Structured Prediction Using Inference Networks

Kalpesh Krishna
140070017
kalpeshk2011@gmail.com

Arka Sadhu
140070011
ark.sadhu2904@gmail.com

Anish Ram Senathi
150100023
anish.rs97@gmail.com

Suman Swaroop
140050032
sumanswar12@gmail.com

1 Introduction

Structured prediction is a general term used for supervised learning, where the goal is to generate structured output in contrast to scalars. Often, the output space is exponentially large, making it difficult to use standard classification techniques.
Luckily, it is often the case that parts of the structured output are correlated with one another. This motivates the energy perspective of neural networks LeCun et al. [2006], where the objective is to find a structured output which minimizes the sum of a "local energy function" (interaction of inputs with outputs) and a "global energy function" (interaction of output units among themselves, independent of the input. The main goals of this project are -

- Study the set of papers developing structured prediction energy networks
- Implement a recent ICLR paper, [Tu and Gimpel, 2018] on structured prediction energy networks, on standard multi-label classification datasets
- Investigate the performance gains on a knowledge-base completion dataset FIGMENT Yaghoobzadeh and Schütze [2016]

We have open sourced our implementation. [1]

2 Related Work

[Belanger and McCallum, 2015] introduced "Structured Prediction Energy Networks (SPENs)", an energy-based approach using the power of deep learning. During training, this architecture uses a "margin-rescaled structured hinge" loss function Taskar et al. [2005]. During training as well as inference, this architecture requires a gradient descent over the energy surface to find the optimal y for every training / testing instance. This is a computational bottleneck for this model. An end-to-end approach for training SPENs is presented in Belanger et al. [2017], which provides strategies to train SPENs for non-MLC tasks.
Our target paper Tu and Gimpel [2018] attempts to speed up training and inference by using the energy network as a teacher for a student "inference network", similar to ideas in neural distillation Hinton et al. [2015]. Training this model requires optimization over a min-max game, which bears resemblance to Generative Adversarial Networks Goodfellow et al. [2014].

[1] https://github.com/TheShadow29/infnet-spen/

3 Description

The description is divided into four parts. First, we describe about the datasets used for multi-label classification and the differences in the datasets. Second, we describe the approach followed in implementing the paper. Third, we detail the evaluation schemes. Finally we show how the methods in WGAN Arjovsky et al. [2017] and Improved WGAN Gulrajani et al. [2017] can be incorporated into this work. To the best of our knowledge, this has not been explored yet in machine learning literature.

3.1 Datasets Used

For the purpose of multi-label classification we used the BibTeX, Bookmarks dataset Katakis et al. [2008] and the FIGMENT dataset Yaghoobzadeh and Schütze [2016].

- Bibtex Katakis et al. [2008] - 1836 input features, 159 labels. 4880 / 2515 train / test split. Each unit in the input vector is binary.
- Bookmarks Katakis et al. [2008] - 2150 input features, 208 labels. 48000 / 12000 / 27856 train / dev / test split. Each unit in the input vector is binary. We use the splits in Belanger and McCallum [2015]'s original implementation.[2]
- FIGMENT Yaghoobzadeh and Schütze [2016] - 200 input features, 102 labels. 101266 / 40220 / 60447 train / dev / test split. This dataset is derived of a Freebase Knowledge Base Completion task. The input vectors are word embeddings derived from a large corpus. This dataset only corresponds to the "Global Model" mentioned in the original paper.[3]

3.2 Implementation of the Paper

3.2.1 Stage 0

We begin the process by training a feed-forward neural network which projects the input vectors into a smaller dense vector representation. For this we use a feature network which is a simple 3 layered fully connected network. This network is directly trained to reduce the cross entropy loss against the output. The cross-entropy loss here is defined for each label (since this is a multi-class problem) and then added for all the labels. **The negative of the last layer of this network is used to initialize b_i, the local energy term's parameter.**
We call this feature network F. It takes input data and gives a suitable feature vector given by $F(x)$. The weights of $F(x)$ are kept constant in the rest of the training process.

