Distractor Generation for Multiple Choice Questions Using Learning to Rank

Chen Liang[1], **Xiao Yang**[2], **Neisarg Dave**[1], **Drew Wham**[3], **Bart Pursel**[3], **C. Lee Giles**[1]
[1]Information Sciences and Technology
[2]Computer Science and Engineering
[3]Teaching and Learning with Technology
Pennsylvania State University
{cul226,xuy111,nud83,fcw5014,bkp10}@psu.edu, giles@ist.psu.edu

Abstract

We investigate how machine learning models, specifically ranking models, can be used to select useful distractors for multiple choice questions. Our proposed models can learn to select distractors that resemble those in actual exam questions, which is different from most existing unsupervised ontology-based and similarity-based methods. We empirically study feature-based and neural net (NN) based ranking models with experiments on the recently released SciQ dataset and our MCQL dataset. Experimental results show that feature-based ensemble learning methods (random forest and LambdaMART) outperform both the NN-based method and unsupervised baselines. These two datasets can also be used as benchmarks for distractor generation.

1 Introduction

Multiple choice questions (MCQs) are widely used as an assessment of students' knowledge and skills. A MCQ consists of three elements: (i) *stem*, the question sentence; (ii) *key*, the correct answer; (iii) *distractors*, alternative answers used to distract students from the correct answer. Among all methods for creating good MCQs, finding reasonable distractors is crucial and usually the most time-consuming. We here investigate automatic *distractor generation* (DG), i.e., generating distractors given the stem and the key to the question. We focus on the case where distractors are not limited to single words and can be phrases and sentences.

Rather than generate trivial wrong answers, the goal of DG is to generate plausible false answers - good distractors. Specifically, a "good" distractor should be at least semantically related to the key (Goodrich, 1977), grammatically correct given the stem, and consistent with the semantic context of the stem. Taking these criterion into consideration, most existing methods for DG are based on various similarity measures. These include WordNet-based metrics (Mitkov and Ha, 2003), embedding-based similarities (Guo et al., 2016; Kumar et al., 2015; Jiang and Lee, 2017), n-gram co-occurrence likelihood (Hill and Simha, 2016), phonetic and morphological similarities (Pino and Eskenazi, 2009), structural similarities in an ontology (Stasaski and Hearst, 2017), a thesaurus (Sumita et al., 2005), context similarity (Pino et al., 2008), context-sensitive inference (Zesch and Melamud, 2014), and syntactic similarity (Chen et al., 2006). Then distractors are selected from a candidate distractor set based on a weighted combination of similarities, where the weights are determined by heuristics.

In contrast to the above-mentioned similarity-based methods, we apply learning-based ranking models to select distractors that resemble those in actual exam MCQs. Specifically, we propose two types of models for DG: feature-based and NN-based models. Our models are able to take existing heuristics as features and learn from these questions a function beyond a simple linear combination. Learning to generate distractors has been previously explored in a few studies. Given a blanked question, Sakaguchi et al. (2013) use a discriminative model to predict distractors and Liang et al. (2017) apply generative adversarial nets. They view DG as a multi-class classification problem and use answers as output labels while we use them as input. Other related work (Welbl et al., 2017) uses a random forest. However, with the reported binary classification metrics, the quality of the top generated distractors is not quantitatively evaluated. Here we conduct a more comprehensive study on various learning models and devise ranking evaluation metrics for DG.

Machine learning of a robust model usually requires large-scale training data. However, to

Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications, pages 284–290
New Orleans, Louisiana, June 5, 2018.

the best of our knowledge, there is no benchmark dataset for DG, which makes it difficult to directly compare methods. Prior methods were evaluated on different question sets collected from textbooks (Agarwal and Mannem, 2011), Wikipedia (Liang et al., 2017), ESL corpuses (Sakaguchi et al., 2013), etc. We propose to evaluate DG methods with two datasets: the recently released SciQ dataset (Welbl et al., 2017) (13.7K MCQs) and the MCQL dataset (7.1K MCQs) that we made. These two datasets can be used as benchmarks for training and testing DG models. Our experimental results show that feature-based ensemble learning methods (random forest and LambdaMART) outperform both the NN-based method and unsupervised baselines for DG.

