Distractor Generation with Generative Adversarial Nets for Automatically Creating Fill-in-the-blank Questions

Chen Liang[†], Xiao Yang[‡], Drew Wham[*], Bart Pursel[*], Rebecca Passonneau[‡], C. Lee Giles[†]

[†]Information Sciences and Technology [‡]Computer Science and Engineering [*]Teaching and Learning with Technology
Pennsylvania State University
University Park, PA 16802
{cul226,xuy111,fcw5014,bkp10,rjp49}@psu.edu,giles@ist.psu.edu

ABSTRACT

Distractor generation is a crucial step for fill-in-the-blank question generation. We propose a generative model learned from training generative adversarial nets (GANs) to create useful distractors. Our method utilizes only context information and does not use the correct answer, which is completely different from previous Ontology-based or similarity-based approaches. Trained on the Wikipedia corpus, the proposed model is able to predict Wiki entities as distractors. Our method is evaluated on two biology question datasets collected from Wikipedia and actual college-level exams. Experimental results show that our context-based method achieves comparable performance to a frequently used word2vec-based method for the Wiki dataset. In addition, we propose a second-stage learner to combine the strengths of the two methods, which further improves the performance on both datasets, with 51.7% and 48.4% of generated distractors being acceptable.

CCS CONCEPTS

• **Computing methodologies → Neural networks**; • **Applied computing → Computer-assisted instruction**;

KEYWORDS

Question Generation, Generative Adversarial Nets

ACM Reference format:
Chen Liang[†], Xiao Yang[‡], Drew Wham[*], Bart Pursel[*], Rebecca Passonneau[‡], C. Lee Giles[†]. 2017. Distractor Generation with Generative Adversarial Nets for Automatically Creating Fill-in-the-blank Questions. In *Proceedings of K-CAP 2017: Knowledge Capture Conference, Austin, TX, USA, December 4–6, 2017 (K-CAP 2017),* 4 pages.
https://doi.org/10.1145/3148011.3154463

1 INTRODUCTION

Among various types of questions, the *fill-in-the-blank* (FITB) question is widely used in many academic settings. A FITB question consists of three parts: (i) a question sentence or *stem*; (ii) the correct answer or *key*; (iii) *distractor answers which we call distractors.* Automatic fill-in-the-blank question generation (FITB-QG) is a promising research area and a successful system would permit faster and less expensive question creation on a large scale.

We here focus on **distractor generation** (DG), i.e., generating distractors given the question sentence and the key to the question. DG is a crucial step in FITB-QG because one of its main challenges is the generation of "good" distractors which can distinguish knowledgeable test takers from less knowledgeable ones, in the sense that the question becomes more effective at testing a student's knowledge. Most existing methods of DG are based on semantic similarities between the key and the candidate distractor [1, 5, 12]. Various similarities have been utilized in different work such as WordNet similarity [19], word2vec similarity [18], n-gram co-occurrence likelihood, etc. Distractors are selected from a ranked list based on a weighted combination of different similarity metrics, where the weights are usually ad-hoc. Additionally, these DG methods have not fully explored how to utilize the context information (stem).

Different from existing approaches which heavily depend on the key, we propose to learn distractor distribution conditioned on the stem, since the semantic information conveyed by the stem is also critical to generate "good" distractors. Specifically, we adapt generative adversarial nets (GANs) [4] to tackle DG. We simultaneously train two models: a generative model G that captures real data (corresponding to the key to the FITB question) distribution given a context (corresponding to the stem), and a discriminative model D that estimates the probability that a sample comes from the real training data rather than G. The training procedure for G is to maximize the probability of D making a mistake. Distractors can be generated according to the distribution estimated by G.