3.2.2 Stage 1

We now have a suitable feature representation of the input vector. We create an energy network $E_\theta(x, y)$ and an inference network A_ϕ. The energy network takes in input the feature representation as well as the all the labels and returns a scalar. The minima of the energy network refers to the inferred output. The inference network takes in the feature representation of the input, and gives the predicted labels. We want the distill the knowledge of the energy network ("teacher") into the inference network ("student"), similar to Hinton et al. [2015]. However unlike traditional distillation settings, we do not have access to a trained energy network, and we must jointly learn the energy and inference network.

The energy network is given as

$$E^{loc}(x, y) = \sum_{i=1}^{L} y_i b_i^T F(x)$$

$$E^{lab}(y) = c_2^T g(C_1 y)$$

$$E_\theta(x, y) = E^{loc}(x, y) + E^{lab}(x, y)$$

[2] `https://github.com/davidBelanger/SPEN`

[3] Find the data in `http://cistern.cis.lmu.de/figment/`

$$\hat{y} = \arg\min_{y} E_\theta(x, y)$$

Here $\hat{y}$ represent the inferred output. While y is restricted to the space of discrete values $\{0, 1\}$ we allow y to be continuous in the range $[0, 1]$ to allow continuity.

The b_i in the energy network are initialized as negatives of the last layer of feature network. (since the feature network is trained on a cross entropy loss, it models "probability" and not "energy").

The inference network is a 3-layer feed forward network which takes in input as the feature representation and ends with a sigmoid activation. **We want the inference network to give the same output as that of the minimizing y of the energy network for the same x.** Denoting the final inference network as A_ψ we have,

$$A_\psi = \arg\min_{y} E_\theta(x, y)$$

The loss function as proposed in Belanger and McCallum [2015] is given as

$$\min_{\theta} \sum_{(x_i, y_i) \in \mathcal{D}} [\max_{y \in \mathcal{Y}_R} \Delta(y, y_i) - E_\theta(x_i, y) + E_\theta(x_i, y_i)]_+$$

The $[\cdot]_+$ is essentially the ReLU function. This is the margin rescaled hinge structured loss Taskar et al. [2005]. We first solve the inner maximization by using gradient ascent over the space of y. The paper by Belanger and McCallum [2015] uses entropic mirror descent over the negative objective, which ensures that the labels y lie in the range [0, 1]. However, we couldn't find any implementation of the algorithm and instead went ahead with simple projected gradient descent on the labels of y.

Taking $y = A_\phi(x)$ and taking the max function outside, we get the loss function as

$$\min_{\theta} \max_{\phi} \sum_{(x_i, y_i) \in \mathcal{D}} [\Delta(A_\phi, y_i) - E_\theta(x_i, A_\phi) + E_\theta(x_i, y_i)]_+$$

This is optimized using alternate maximization and minimization. First we take max over ϕ values and then min over the θ values and continue the process. The paper [Tu and Gimpel, 2018] gives four different Δ functions that can be used. In our experiments we stuck to using the squared difference loss.

This min-max game resembles GANs Goodfellow et al. [2014], with the energy networks resembling discriminators and the inference networks generators. This model faces the common optimization challenges GANs are notoriously known for Salimans et al. [2016]. To avoid this, many regularization terms are added to the ϕ objective. In our experiments we particularly used the following objective function:

$$\begin{aligned}
\hat{\phi} &= \arg\max_{\phi} [\Delta(A_\phi(x_i), y_i) - E_\theta(x_i, A_\phi(x_i)) + E_\theta(x_i, y_i)]_+ \\
&\quad - \lambda_1 ||\phi||_2^2 - \lambda_2 H(A_\phi(x_i)) - \lambda_3 ||\phi - \phi_0||_2^2 \\
\hat{\theta} &= \arg\min_{\theta} [\Delta(A_\phi(x_i), y_i) - E_\theta(x_i, A_\phi(x_i)) + E_\theta(x_i, y_i)]_+ \\
&\quad - \lambda_1 ||\theta||_2^2
\end{aligned}$$

We use Adam optimizer with learning rate 1e-3 for alternate optimization of ϕ and θ. We keep $\lambda_2 = 1, \lambda_3 = 1$ and $\lambda_1 = 0.001$. We also regularize the θ optimization with a $l2$ weight regularization having a weight of 0.001