2 Learning to Rank for Distractor Generation

We solve DG as the following ranking problem:

Problem. Given a candidate distractor set $\mathcal{D}$ and a MCQ dataset $\mathcal{M} = \{(q_i, a_i, \{d_{i1}, ..., d_{ik}\})\}_{i=1}^{N}$, where q_i is the question stem, a_i is the key, $D_i = \{d_{i1}...d_{ik}\} \subseteq \mathcal{D}$ are the distractors associated with q_i and a_i, find a point-wise ranking function r: $(q_i, a_i, d) \rightarrow [0, 1]$ for $d \in \mathcal{D}$, such that distractors in D_i are ranked higher than those in $\mathcal{D} - D_i$.

This problem formulation is similar to "learning to rank" (Liu et al., 2009) in information retrieval. To learn the ranking function, we investigate two types of models: feature-based models and NN-based models.

2.1 Feature-based Models

2.1.1 Feature Description

Given a tuple (q, a, d), a feature-based model first transforms it to a feature vector $\phi(q, a, d) \in \mathbb{R}^d$ with the function ϕ. We design the following features for DG, resulting in a 26-dimension feature vector:

- *Emb Sim*. Embedding similarity between q and d and the similarity between a and d. We use the average GloVe embedding (Pennington et al., 2014) as the sentence embedding. Embeddings have been shown to be effective for finding semantically similar distractors (Kumar et al., 2015; Guo et al., 2016).

- *POS Sim*. Jaccard similarity between a and d's POS tags. The intuition is that ditractors might also be noun phrases if the key is a noun phrase.

- *ED*. Edit distance between a and d. This measures the spelling similarity and is useful for cases such as selecting "RNA" as a distractor for "DNA".

- *Token Sim*. Jaccard similarities between q and d's tokens, a and d's tokens, and q and a's tokens. This feature is motivated by the observation that distractors might share tokens with the key.

- *Length*. a and d's character and token lengths and the difference of lengths. This feature is designed to explore whether distractors and the key are similar in terms of lengths.

- *Suffix*. The absolute and relative length of a and d's longest common suffix. The key and distractors often have common suffixes. For example, "maltose", "lactose", and "suctose" could be good distractors for "fructose".

- *Freq*. Average word frequency in a and d. Word frequency has been used as a proxy for words' difficulty levels (Coniam, 1997). This feature is designed to select distractors with a similar difficulty level as the key.

- *Single*. Singular/plural consistency of a and d. This checks the consistency of singular vs. plural usage, which will select grammatically correct distractors given the stem.

- *Num*. Whether numbers appear in a and d. This feature will cover cases where distractors and keys contain numbers, such as "90 degree", "one year", "2018", etc.

- *Wiki Sim*. If a and d are Wikipedia entities, we calculate their Wiki embedding similarity. The embedding is trained using word2vec (Mikolov et al., 2013) on Wikipedia data with each Wiki entity treated as an individual token. This feature is a complement to Emb Sim where sentence embedding is a simple average of word embeddings.

2.1.2 Classifiers

We study the following three feature-based classifiers: (i) Logistic Regression: an efficient generalized linear classification model; (ii) Random Forest (Breiman, 2001): an effective ensemble

classification model; (iii) LambdaMART (Burges, 2010): a gradient boosted tree based learning-to-rank model. To train these models, following previous notations, we use D_i as positive examples and sample from $\mathcal{D} - D_i$ to get negative examples.

2.2 NN-based Models

Based on the recently proposed method IRGAN (Wang et al., 2017), we propose an adversarial training framework for DG. Our framework consists of two components: a generator G and a discriminator D. G is a generative model that aims to capture the conditional probability of generating distractors given stems and answers $P(d|q,a)$. D is a discriminative model that estimates the probability that a distractor sample comes from the real training data rather than G.