The proposed GAN model is trained on the Wikipedia corpus and is able to predict Wiki entities as distractors. We evaluate the proposed method on two biology question datasets: (i) Wiki-FITB where 30 sentences from Wikipedia are selected and transformed into FITB questions; (ii) Course-FITB where 92 FITB questions are selected from actual college-level exams. Our method is compared with a widely used word2vec-based method. For each question, a list of distractors is generated and evaluated by domain experts. Given predictions of the two methods, we propose to apply a second-stage learner to utilize information in both the stem and the key. This outperforms both the proposed GAN-based method and the word2vec-based method on two datasets, with 51.7% and 48.4% of distractors generated being acceptable.

K-CAP 2017, December 4–6, 2017, Austin, TX, USA

ACM ISBN 978-1-4503-5553-7/17/12…$15.00
https://doi.org/10.1145/3148011.3154463

Our major contribution is summarized as follows.

- The proposed machine learning-based approach is fundamentally different from previous unsupervised similarity-based approaches and it is the first application of GANs to DG.
- The proposed method only uses stem information and it can be used in combination with existing key-based methods for generating distractors that better fit question context.
- The college-level exam FITB question set can be used for evaluating distractor generation or general FITB-QG.

2 RELATED WORK

Early work in FITB-QG mainly focused on English language learning, with applications including evaluating students' vocabulary [22] and testing knowledge of using verbs [24], adjectives [14], and prepositions [13]. Recently, FITB-QG generated exercise questions as multiple-choice quizzes for one or multiple subjects [1, 2, 5, 12]. Our method aligns more closely to FITB-QG for general knowledge assessment.

For the DG problem, many systems utilize *WordNet* [19] to find synonyms or other related words as distractors [21]. Although WordNet has 117,000 synsets, its coverage is limited when compared to general knowledge bases such as Wikipedia. Other work has explored using the link structure of ontologies [23], which usually requires a pre-existing domain-specific ontology. Our method is text-based and orthogonal to these link-based methods. Other methods choose distractors from sentences in a constrained set of source texts [1, 9]. Our method does not have such constraints and can select distractors from the entire Wikipedia. Others also investigate various similarity metrics for DG, including embedding-based similarity [5, 12], co-occurrence likelihoods [6], syntactic similarities, etc. Our method is fundamentally different from these unsupervised similarity-based methods in that the training is supervised and distractors are selected with only the context information rather than the correct answer.

3 DISTRACTOR GENERATION

3.1 Generative Adversarial Nets

Proposed by Goodfellow et al. [4], generative adversarial nets are a novel approach to train a generative model. The key to GANs are two "adversarial" models: the Generator G and the Discriminator D. G is a generative model that aims to capture real data distribution $p_{data}(\mathbf{x})$. D is a discriminative model that estimates the probability that a sample came from the real training data rather than G. Both G and D could be a non-linear mapping function. To learn the generator's distribution p_g over data $\mathbf{x}$, G parameterized by θ_g maps a prior noise distribution $p_z(\mathbf{z})$ to the data space as $G(\mathbf{z};\theta_g)$. On the other hand, the discriminator $D(\mathbf{x};\theta_d)$ parameterized by θ_d outputs a single scalar representing the probability that a sample $\mathbf{x}$ came from training data rather than p_g.

D is trained to maximize the probability of assigning the correct label to both training examples and samples from G. Simultaneously G is trained to maximize the probability of D making a mistake, i.e., minimizing $\log(1 - D(G(\mathbf{z})))$. The whole training procedure for G and D follows a two-player minimax game:

$$\min_G \max_D \mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_z(\mathbf{z})}[\log(1 - D(G(\mathbf{z})))] \quad (1)$$

3.2 Conditional GANs for DG

Since our goal is to generate distractors given a question sentence, we therefore adapt a GAN to distractor generation such that both the generator and the discriminator are conditioned on the extra context $\mathbf{c}$ learned from the question sentence (see Section 3.3). For this we put $\mathbf{c}$ into both the discriminator and generator as additional input. More precisely in the generator, the combination of a noise vector $\mathbf{z}$ and $\mathbf{c}$ is taken as a joint input, while in the discriminator, both the generated sample $\mathbf{x}$ and $\mathbf{c}$ are utilized to determine whether $\mathbf{x}$ came from training data. As a consequence, the minimax game in Equation 1 can be rewritten as:

$$\min_G \max_D \mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})}[\log D(\mathbf{x}|\mathbf{c})] + \mathbb{E}_{\mathbf{z}\sim p_z(\mathbf{z})}[\log(1 - D(G(\mathbf{z}|\mathbf{c})))] \quad (2)$$

which is the conditional GAN proposed by [20].