3.2.3 Stage 2

After the optimizations in stage 0 and stage 1, it is assumed that the energy network has learned a good surface over the labels. We simply re-tune the inference network on a small learning rate of 1e-5 or 1e-6, and the equation is given as:

$$\psi = \arg\min_{y \in \mathcal{Y}_R} \sum_{x \text{ in } \mathcal{X}} E_\theta(x, A_\psi(x)) + \lambda_2 H(A_\phi(x_i))$$

In our experiments we do this optimization over 20 epochs. Since this doesn't require ground truth labels, this can be performed in the validation set or test set inputs in a transductive setting.

3.2.4 Test-Time Inference

We first do inference over the validation set. This is carried out after each epoch. After every epoch, we note that the output we get from the inference network are not necessarily the probability associated with the corresponding label, as we have not explicitly trained with cross-entropy loss. For this reason, we compute the threshold which gives the best F1 score for the validation dataset and use the same threshold for the test dataset. We have observed very similar F1 scores and accuracy on both the validation and test datasets.

3.2.5 Parameter Tuning

We largely stuck to the original space of hyper-parameters adopted by Belanger and McCallum [2015] and Tu and Gimpel [2018]. This was confirmed after conversations with the authors. In our initial failed experiments we did run grid searches over much larger hyper-parameter spaces, primarily tuning the regularization λ values.

3.3 Evaluation Scheme

Two evaluation schemes are used. First is the accuracy which is ratio of correctly classified data to the total data. For multi-label classification an input is correctly classified if and only if **all** the output labels are correctly predicted. The other is F1-score (macro-averaged over examples) where each individual F1-score is the harmonic mean of the precision and recall. In many cases of multi-label classification, the F1-score may represent a better representation of the power of the classifier.[4]

In our experiments we used F1-score for the Bibtex and Bookmarks dataset and accuracy for the FIGMENT dataset.

3.4 Extensions of The Paper

3.4.1 WGAN Extension

As pointed out in [Tu and Gimpel, 2018], there is a resemblance of optimization schemes to the generative adversarial networks Goodfellow et al. [2014]. The inference network behaves like a generator while the energy network behaves like the discriminator.

We try to extend this resemblance to the WGAN Arjovsky et al. [2017]. We note that WGAN simply requires a K-Lipschitz function. The task of the network denoted by f_w can corresponded to the task of the Energy network. Energy network is as it is required to be K-Lipschitz. As in the WGAN paper, we can simply restrict or clamp the weights of energy network to a fixed box.

3.5 Improved WGAN Extension

We now refer to an extension of the original GAN paper Gulrajani et al. [2017]. This improves on the WGAN implementation by using a gradient penalty term instead of the usual weight clipping and that it demonstrates a more stable training. The gradient penalty term as used in the paper is

$$\lambda(||\nabla_{\hat{x}} D_w(\hat{x})||_2 - 1)^2$$

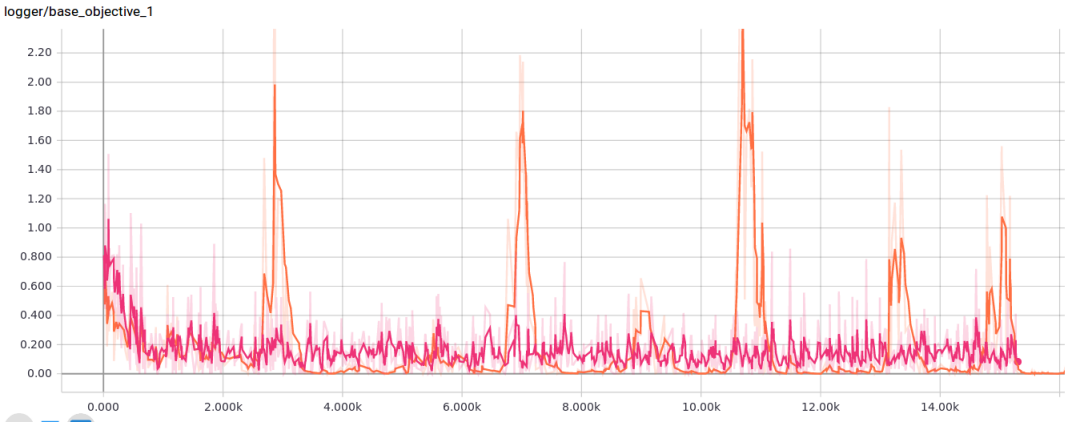
Here $\hat{x} = \epsilon x + (1-\epsilon) G_\theta(z)$ Here λ is just a regularization constant. As noted earlier, the discriminator corresponds to the energy network. The random noise z in the GAN framework corresponds to x in the SPEN framework, and the parameter x in the GAN framework would correspond to y which are the output labels. The generator in GAN framework would correspond to the inference network A.

