techniques. A number of methods exist already but few of them consider the domain architecture or exploit the graph representation of proteins in their biological context.

In the recent years, research in knowledge graph has found

[1]Department of Computer Science and Engineering, Khulna University of Engineering & Technology, Khulna, Bangladesh [2]University of Lorraine, CNRS Inria LORIA, Nancy, France [3]Department of Statistics and Mathematics, University of Montreal, Montreal, Canada. Correspondence to: Bishnu Sarker <bishnukuet@gmail.com>.

The 2021 ICML Workshop on Computational Biology.

[1]https://www.uniprot.org/

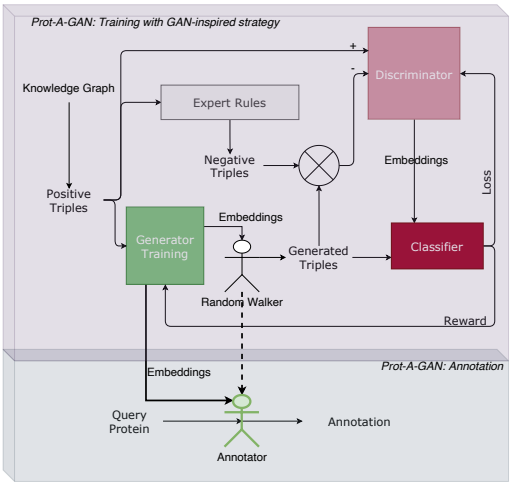

Figure 1. Schematic diagram of the proposed Prot-A-GAN framework.

a new surge in the machine learning community, especially, in the domain of representation learning. Learning representation on knowledge graphs, also known as knowledge graph embedding, aims a transforming entities and relations of knowledge graph into numerical vectors by encoding its topological properties. These numerical vectors can be used to perform downstream predictive tasks, for example, link prediction (Rossi et al., 2020; Bordes et al., 2013), entity resolution (Bordes et al., 2014; Nickel et al., 2011) , and entity classification (Nickel et al., 2012). And, thus it helps in effective and scalable discoveries in knowledge graphs (Mohamed et al., 2019). Generative Adversarial Network (GAN) (Goodfellow et al., 2014) has been a big success in computer vision in generating realistic images after training with sufficient amount of data.

Very recently, GAN has been re-purposed in the context of graph embedding (Wang et al., 2017; Cai & Wang, 2018; Ding et al., 2018; Wang et al., 2019). For example, GraphGAN (Wang et al., 2019) is a GAN-inspired node embedding technique where a GAN-like adversarial training is performed to train 1) a discriminator which is a simple cosine similarity function over two nodes (v_i, v_j), and 2) a generator which probabilistically selects node neighbors from the graph, given a node, and following a breadth-first search (BFS). One of the drawbacks of this technique is that it builds an entire BFS tree from the graph. This can get computationally intractable when the graph grows to be larger.

Although knowledge graph embedding techniques have generated much interests, GAN-based knowledge graph has not been explored in the context of automatic protein function annotation. The problem can be seen from the perspective of generating the annotations. In this case, given a protein, and an edge type, we are interested to find GO terms that are most likely to be associated with the proteins. Applying domain knowledge, and following the work of GraphGAN (Wang et al., 2019), the model can be trained to select the best annotations for the query protein.

In this work, we build a knowledge graph putting the biological constraints applicable in the case of protein function annotation. We propose a automatic protein function annotation framework using knowledge graph embedding technique to utilize protein meta-data in function prediction. We follow the works of GraphGAN to devise a discriminator that takes into account negative sample produced by applying the domain-specific biomedical knowledge, and a generator to guide a target specific depth-limited random walk to discover appropriate annotations from the knowledge graph. We formulate automatic protein function annotation task as a link prediction problem over a knowledge graph.