The neural-network based training framework provides additional flexibility on how the input vectors are combined. In the generator, we simply concatenate both $\mathbf{z}$ and $\mathbf{c}$ to build another vector representation: $\tilde{\mathbf{z}} = \mathbf{z} \odot \mathbf{c}$ where $\odot$ represents the concatenation. In the discriminator, we concatenate the linear transformation of the generated sample $\mathbf{x}$ and $\mathbf{c}$: $\tilde{\mathbf{x}} = (W_x\mathbf{x}) \odot \mathbf{c}$, where W_x is a weight matrix to be learned. $\tilde{\mathbf{x}}$ is then fed to a multi-layer perceptron.

GANs have a serious limitation requiring that the composition of the generator and the discriminator are fully differentiable. This is not true for discrete variables such as tokens in the text. Since the generator has an output softmax layer which can be interpreted as the probability of yielding each token, we sample a discrete token value from the distribution. As such, the back-propagation algorithm alone cannot provide a valid training signal for the generator since the sampling operation is not differentiable.

For this a number of approaches have been proposed, including policy gradient [25] and Gumbel softmax trick [8, 16]. We adopt the Gumbel softmax method since the policy gradient method involves the design of a reward function, which can lead to training instability. While prior work on Gumbel softmax trick were on datasets with a small number of classes (e.g. 10 classes for MNIST dataset), here we investigate its effectiveness on datasets with orders of magnitude more classes.

Similar to the re-parameterization trick in variational auto-encoder (VAE) [11], we efficiently draw samples $\mathbf{x}$ from a categorical distribution with class probabilities $\boldsymbol{\pi}$ using the Gumbel-Max trick [17]:

$$\mathbf{x} = \text{one\_hot}(\operatorname*{argmax}_i[g_i + \log \pi_i]) \approx \text{softmax}(\mathbf{g} + \log \boldsymbol{\pi}) \quad (3)$$

where g_i $(i = 1, 2, \cdots, K)$ is an i.i.d sample drawn from Gumbel$(0, 1)$ distribution and K is the total number of classes. The softmax function is used as a continuous, differentiable approximation to argmax, generating a K-dimensional sample vector $\mathbf{x} \in \mathbb{R}^{K-1}$ where

$$x_i = \frac{\exp((g_i + \log \pi_i)/\tau)}{\sum_{j=1}^{K} \exp((g_j + \log \pi_j)/\tau)} \quad (4)$$

with τ being a temperature hyper-parameter.

Since the Gumbel-Max trick enables back-propagating training signals from the discriminator to the generator, we can use the minimax optimization for discrete variables.

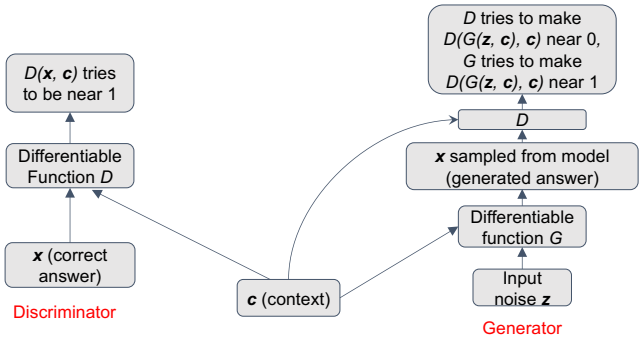

Figure 1: Conditional GANs for Distractor Generation.