So we define $\hat{y} = \epsilon y + (1-\epsilon) A(x)$ where ϵ is a randomly sampled number between 0 and 1. We then compute the gradient of the function $E(x, y)$ with respect to y. Let this gradient be denoted by $GradE(x, y)$. The penalty term would be

$$\lambda(||\nabla_{\hat{y}} E(x, \hat{y})||_2 - 1)^2$$

[4]Here is an exhaustive list of MLC benchmarks `http://manikvarma.org/downloads/XC/XMLRepository.html`

Figure 1: The Y-axis represents the base objective, a margin-rescaled structured hinge loss. The pink curve is an optimization over just the label energy terms, keeping the pre-trained local energy terms fixed. The orange curve jointly optimizes the two energy terms. Unfortunately, the orange curve training is very unstable in comparison to the pink curve. We propose a solution to this using W-GAN ideas.



This is very easy to implement in the current framework, as we already expect the energy function to be differentiable. **This term is only added to the θ objective and NOT the ϕ objective**.

4 Experiments

4.1 Code Description

- The code is written in Python language using the Tensorflow libary (tested on both v1.4 and v1.6) Abadi et al. [2015]. Total number of lines of code in the repository are 1863.
- We started the project from a generic TensorFlow template project MrGemy95 [2018]. Other than the template, the whole code is written from scratch.
- Our code can be found at the following link : `https://github.com/TheShadow29/infnet-spen`. The `README.md` file contains details about the code.

4.2 Experimental platform

- Since the datasets for the multi-label classification are small, we were able to work with it on our laptops. The main laptop we used was Dell XPS 13, running an 8^{th} generation Intel $i5$ processor with 8 cores (after hyperthreading) and 8GB RAM. Each experiment took less than 15 minutes or at the very least we were able to identify if the model was training properly. We had to do many runs of the experiment for debugging.

4.3 Results

4.4 Base Objective Optimization

To verify the success of our adversarial training, we plotted the base objective $[\Delta + E(x, y) - E(x, A(x))]_+$ vs the number of updates. We expect an overall decrease in this objective. We train this objective on two settings, with and without the local energy parameters fixed. (Belanger and McCallum [2015] suggest keeping the local energy parameters fixed during the initial optimization of label energy). We also hope to see an overall decrease in energy of the ground truth $E(x, y)$ for a fixed local energy term. We present these curves in **Fig. 1** and **Fig. 2** respectively.
Unfortunately, we see the joint optimization is more unstable, and the weights seem to be leaving the saddle point often during the process of optimization.

Figure 2: The Y-axis represents the ground truth energy for current mini-batch. The pink curve is an optimization over just the label energy terms, keeping the pre-trained local energy terms fixed. The orange curve jointly optimizes the two energy terms. As expected, the energy for the pink curve reduces due to correlations between the labels.