2. Prot-A-GAN framework

A knowledge graph $G = \{\mathcal{V}, \mathcal{E}, \mathcal{R}, \mathcal{A}\}$ is a kind of directed heterogeneous graph where, $\mathcal{V}$ is a set of objects/entities of different types, $\mathcal{A}$ denotes the set of node types., $\mathcal{R}$ is the set of relation types that connect the objects in $\mathcal{V}$ and $\mathcal{E}$ is the set of edges represented as triple of the form (s, p, o), s: subject/head/source node, p: predicate/relation/type and o: object/tail/destination node. The entities of the knowledge graph are mapped to their corresponding node type with a mapping function $\phi : V \longrightarrow \mathcal{A}$ and a link type mapping function $\psi : \mathcal{E} \longrightarrow \mathcal{R}$. Each entity $u \in \mathcal{V}$ belongs to an entity type $\phi(u) \in \mathcal{A}$, and each link $e \in \mathcal{E}$ belongs to a link type (relation) $\psi(e) \in \mathcal{R}$. $\mathcal{N}_s^p$ is the set of neighbors for node s for a particular relation p. Let us also consider that $f(s, p, o)$ with $s, o \in \mathcal{V}$, $p \in \mathcal{R}$ is a scoring function that estimates the likelihood of a triple to be a positive fact. Θ_G^V and Θ_G^R denotes d-dimensional embedding of nodes and relations, respectively, for the Generator. Similarly, Θ_D^V and Θ_D^R denotes d-dimensional embeddings of entity and relation, respectively, for the Discriminator.

2.1. Model Description

In the proposed framework shown in Figure 1, we adopt a training framework that closely follow GraphGAN. GraphGAN is proposed for node embedding in homogeneous networks and is not readily applicable in the case automatic protein function annotation from heterogeneous biomedical network. Here, we adapt the training framework for knowledge graphs which are heterogeneous graphs, and we design the training specifically for automatic protein function annotation.

As depicted in Figure 1, the first component to be trained is the **Generator Model** that tries to learn connectivity distribution as $prob(o|s, p)$ for triple (s, p, o) with the help

of a depth-limited random walk. It starts with a random initialization of its parameters, 1) entity embeddings Θ_G^V and 2) relation embeddings Θ_G^R. For each subject entity v_i, generator performs a depth-limited random walk. It stops as soon as it reaches an entity v_j for which $\phi(v_j)$ is the type of the desired target entities. The target entity and subject entity together with the prescribed relation r form a generated triple (v_i, r, v_j). The random walk starts from the protein, moves along the path chosen by probabilistic relevance score computed using Θ_G^V and Θ_G^R, and ends up selecting entities that are GO terms. Each random walk selects one GO term. Random walk is run several times with the same subject protein to find multiple GO annotations.

All of these generated samples are sent back to the discriminator to classify, and rewards are returned back to the generator, telling how negative the samples were. These rewards are then used to update the generator embeddings so that the random walk discovers more positive-likely triples.

Let us consider a set of triples made up of directly connected neighbor entities, and X be the list of the scores of the triples. The relevancy score, $relevance\_score = \frac{e^{X-max(X)}}{\sum e^{X-max(X)}}$ is computed as a softmax probability over X, and finally the relevancy score is used as probability in random selection of the next node in the random walk.

The second component is the **Discriminator Model** that tries to associate triples with a likelihood score that probabilistically classifies them into positive or negative triples. In the proposed model, we adopted translational distance as the scoring function for discriminator. The discriminator gives high score for negative triples and low score for positive triples. During the training the discriminator model parameters, namely Θ_D^V and Θ_D^R, are updated following the binary cross- entropy loss. The training algorithm is shown in Algorithm 1.

Sampling negative triples for training discriminator is vital to efficiently train a discriminator. Followings are the steps we take to sample the negative facts from the knowledge graph. For a particular relation type, we collect all the nodes acting as tail in the triples. Let us call this set U. For a particular entity s, and for the same particular relation type as above, we find the directly connected tail nodes. Let us call this set t_s. After that, we compute the set difference $U \setminus t_s$ to find the candidates for forming negative facts. For a particular entity s and relation p, the item $o \in t_s$ serves as the tail nodes when building the positive triples of the form (s, p, o). For each candidate triple, we compute the relevancy score and rank them based on this score. The least relevant triples are popped up as most negative triples. The relevancy score is computed as a softmax probability over all the the candidate facts.

3. Experiments and results

We have designed and built a knowledge graph supporting protein function annotation. The construction of the knowledge graph is explained in Section 3.1

3.1. Knowledge graph construction

We are interested in leveraging the huge amount of curated information available in UniProtKB/SwissProt for discovering new knowledge by building a knowledge graph named "UniProtinKG": UniprotKB in Knowledge Graph. In UniProtinKG, nodes are heterogeneous, i.e. nodes are from various types: proteins, domains, GOA terms, pathway reactions, tissue, genotypes and phenotypes, and glycosylation. While deciding the edge, we have hand-engineered the different edge types to reflect the nodes and the relations among them.