Figure 1 presents the general architecture of the proposed conditional GAN for DG, where $\mathbf{c}$ is the learned context vector representation of the question stem, $\mathbf{x}$ the vector representation of the correct answer or the generated sample, and $\mathbf{z}$ a noise vector sampled from a prior noise distribution. G represents the generator, and D represents the discriminator.

3.3 Context Modeling of the Question Sentence

The architecture proposed in Figure 1 requires a context vector $\mathbf{c}$ of the question stem. To model the context, we use long short-term memory (LSTM) [7]. Specifically, since each question sentence could be divided into a left (s_L) and a right (s_R) part by the position of the blank, two LSTMs (LSTM_L and LSTM_R) are used to model the left context ($\mathbf{c}_L$) and the right context ($\mathbf{c}_R$) separately. The context vector $\mathbf{c}$ is therefore a concatenation of $\mathbf{c}_L$ and $\mathbf{c}_R$. Consider the question sentence "*A _____ is a networking device that forwards data packets between computer networks.*". The question is first split into two parts:"*A*" and "*is a networking device that forwards data packets between computer networks.*" The left and right part is then fed into LSTM_L and LSTM_R respectively, resulting in $\mathbf{c}_L$ and $\mathbf{c}_R$. Next we concatenate $\mathbf{c}_L$ and $\mathbf{c}_R$ to obtain the context vector $\mathbf{c}$. In practice, we reverse the order of the words in the right part s_R when feeding it to LSTM_R, in order to emphasize neighboring words around the blank.

3.4 Implementation Details

The generator and the discriminator is implemented by a 2-layer perceptron with a hidden size of 350. The network utilizes Leaky ReLU [15] activation. We set the noise vector $\mathbf{z}$ as a 50-dimensional vector drawn from $\mathcal{N}(0,1)$ and each LSTM to have a hidden size of 150. During training, Adam algorithm [10] with a learning rate of 0.001 is used. The temperature τ is fixed as 1.

4 EXPERIMENTS

4.1 Data Preparation

Training the proposed conditional GAN model requires a large number of (stem, key) pairs. We propose to utilize the Wikipedia corpus for creating the training set. To remove part of the sentence as a blank, we exploit the link structure among different Wiki pages. Specifically, we substitute the link in a sentence with a blank to get the stem and use the linked Wiki concept[1] as the key. Thus sentences with links could be transformed into (stem, key) pairs. Note that FITB-QG model usually is tailored to a certain domain, such as physics, biology, mathematics, etc. For each domain, we select a subset of all created pairs as training data based on whether the key appears in the domain-specific concept vocabulary. Such vocabulary is built by breadth-first searches starting from several main concepts of the domain and filtering with several semantic similarities such as LDA [3], word2vec, etc.

Our experiments here focus on biology. With the procedure described above, we build a vocabulary with 8879 biology-related concepts and create a training set with 1.62 million (stem, key) pairs. In addition, we create two test sets for evaluation: (i) **Wiki-FITB**: 30 FITB questions based on sentences in Wikipedia, selected by a domain expert; (ii) **Course-FITB**: 92 FITB questions from actual exams for two college-level biology courses and GRE (biology subject) 2016.

4.2 Experiment Settings

For each (stem, key) pair, we apply the proposed FITB-QG method to generate a list of distractors. Three domain experts with teaching experience, a Ph.D. in biology, a Ph.D. candidate in biology and a Ph.D. candidate in entomology were then asked collaboratively to label each of the top-4 predictions as a *Good*, *Fair*, or *Bad* distractor.

We compare the proposed GAN-based FITB-QG model (**GAN**) with a frequently used similarity-based method (**W2V**), which generates distractors based on the word2vec similarity between the candidate and the key. We trained a word2vec model on the Wiki corpus with each concept treated as an individual token.