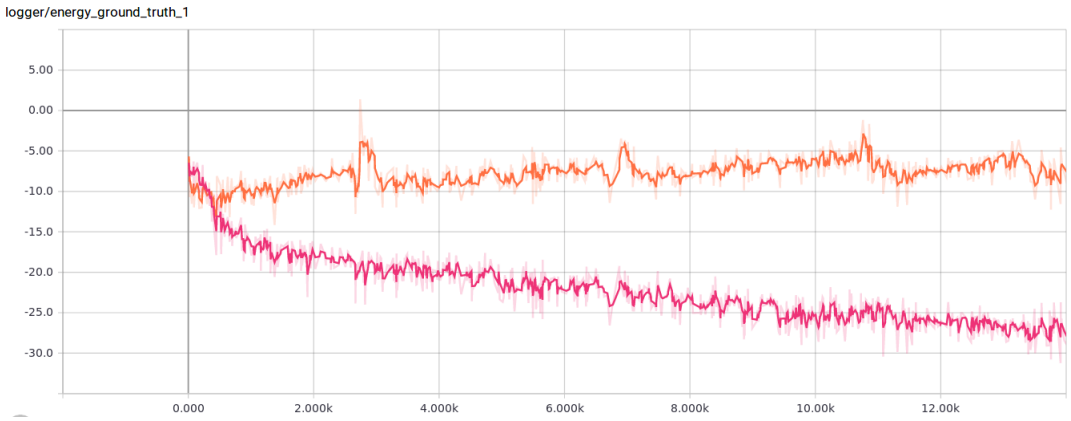
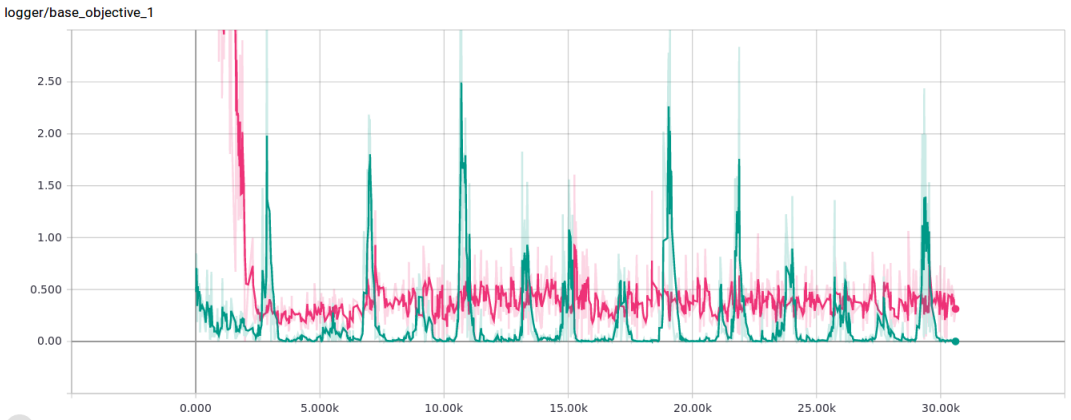


Figure 3: The Y-axis represents the base objective, a margin-rescaled structured hinge loss. The green curve is an optimization without the W-GAN condition whereas orange curve keeps a input gradient penalty to enforce Lipschitz continuity. We notice much more stable curves.



4.5 Benefits of W-GAN Techniques

Noticing the unstable parts of the optimization curve in the local energy setting, we try to introduce a gradient regularization penalty to enforce Lipschitz continuity of the energy function. We use $\lambda = 1$ in our experiments. We present our results in **Fig. 3**. While the W-GAN stabilizes the training process, we notice slightly inferior optimization which are reflected in slightly inferior results (up to 1 F1 point on BibTeX). Our W-GANs take more time to converge as well, and have no effect of pre-training as evident from **Fig. 4**. We want to emphasize that the instabilities in **Fig. 3** are significant (since a reader might believe otherwise after looking at **Fig. 4**), and correspond to F-1 score changes of over 15 points on BibTeX.
We eventually do not report W-GAN results in any of our further experiments (due to inferior F-1 scores), and leave this for future work.

4.5.1 F1-scores and Accuracy

All the F1-scores and Accuracy are averaged over 3 epochs. We note that our F1-scores are quite comparable to the ones reported in the paper by Tu and Gimpel [2018]. We beat the Knowledge-Base completion accuracy mentioned in Yaghoobzadeh and Schütze [2016], due to a stronger baseline as well as the Energy Network formulation.

Figure 4: A zoomed out version of **Fig. 3**. The W-GANs take longer to get to good region, and pre-training has no effect in stabilizing the process.