Assume that the discriminator is based on an arbitrary scoring function $f_\phi(d,q,a) \in \mathbb{R}$ parameterized by ϕ, then the objective for D is to maximize the following log-likelihood:

$$\max_{\phi} \mathbb{E}_{d\sim P_{\text{true}}(d|q,a)}[\log(\sigma(f_\phi(d,q,a)))] + \mathbb{E}_{d\sim P_\theta(d|q,a)}[\log(1-\sigma(f_\phi(d,q,a)))] \quad (1)$$

where σ is the sigmoid function. For the generator G, we choose another scoring function $f_\theta(d,q,a) \in \mathbb{R}$ parameterized by θ, evaluate it on every possible distractor d_i given a (q,a) pair, and sample generated distractors based on the discrete probability after applying softmax:

$$p_\theta(d_i|q,a) = \frac{\exp(\tau \cdot f_\theta(d_i,q,a))}{\sum_j \exp(\tau \cdot f_\theta(d_j,q,a))} \quad (2)$$

where τ is a temperature hyper-parameter.

In practice, since the total size of distractors is large, it is very time-consuming to evaluate on every possible d_i. Following the common practice as in (Wang et al., 2017; Cai and Wang, 2018), we uniformly sample K candidate distractors for each (q,a) pair and evaluate f_θ on each $d_i, \forall i \in [1,K]$. The objective for G is to "fool" D so that D misclassifies distractors generated by G as positive:

$$\min_{\theta} \mathbb{E}_{d\sim P_\theta(d|q,a)}[\log(1-\sigma(f_\phi(d,q,a)))] \quad (3)$$

The training procedure follows a two-player minimax game, where D and G are alternatively optimized towards their own objective.

The scoring function f_ϕ and f_θ can take arbitrary forms. IRGAN utilizes a convolutional neural network based model to obtain sentence embeddings and then calculates the cosine similarities. However, such a method ignores the word-level interactions, which is important for the DG task. For example, if the stem asks "which physical unit", good distractors should be units. Therefore, we adopt the Decomposable Attention model (DecompAtt) (Parikh et al., 2016) proposed for Natural Language Inference to measure the similarities between q and d. We also consider the similarities between a and d. Since they are usually short sequences, we simply use the cosine similarity between summed word embeddings. As such, the scoring function is defined as a linear combination of DecompAtt(d, q) and Cosine(d, a).

Dataset	$\lvert\mathcal{D}\rvert$	# MCQs	# Train	# Valid	# Test	Avg. # Dis
SciQ	22379	13679	11679	1000	1000	3
MCQL	16446	7116	5999	554	563	2.91

Table 1: Dataset Statistics.

2.3 Cascaded Learning Framework

To make the ranking process more efficient and effective, we propose a cascaded learning framework, a multistage ensemble learning framework that has been widely used for computer vision (Viola and Jones, 2001). We experiment with 2-stage cascading, where the first stage ranker is a simple model trained with part of the features in Sec. 2.1.1 and the second stage ranker can be any aforementioned ranking model. Such cascading has two advantages: (i) The candidate size is significantly reduced by the first stage ranker, which allows the use of more expensive features and complex models in the second stage; (ii) The second stage ranker can learn from more challenging negative examples since they are top predictions from previous stage, which can make the learning more effective.

3 Experiments

3.1 Datasets

We evaluate the proposed DG models on the following two datasets: (i) **SciQ** (Welbl et al., 2017): crowdsourced 13.7K science MCQs covering biology, chemistry, earth science, and physics. The questions span elementary level to college introductory level in the US. (ii) **MCQL**: 7.1K MCQs crawled from the Web. Questions are about biology, physics, and chemistry and at the Cambridge

O level and college level.

For SciQ, we follow the original train/valid/test splits. For MCQL, we randomly divide the dataset into train/valid/test with an approximate ratio of 10:1:1. We convert the dataset to lowercase, filter out the distractors such as "all of them", "none of them", "both A and B", and keep questions with at least one distractor. We use all the keys and distractors in the dataset as candidate distractor set $\mathcal{D}$. Table 1 summarizes the statistics of the two datasets after preprocessing. $|\mathcal{D}|$ is the number of candidate distractors. # MCQs is the total number of MCQs. # Train/Valid/Test is the number of questions in each split of the dataset. Avg. # Dis is the average number of distractors per question.