Algorithm 1 Training of Prot-A-GAN

Input: Knowledge graph G, Number of training epochs n_epochs, discriminator Interval $interval\_dis$, Generator Interval $interval\_gen$
Output: Θ_G^V, Θ_G^R, Θ_D^V, Θ_D^R

Initialize the model parameters Θ_G^V, Θ_G^R, Θ_D^V, Θ_D^R **for** $epoch <= n\_epochs$ **do**
  for $d\_epoch <= n\_epochs\_dis$ **do**
    if d_epoch *reaches* $interval\_dis$ **then**
      $facts\_dis \leftarrow get\_train\_data\_for\_dis(G)$
    Compute the discriminator loss Update the discriminator parameters Θ_D^V, Θ_D^R
  for $g\_epoch <= n\_epochs\_gen$ **do**
    if g_epoch *reaches* $interval\_gen$ **then**
      $facts\_gen \leftarrow get\_train\_data\_for\_gen(G)$
    Compute the reward for the generator using the discriminator parameters Θ_D^V, Θ_D^R Compute the loss for the generator Update the generator parameters Θ_G^V, Θ_G^R
  $epoch \leftarrow epoch + 1$

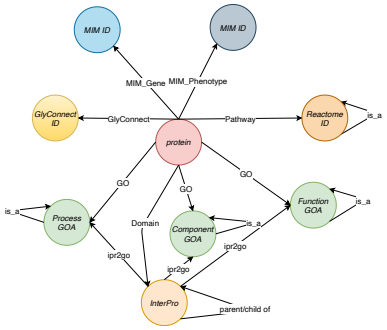

Figure 2. Schema of the UniProtinKG metagraph.

The schema of the UniProtinKG metagraph is shown in Figure 2. It shows how a protein is connected with its attributes. When many proteins are put together following this schema, the knowledge graph is formed from the connection to common attributes. This eventually opens up possibilities of discovering remote links among the entities.

As a proof of concept, the knowledge graph is built on the disease proteins from UniProtKB Disease database accessed on September, 2020 and available at: $https://www.uniprot.org/diseases/$. We call this as UniProtinKG-Disease graph. The UniProtinKG-Disease KG contains $|\mathcal{V}| = 26755$, $|\mathcal{E}| = 321596$, $|\mathcal{R}| = 18$ and $|\mathcal{A}| = 8$.

3.2. Evaluation Protocol

Prot-A-GAN is implemented using python and tensorflow machine learning framework following the implementation of GraphGAN with the modifications described above for heterogenous graph. The training is run for 30 epochs and inside, discriminator and generator modules are run for 5 epochs. Embedding dimension is 150.

Following previous works on protein function annotation, we adopted $precision = \frac{|A \cap P|}{|P|}$, $recall = \frac{|A \cap P|}{|A|}$, $F1 - measure = \frac{2 \times precision \times recall}{precision + recall}$ for P is the set of predicted GO terms and A is the set of ground-truth GO terms. To proceed with evaluation, we first prepared a set of proteins as test proteins. These test proteins are sampled at random from UniprotKB/SwissProt Disease database. We run the annotator for each test protein and record the predicted GO terms. They are not included in the set of disease proteins used for training the models. Moreover, because the final purpose of the method is to annotate proteins from which only the sequence is known, the test proteins will be connected to the UniProtinKG-Disease only through their domain information (from UniProtKB). The random walk searches for GO annotations following the guide of the trained generator, precisely using Θ_G^V and Θ_G^R. The final measure includes average precision, average recall and average F1-measure, calculated over the 799 test proteins, with and without post-processing. To improve the recall measure, we add a post-processing step consisting in expanding the list of predicted GO terms with all their ancestors in GO.

3.3. Results

In Table 1, we present the average precision, average recall and average F1 score for the 799 test proteins. For each protein in the test-set, we run Prot-A-GAN annotator and record the predicted GO terms.