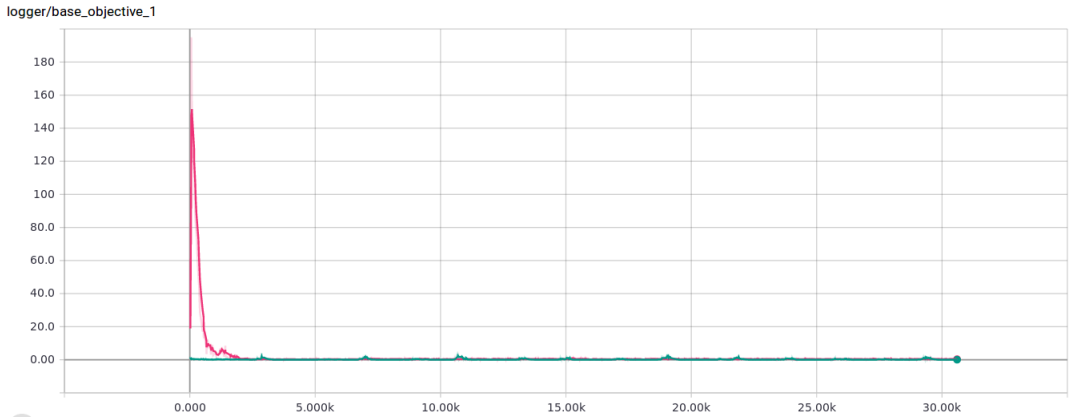


Table 1: F1-scores macro-averaged. Reported figures are taken from Tu and Gimpel [2018]

		BibTex	Bookmarks
MLP	OurModel-MLP	0.3787	0.3203
	Reported-MLP	0.389	0.338
Infnet	OurModel	0.4221	0.3574
	Reported-SPEN-BM16	0.422	0.344
	Reported-SPEN-E2E	0.381	0.339
	Reported-InfNet	0.422	0.376

4.5.2 Timing Analysis

We also perform a timing analysis of our work. The GPU used is GeForce GTx 1080 Ti. For training, we note that our base implementation of the MLP is around 1.75 times faster than that of authors, and inference implementation is about 2.4 times faster. The timing analysis is subject to batch size and the GPU used. So we note that if we normalize with respect to the train time of normal feed forward network, we still note that our project is a faster implementation.

Unfortunately, the test time results are way off. In our case, the inference is able to process only half as examples as that in the paper. But our MLP and InfNet inference are comparable which is expected behavior (since they both are identical architectures) and is also seen in Tu and Gimpel [2018]. We suspect that the original implementation did not divide the test data into mini-batches, which could account for the performance gap.

5 Effort

5.1 Timeline

The code commit history can be found in our repository[5].

- March first 3 weeks: Abstract Submission. We read the 3 main papers (Belanger and McCallum [2015], Belanger et al. [2017], Tu and Gimpel [2018]) for our task. We mainly discussed the Tu and Gimpel [2018] which is what we chose to implement for this work. We contacted the authors and according to their suggestion we first tried with the FIGMENT Dataset. We also read the Yaghoobzadeh and Schütze [2016] paper to understand the Knowledge Base Completion Task. We initialized the repository with the template project.
- March 4th week - April 1st week: We then set up the data loader for the FIGMENT dataset and then implemented the main algorithm. Unfortunately, the model did not train as expected.

[5] `https://github.com/TheShadow29/infnet-spen`

Table 2: Accuracy measures for the Figment Dataset. Reported measures is taken from Yaghoobzadeh and Schütze [2016]

	Test Set
OurModel - MLP	0.4463
OurModel	0.4565
Reported-Figment(GM)	0.426

Table 3: Timing Analysis. Batch size of 32 is used everywhere. All values are examples processed per second

		Training Time (On train set)			Inference Time (On dev set)		
		BibTex	Bookmarks	Figment	BibTex	Bookmarks	Figment
MLP	OurModel	33904.14	35723.41	50643.78	47985.57	46793.62	88182.58
	Reported-MLP	21670	19591	-	90706	92307	-
Infnet	OurModel	13275.04	12344.86	15013.35	49817.87	46429.22	85749.82
	Reported-SPEN-E2E	551	559	-	1420	1401	-
	Reported-InfNet	5533	5467	-	94194	88888	-