3.2 Experiment Settings

We use Logistic Regression (**LR**) as the first stage ranker. As for the second stage, we compare LR, Random Forest (**RF**), LambdaMART (**LM**), and the proposed NN-based model (**NN**). Specifically, we set C to 1 for LR, use 500 trees for RF, and 500 rounds of boosting for LM. For first stage training, the number of negative samples is set to be equal to the number of distractors, which is 3 for most questions. And we sample 100 negative samples for second stage training. More details can be found in the supplementary material. In addition, we also study the following unsupervised baselines that measure similarities between the key and distractors: (i) pointwise mutual information (**PMI**) based on co-occurrences; (ii) edit distance (**ED**), which measures the spelling similarity; and (iii) GloVe embedding similarity (**Emb Sim**). For evaluation, we report top recall (R@10), precision (P@1, P@3), mean average precision (MAP@10), normalized discounted cumulative gain (NDCG@10), and mean reciprocal rank (MRR).

3.3 Experimental Results

First Stage Ranker The main goal of the first stage ranker is to reduce the candidate size for the later stage while achieving a relatively high recall. Figure 1 shows the Recall@K for the first stage ranker on the two datasets. Validation set is used for choosing top K predictions for later stage training. We empirically set K to 2000 for SciQ and 2500 for MCQL to get a recall of about 90%.

Distractor Ranking Results Table 2 lists the ranking results for DG. From the table we observe

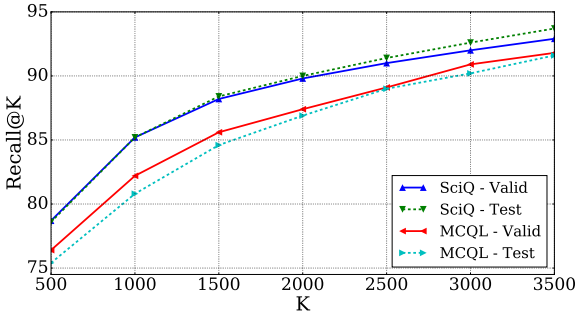

Figure 1: Recall@K for the first stage ranker.

1st Stage Ranker	2nd Stage Ranker	R@10	P@1	P@3	MAP @10	NDCG @10	MRR
	PMI	11.0	2.1	3.1	3.6	6.8	8.8
	ED	14.3	12.6	9.2	8.7	12.5	18.9
	Emb Sim	19.3	9.3	9.0	9.6	14.2	17.5
LR	LR	29.7	14.8	14.1	14.7	22.1	27.6
	RF	**44.1**	36.8	**27.0**	**28.4**	**38.0**	49.2
	LM	43.3	**37.2**	26.4	28.0	37.5	**49.3**
	NN	24.6	11.7	11.7	11.6	23.1	25.7
RF	—	41.4	31.2	23.7	25.0	34.4	44.0
LM	—	39.1	26.5	22.6	22.9	31.8	40.4

(a) SciQ

1st Stage Ranker	2nd Stage Ranker	R@10	P@1	P@3	MAP @10	NDCG @10	MRR
	PMI	20.7	5.9	6.8	7.8	13.5	16.2
	ED	32.1	34.6	23.6	23.7	30.5	42.8
	Emb Sim	32.1	25.6	18.4	20.4	26.9	33.9
LR	LR	42.9	29.3	24.5	26.6	35.1	42.2
	RF	48.4	**45.5**	**32.7**	**35.4**	**43.8**	**54.8**
	LM	**49.4**	42.8	31.5	34.5	43.4	53.6
	NN	36.5	22.9	22.5	22.7	34.6	36.7
RF	—	48.0	40.9	30.4	33.6	42.0	51.1
LM	—	46.7	42.5	30.6	33.0	41.6	52.7

(b) MCQL

Table 2: Ranking results (%) for DG.

the following: (i) The proposed ranking models perform better than unsupervised similarity-based methods (PMI, ED, and Emb Sim) most of the time, which is expected since similarity-based heuristics are used as features. (ii) Ensemble models - RF and LM - have comparable performance and are significantly better than other methods. These ensemble methods are more suitable for capturing the nonlinear relation between the proposed feature set and distractors. (iii) NN performs worse than feature-based models. The main reason is that NN is solely based on word embeddings. Although embedding similarity is the most important feature, information provided by other top features such as ED, Suffix, Freq is missing in NN. Given the limited training examples (11.6K for SciQ and 6K for MCQL), it is difficult to learn a robust end-to-end NN-based model.