Since a GAN only utilizes information in the *stem* part while W2V only utilizes information in the *key* part, we additionally apply *a second-stage learner* (**GAN+W2V**) to combine the strengths of GAN and W2V. For each (stem, key, distractor) tuple, we use the prediction score and the ranking of GAN and W2V as four features, and train a logistic regression classifier to predict the probability of a distractor being good, fair, or bad. The final distractor predictions are ranked by the probability of being bad estimated by the second-stage learner.

4.3 Experimental Results

The distractor generation results on two datasets are shown in Table 1. We evaluate each method in a leave-one-out manner and report the 95% confidence intervals of percentages of generated distractors being good, fair, or bad.

When comparing GAN with W2V, we can see that they achieve comparable performance on Wiki-FITB and that W2V is significantly better than GAN on Course-FITB. Since the GAN is based on the question stem, its distractor generation process is solely dependent on the learned association between the context information and distractors. Course-FITB, collected from actual college exams, is a more challenging dataset because its writing style is different from the part of Wikipedia on which the GAN is trained. Such difference makes it difficult for the GAN to apply the association learned from Wikipedia to questions in Course-FITB. By design distractors are often semantically related to the key (e.g. DNA and RNA). As such similarity-based methods like W2V can provide a strong baseline since they explicitly utilize information

[1]Each concept corresponds to an English Wiki article.

Table 1: Distractor generation results. Numbers are 95% confidence intervals of percentages of generated distractors being good, fair, or bad, calculated in a leave-one-out manner.

Methods	Good	Fair	Bad
GAN	28.4 (10.8)	10.8 (5.5)	60.8 (10.5)
W2V	35.8 (6.8)	10.0 (5.0)	54.2 (8.0)
GAN + W2V	40.0 (7.8)	11.7 (5.0)	48.3 (8.6)

(a) Wiki-FITB

Methods	Good	Fair	Bad
GAN	17.7 (5.0)	9.2 (3.5)	73.1 (5.9)
W2V	32.9 (5.3)	11.9 (3.5)	55.2 (5.7)
GAN + W2V	34.3 (5.7)	14.1 (3.9)	51.6 (6.0)

(b) Course-FITB

Table 2: Distractor generation examples for question "Changes in gene frequency over time describes the process of _____.", whose key is *Evolution*. (Legend: Good, Fair, Bad)

GAN	W2V	GAN + W2V
Speciation	Natural selection	Speciation
Transcription	Macroevolution	Natural selection
Inbreeding	Evolutionary biology	Microevolution
Genetic drift	Microevolution	Genetic drift

about the key. W2V is limited in that it can only output the same distractors for a key regardless of question stems being different. Since question stems sharing the same keys may still emphasize on different aspects, it is desirable to generate diverse distractors for each specific question stem. As such context-based methods such as GAN are still valuable.

We can see that GAN + W2V reduces the mean percentages of badly generated distractors, compared to both GAN and W2V. The percentages of "acceptable" (good + fair) distractors are 51.7% for Wiki-FITB and 48.4% for Course-FITB. Although the difference is not significant given the small sizes of test set, the proposed second-stage learner provides an initial attempt to combine the strengths of GAN and W2V. Such learning-based method is a more systematic way than the ad-hoc weighted combination of different predictions.

Table 2 shows an example of generated distractors for the question "Changes in gene frequency over time describes the process of ____." We can see that GAN and W2V generate very different distractors. Since GAN is based on context, its predictions are related to "gene" in the stem. As for W2V, the key "Evolution" is utilized to retrieve similar concepts. In addition, we observe that GAN + W2V's predictions are a mix of GAN's and W2V's results, which reduces the percentages of bad distractors and makes the distractors more diverse.