We see that Prot-A-GAN has better precision when we run the annotator only once. In this case, the Prot-A-GAN executes random walks as many times as the number of domains connecting the test protein to the UniProtInKG network. As each domain gives a new path leading to a GO term, the number of predicted GO terms is equal to the number of domains or less if Prot-A-GAN fails to reach a GO term. As we increase the number of runs, we see a gradual decrease in the precision and a gradual increase in the recall. This behavior is explainable from the fact that when the number of runs increases, the number of predicted GO terms increases. This increases the number of false positives. Thus a reduced precision is observed. However, the high precision for single run indicates that Prot-A-GAN has the potential to discover high-quality annotations. At the same time, the low recall indicates the Prot-A-GAN can not discover all of the annotations. In the case of 10-run, we see a high recall compared to 1-run, 2-run and 5-run as it discovers higher number of GO terms by running for 10 times. Intuitively, the high number of GO terms introduce high number of false positives leading to a reduced precision.

Table 1. Automatic protein function annotation using Prot-A-GAN framework, trained on UniProtInKG-Disease

#Run	Post-Processing	Precision	Recall	F1-max	#Annotated
1	Yes	0.609	0.190	0.255	746/799
	No	0.325	0.074	0.108	746/799
2	Yes	0.587	0.281	0.331	764/799
	No	0.330	0.124	0.156	764/799
5	Yes	0.498	0.394	0.376	765/799
	No	0.302	0.204	0.199	746/799
10	Yes	0.415	0.499	0.376	765/799
	No	0.257	0.281	0.207	765/799

4. Conclusion

To the best of our knowledge, this is the first time GAN is merged with knowledge graph to design an automatic protein function annotator. The objective was to build a machine learning pipeline that leverages the power of adversarial learning on knowledge graph to discover functional annotations of proteins. The proposed approach opens a new direction in the research of automatic protein function annotation leveraging the power of GAN and knowledge graphs. Functional annotation using GO is relatively difficult. GO terms are arranged hierarchically in gene ontology. Moreover, each protein is annotated with multiple GO terms that can be distinctly placed in the ontology. Sometimes, a single protein can have more than 30 GO terms. Therefore, deciding on the number of annotations to generate is a dynamic decision. Apparently, the precision seems to be lower. However, applying post-processing approaches to gather around the ancestor annotations improved the outcome significantly.

Training adversarial models are very resource intensive. Applying adversarial training on knowledge graph is computationally very expensive and requires advanced hardware

settings. The hyper-parameters involved in the process has huge impact on the outcome of the experiment. Finding the right hyper-parameter configuration is very challenging due to the large search space and resource intensive training. Due to time and resource constraints, we could not present a thorough analysis of the impact of hyper-parameters on the experimental outcome. However, after few trials, we found this setting to be promising. This is a proof-of-concept and results are preliminary. Further experimentation is necessary to justify the performance. Moreover, it is well-known that the results of DL can be unstable,Thus, further experimentation is necessary

References

Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., and Yakhnenko, O. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, pp. 2787–2795, 2013.

Bordes, A., Glorot, X., Weston, J., and Bengio, Y. A semantic matching energy function for learning with multi-relational data. *Machine Learning*, 94(2):233–259, 2014.

Cai, L. and Wang, W. Y. Kbgan: Adversarial learning for knowledge graph embeddings. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 1470–1480, 2018.

Ding, M., Tang, J., and Zhang, J. Semi-supervised learning on graphs with generative adversarial nets. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pp. 913–922, 2018.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680, 2014.

Mohamed, S. K., Novácek, V., Vandenbussche, P.-Y., and Muñoz, E. Loss functions in knowledge graph embedding models. In *DL4KG@ ESWC*, pp. 1–10, 2019.

Nickel, M., Tresp, V., and Kriegel, H.-P. A three-way model for collective learning on multi-relational data. In *Icml*, volume 11, pp. 809–816, 2011.

Nickel, M., Tresp, V., and Kriegel, H.-P. Factorizing yago: scalable machine learning for linked data. In *Proceedings of the 21st international conference on World Wide Web*, pp. 271–280, 2012.

Rossi, A., Firmani, D., Matinata, A., Merialdo, P., and Barbosa, D. Knowledge graph embedding for link prediction: A comparative analysis. *arXiv preprint arXiv:2002.00819*, 2020.

Wang, H., Wang, J., Wang, J., Zhao, M., Zhang, W., Zhang, F., Li, W., Xie, X., and Guo, M. Learning graph representation with generative adversarial nets. *IEEE Transactions on Knowledge and Data Engineering*, 2019.

Wang, J., Yu, L., Zhang, W., Gong, Y., Xu, Y., Wang, B., Zhang, P., and Zhang, D. Irgan: A minimax game for unifying generative and discriminative information retrieval models. In *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*, pp. 515–524, 2017.