- April 2nd week - April 3rd week: We exchanged a few more emails with the authors to find where we were going wrong. A few things were not explicitly mentioned in the paper which were corrected. In the process we also partially implemented (not exact implementation) of the paper by [Belanger and McCallum, 2015].
- April 4th week - May 1st week: We identified a few more mistakes in our code and implemented the thresholding strategy which was suggested by the authors. The authors were also kind enough to share a part of the code. This led us to very similar F1-scores for the Bibtex dataset. We further extended this to Bookmarks dataset and FIGMENT dataset. We also designed the WGAN extensions Gulrajani et al. [2017] to InfNets.

5.2 Challenges

The most challenging part was definitely debugging the whole system. Unfortunately, there doesn't exist any easy method to find probable bugs in the system. Often it was the case that we couldn't be sure if the system is bug-free and the only problem was hyper-parameter tuning. After exchanging emails with the authors, we concluded that these four mistakes (which were not crystal-clear in Tu and Gimpel [2018]) hampered our progress -

- Not having a classification threshold and assuming it to be 0.5.
- Not pre-training b_i jointly with $F(x)$ in the first stage, and misunderstanding the scheme used to load the initial $A(x)$ parameters.
- Calculating $\sum_{minibatch}$ loss rather than $\frac{1}{B}\sum_{minibatch}$ loss. While Adam does not care about scaling parameter of the loss function, this design choice will not allow $\lambda = 1$ for the entropy regularization term as well as the pre-train bias term.
- Calculating label averaged F-1 scores rather than example averaged F-1 scores.

5.3 Work Distribution

The main coding part is done by Kalpesh Krishna. A few sub-parts have been implemented by Arka Sadhu. Debugging the code was done mostly together by Kalpesh Krishna and Arka Sadhu.

Suman also helped in finding a bug. Anish worked on the contextual model of the figment dataset, which could not get completed.

The contribution scale decided by the team members is Kalpesh: 40, Arka:40, Anish:10, Suman:10.

6 Acknowledgements

We thank the authors Lifu Tu and Prof. Kevin Gimpel who were very prompt in answering our queries. We also thank Lifu for sharing his code and releasing his hyper-parameters used in the optimization techniques. We thank Prof. Soumen Chakrabarti (IIT Bombay) for sharing his copy of the ClueWeb09 dataset. We thank Prof. Kevin Gimpel for suggesting the FIGMENT dataset and sharing a few novel energy function ideas which could be incorporated into FIGMENT. Finally, we thank Prof.Sunita Sarawagi for guiding us throughout the project.

References

Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, and F Huang. A tutorial on energy-based learning. *Predicting structured data*, 1(0), 2006.

Lifu Tu and Kevin Gimpel. Learning approximate inference networks for structured prediction. *CoRR*, abs/1803.03376, 2018. URL `http://arxiv.org/abs/1803.03376`.

Yadollah Yaghoobzadeh and Hinrich Schütze. Corpus-level fine-grained entity typing using contextual information. *CoRR*, abs/1606.07901, 2016. URL `http://arxiv.org/abs/1606.07901`.

David Belanger and Andrew McCallum. Structured prediction energy networks. *CoRR*, abs/1511.06350, 2015. URL `http://arxiv.org/abs/1511.06350`.

Ben Taskar, Vassil Chatalbashev, Daphne Koller, and Carlos Guestrin. Learning structured prediction models: A large margin approach. In *Proceedings of the 22nd international conference on Machine learning*, pages 896–903. ACM, 2005.

D. Belanger, B. Yang, and A. McCallum. End-to-End Learning for Structured Prediction Energy Networks. *ArXiv e-prints*, March 2017.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014. URL `http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf`.

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, pages 5769–5779, 2017.

I. Katakis, G. Tsoumakas, and I. Vlahavas. Multilabel Text Classification for Automated Tag Suggestion. 2008.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, pages 2234–2242, 2016.

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `https://www.tensorflow.org/`. Software available from tensorflow.org.

MrGemy95. Mrgemy95/tensorflow-project-template, Mar 2018. URL `https://github.com/MrGemy95/Tensorflow-Project-Template`.