#	SciQ	MCQL
1	Emb Sim (a, d)	Emb Sim (a, d)
2	Freq d	Token Sim (a, d)
3	Freq a	ED
4	Wiki Sim	Suffix
5	Emb Sim (q, d)	Suffix / len(d)
6	Suffix	Freq a
7	Suffix / len(d)	Wiki Sim
8	Suffix / len(a)	Freq d
9	Token Sim (a, d)	Emb Sim (q, d)
10	ED	Suffix / len(a)

Table 3: Top 10 important features.

Feature Analysis We conduct a feature analysis to have more insights on the proposed feature set. Feature importance is calculated by "mean decrease impurity" using RF. It is defined as the total decrease in node impurity, weighted by the probability of reaching that node, averaged over all trees of the ensemble. Table 3 lists the top 10 important features for SciQ and MCQL datasets. We find that: (i) the embedding similarity between a and d is the most important feature, which shows embeddings are effective at capturing semantic relations between a and d. (ii) String similarities such as Token Sim, ED, and Suffix are more important in MCQL than those in SciQ. This is consistent with the observation that ED has relatively good performance as seen in Table 2b. (iii) The set of top 10 features is the same for SciQ and MCQL, regardless of order.

Effects of Cascaded Learning Since we choose the top 2000 for SciQ and 2500 for MCQL from first stage, the ranking candidate size is reduced by 91% for SciQ and 85% for MCQL, which makes the second stage learning more efficient. To study whether cascaded learning is effective, we experiment with RF and LM without 2-stage learning, as shown as the bottom two rows in Table 2. Here we sample 100 negative samples for training models in order to make a fair comparison with other methods using 2-stage learning. We can see that the performance is better when cascaded learning is applied.

4 Conclusion

We investigated DG as a ranking problem and applied feature-based and NN-based supervised ranking models to the task. Experiments with the SciQ and the MCQL datasets empirically show that ensemble learning models (random forest and LambdaMART) outperform both the NN-based method and unsupervised baselines. The MCQL data is publicly available upon request. The two datasets can be used as benchmarks for further DG research. Future work will be to design a user interface to implement the proposed models to help teachers with DG and collect more user data for model training.

Acknowledgments

We gratefully acknowledge partial support from the Pennsylvania State University Center for Online Innovation in Learning and helpful comments from the reviewers.

A Training and Implementation Details

Feature-based Models. We use the implementations of scikit-learn (Pedregosa et al., 2011) for logistic regression and random forest experiments. For LambdaMART experiments, we use the XGBoost library (Chen and Guestrin, 2016). For both SCIQ and MCQL datasets we train with 500 rounds of boosting, step size shrinkage of 0.1, maximum depth of 30, minimum child weight of 0.1 and minimum loss reduction of 1.0 for partition. For calculating Wiki Sim features, we use a Wikipedia dump of Oct. 2016. Part of speech tags are calculated with NLTK (Bird and Loper, 2004).

The logistic regression used for the first stage ranker is based on features including: Emb Sim, POS Sim, ED, Token Sim, Length, Suffix, and Freq. Models for the second stage ranker is based on all features described in Sec. 2.1.1.

NN-based Models. Our NN-based models are implemented with TensorFlow (Abadi et al., 2016). When training the generator, we first uniformly select $K = 512$ candidates and then sample 16 distractors according to Equation 2. The temperature τ is set to 5. Our scoring functions are based on Decomposable Attention Model (Parikh et al., 2016). The word embeddings are initialized using the pre-trained GloVe (Pennington et al., 2014) (840B tokens), and the embedding size is 300. Our model is optimized using Adam algorithm (Kingma and Ba, 2015) with a learning rate of 1e-4 and a weight decay of 1e-6.

Since the sampling process in G is not differentiable, the gradient-decent-based optimization in the original GAN paper (Goodfellow et al., 2014) is not directly applicable. To tackle this problem, we use policy gradient based reinforcement learning as in IRGAN.

References

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: A system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283.

Manish Agarwal and Prashanth Mannem. 2011. Automatic gap-fill question generation from text books. In *BEA*, pages 56–64. ACL.