5 CONCLUSION

We used conditional GANs for distraction generation for FITB problems which to out knowledge is the first use of GANs for this problem. Experiments on two collected biology question sets showed that the proposed context-based method is a valuable complement to previous similarity-based methods and that a second-stage learner can be applied to combining the strengths of two types of DG methods in order to achieve a better performance. Such methods should significantly help instructors create better FITB questions. Future work could be to explore: (i) a unified GAN model which can include key information; (ii) better context modeling to improve model generalization.

6 ACKNOWLEDGEMENTS

We gratefully acknowledge partial support from the Penn State Center for Online Innovation in Learning.

REFERENCES

[1] Manish Agarwal and Prashanth Mannem. 2011. Automatic gap-fill question generation from text books. In *BEA@ACL*. ACL, 56–64.
[2] Lee Becker, Sumit Basu, and Lucy Vanderwende. 2012. Mind the gap: learning to choose gaps for question generation. In *NAACL*. ACL, 742–751.
[3] David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *JMLR* 3 (2003), 993–1022.
[4] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *NIPS*. 2672–2680.
[5] Qi Guo, Chinmay Kulkarni, Aniket Kittur, Jeffrey P. Bigham, and Emma Brunskill. 2016. Questimator: Generating Knowledge Assessments for Arbitrary Topics. In *IJCAI*. 3726–3732.
[6] Jennifer Hill and Rahul Simha. 2016. Automatic Generation of Context-Based Fill-in-the-Blank Exercises Using Co-occurrence Likelihoods and Google n-grams.. In *BEA@NAACL*. 23–30.
[7] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.
[8] Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical Reparameterization with Gumbel-Softmax. In *ICLR*.
[9] Nikiforos Karamanis, Le An Ha, and Ruslan Mitkov. 2006. Generating multiple-choice test items from medical text: A pilot study. In *Proceedings of the Fourth International Natural Language Generation Conference*. ACL, 111–113.
[10] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR*.
[11] Diederik P Kingma and Max Welling. 2014. Auto-encoding variational bayes. In *ICLR*.
[12] Girish Kumar, Rafael E Banchs, and Luis Fernando D'Haro Enriquez. 2015. Revup: Automatic gap-fill question generation from educational texts. In *BEA*. ACL.
[13] John Lee and Stephanie Seneff. 2007. Automatic generation of cloze items for prepositions.. In *INTERSPEECH*. 2173–2176.
[14] Yi-Chien Lin, Li-Chun Sung, and Meng Chang Chen. 2007. An automatic multiple-choice question generation scheme for english adjective understanding. In *ICCE*. 137–142.
[15] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. 2013. Rectifier nonlinearities improve neural network acoustic models. In *in ICML Workshop on Deep Learning for Audio, Speech and Language Processing*.
[16] Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. 2017. The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables. In *ICLR*.
[17] Chris J Maddison, Daniel Tarlow, and Tom Minka. 2014. A* sampling. In *NIPS*. 3086–3094.
[18] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*. 3111–3119.
[19] George A Miller. 1995. WordNet: a lexical database for English. *CACM* 38, 11 (1995), 39–41.
[20] Mehdi Mirza and Simon Osindero. 2014. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784* (2014).
[21] Ruslan Mitkov, Le An Ha, and Nikiforos Karamanis. 2006. A computer-aided environment for generating multiple-choice test items. *Natural language engineering* 12, 2 (2006), 177–194.
[22] Juan Pino and Maxine Eskenazi. 2009. Semi-automatic generation of cloze question distractors effect of students' L1.. In *SLaTE*. 65–68.
[23] Katherine Stasaski and Marti Hearst. 2017. Multiple Choice Question Generation Utilizing An Ontology. In *BEA@EMNLP*. 303.
[24] Eiichiro Sumita, Fumiaki Sugaya, and Seiichi Yamamoto. 2005. Measuring non-native speakers' proficiency of English by using a test with automatically-generated fill-in-the-blank questions. In *BEA*. ACL, 61–68.
[25] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. Seqgan: sequence generative adversarial nets with policy gradient. In *AAAI*.