Steven Bird and Edward Loper. 2004. Nltk: the natural language toolkit. In *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions*, page 31. ACL.

Leo Breiman. 2001. Random forests. *Machine learning*, 45(1):5–32.

Chris J.C. Burges. 2010. From ranknet to lambdarank to lambdamart: An overview. Technical report.

Liwei Cai and William Yang Wang. 2018. Kbgan: Adversarial learning for knowledge graph embeddings. In *NAACL*. ACL.

Chia-Yin Chen, Hsien-Chin Liou, and Jason S Chang. 2006. Fast: an automatic generation system for grammar tests. In *Proceedings of the COLING/ACL on Interactive presentation sessions*, pages 1–4. ACL.

Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *KDD*, pages 785–794. ACM.

David Coniam. 1997. A preliminary inquiry into using corpus word frequency data in the automatic generation of english language cloze tests. *Calico Journal*, 14(2-4):15–33.

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*, pages 2672–2680.

Hubbard C Goodrich. 1977. Distractor efficiency in foreign language testing. *TESOL Quarterly*, pages 69–78.

Qi Guo, Chinmay Kulkarni, Aniket Kittur, Jeffrey P. Bigham, and Emma Brunskill. 2016. Questimator: Generating knowledge assessments for arbitrary topics. In *IJCAI*, pages 3726–3732.

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*, pages 23–30. ACL.

Shu Jiang and John Lee. 2017. Distractor generation for chinese fill-in-the-blank items. In *BEA@EMNLP*, pages 143–148. ACL.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Girish Kumar, Rafael E Banchs, and Luis Fernando D'Haro Enriquez. 2015. Revup: Automatic gap-fill question generation from educational texts. In *BEA*. ACL.

Chen Liang, Xiao Yang, Drew Wham, Bart Pursel, Rebecca Passonneau, and C. Lee Giles. 2017. Distractor generation with generative adversarial nets for automatically creating fill-in-the-blank questions. In *Proceedings of the Knowledge Capture Conference*, page 33. ACM.

Tie-Yan Liu et al. 2009. Learning to rank for information retrieval. *Foundations and Trends® in Information Retrieval*, 3(3):225–331.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119.

Ruslan Mitkov and Le An Ha. 2003. Computer-aided generation of multiple-choice tests. In *BEA@NAACL*, pages 17–22. ACL.

Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *EMNLP*, pages 2249–2255. ACL.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543. ACL.

Juan Pino and Maxine Eskenazi. 2009. Semi-automatic generation of cloze question distractors effect of students' l1. In *SLaTE*, pages 65–68.

Juan Pino, Michael Heilman, and Maxine Eskenazi. 2008. A selection strategy to improve cloze question quality. In *Proceedings of the Workshop on Intelligent Tutoring Systems for Ill-Defined Domains. 9th International Conference on Intelligent Tutoring Systems.*, pages 30–42.

Keisuke Sakaguchi, Yuki Arase, and Mamoru Komachi. 2013. Discriminative approach to fill-in-the-blank quiz generation for language learners. In *ACL*, volume 2, pages 238–242. ACL.

Katherine Stasaski and Marti A Hearst. 2017. Multiple choice question generation utilizing an ontology. In *BEA@EMNLP*, pages 303–312. ACL.

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*, pages 61–68. ACL.

Paul Viola and Michael Jones. 2001. Rapid object detection using a boosted cascade of simple features. In *CVPR*, volume 1, pages 511–518. IEEE.

Jun Wang, Lantao Yu, Weinan Zhang, Yu Gong, Yinghui Xu, Benyou Wang, Peng Zhang, and Dell Zhang. 2017. Irgan: A minimax game for unifying generative and discriminative information retrieval models. In *SIGIR*, pages 515–524. ACM.

Johannes Welbl, Nelson F. Liu, and Matt Gardner. 2017. Crowdsourcing multiple choice science questions. In *Proceedings of the 3rd Workshop on Noisy User-generated Text, NUT@EMNLP*, pages 94–106. ACL.

Torsten Zesch and Oren Melamud. 2014. Automatic generation of challenging distractors using context-sensitive inference rules. In *BEA*, pages 143–148. ACL.