Ranking Distractors for Multiple Choice Questions Using Multichannel Semantically Informed CNN-LSTM Networks

Manjira Sinha, Jatin Mandav and Tirthankar Dasgupta
[1]IIT Kharagpur, [2]IIIT Kalyani, [3]Tata Consultancy Services Ltd.
{manjira87, jatinmandav3, iamtirthankar}@gmail.com

Abstract

Automatically generating or ranking distractors for multiple-choice questions (MCQs) is still a challanging problem. In this work, we have focused towards automatic ranking of distractors for MCQs. Accordingly, we have proposed an semantically aware CNN-BiLSTM model. We evaluate our model with different word level embeddings as input over two different openly available datasets. Experimental results demonstrate our proposed model surpasses the performance of the existing baseline models. Furthermore, we have observed that intelligently incorporating word level semantic information along with context specific word embeddings boost up the predictive performance of distractors, which is a promising direction for further research.

Introduction

Multiple choice questions (MCQs) are widely used in many different teaching domains. Not only are they easy to score, students can answer them more quickly than typing answers to open response questions and are easier to evaluate through computer programs(Agarwal and Mannem 2011). Typically, an MCQ consist of a question (called the *stem*), the correct answer (or the *key*) and the alternate answers (called the distractors). Creating multiple-choice items is a challenging task, particular when it comes to distractor development. A good distractor distract students from the correct answer and restrict them to make any strategic guess about the correct answer(Gates 2011). Consequently, the distractors must be chosen in such a way so that they will be hard to distinguish from the correct answer, yet cannot be considered as correct answers. There are several guidelines describing features of generating good distractors along with good stems and alternatives (Al-Rukban 2006)(McMillan 2001)(McMillan 2001). For example, All distractors should be equally plausible, they should be homogeneous in content, they should be mutually exclusive, and must be free from clues about which response is correct.

Taking the aforementioned criterion into consideration, there have been many attempts to generate distractors from texts. The existing works spans across different domain and genre of texts, based on various similarity measures. These include WordNet-based metrics (Mitkov et al. 2009), embedding-based similarities (Guo et al. 2016)(Kumar, Banchs, and D'Haro 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 deep neural network based models to select distractors that resemble those in actual exam MCQs. More specifically, we propose a semantically informed multi-channel CNN-BiLSTM based hybrid architecture composed by a convolutional neural network(CNN) and a Bidirectional long short-term memory(BiLSTM) model, further enhanced by a novel inverted word-pair similarity module. The proposed distractor generation model is composed of two main components, i.e.,pair-wise word similarity matching and sentence modelling. The pair-wise similarity matching model is used to extract fine-grained similarity information between pairs of stem, key and distractor word sequences. We use a CNN to learn the patterns in the semantic correspondence between each pair of words in the two sentences that are intuitively useful. The idea to apply convolutions over a pair-wise word to word similarity matrix to extract the important word-word similarity pairs is motivated by how convolutions over text can extract the most important parts of a sentence. In sentence modelling architecture, we extract the local region information in form of important n-grams from the text using the CNN, and the long-term dependency information using the BiLSTM. By using this architecture, we are able to develop an informative semantic representation of each sentence. To demonstrate its robustness, we evaluated the proposed approach and compare it with the state-of-the-art models, using two different datasets - i.e., the RACE benchmark dataset, and SCiQ dataset.

The primary contributions of this paper are:

- We propose a novel deep neural network architecture leveraging coarse-grained sentence-level features and fine-grained word-level features for generating distractors for MCQs. The model combines sentence-level and word-level semantic similarity information and process them in a combined manner.

- We show how the proposed pair-wise similarity model can used to extract word-level semantic information, and demonstrate its usefulness in the distractor generation task.

- We evaluate our model by comparing with some of the most popular state-of-the-art neural network architecture including the BERT based classification. We have clearly observed that the semantic relatedness among words and sentence plays an important role in distractor ranking. Further, when the models are trained on individual datasets, the proposed architecture exceeds the performance of the baseline systems. However, as both the datasets are merged, BERT based ranking performs better.

Related Works

There exist a large body of work on automatic question generation specifically focusing on distractor generation or ranking (Al-Rukban 2006). Yet distractor generation still remains a difficult task. Attempts have been made to generate distractors for fill-in-the-blank questions using random selection from words in the same document (Hoshino and Nakagawa 2007), using thesaurus (Sumita, Sugaya, and Yamamoto 2005) (Gates 2011) (Agarwal and Mannem 2011) and collecting similar words in terms of their frequency. Often graphemic and phonemic cues plays important role in generation of distractor candidates (Pino and Eskenazi 2009). (Sakaguchi, Arase, and Komachi 2013) utilised common learner errors that were constructed from error-correction pairs on a language learning site. (Zesch and Melamud 2014) applied context-sensitive lexical inference rules to generate verb distractors that are not semantically similar to the target in the fill-in-the blank context but might be similar in another context. The advancement of Word2Vec models have also seen use of a semantic similarity measure based on the word2vec model for generating plausible distractors (Al-Rukban 2006)(Mitkov et al. 2009) (Araki et al. 2016) . Recently, a lot of efforts have been made on generating datasets for such a complex task using deep neural networks (Liang et al. 2017)(Liang et al. 2018) (Stasaski and Hearst 2017). For example, The SciQ dataset (Welbl, Liu, and Gardner 2017) consisting of 13.7K science MCQs collected through crowd-sourcing and covering subjects like, biology, chemistry, earth science, and physics. RACE (Lai et al. 2017) dataset contains 27,933 articles with 97,687 questions from English examinations of Chinese students from grade 7 to 12.

The Neural Network Architecture for Distractor Generation

In this section, we present the proposed Semantically informed Multi-Channel CNN-LSTM model for generating distractors. Following () we have also considered the problem as a ranking task. However, unlike the previous approach where only the stem, key, and the set of candidate distractors are provided as an input to the proposed model, we also provide the supporting text snippets to the model as input. These text snippets acts as a context based on which the candidate distractors are ranked. Based on the given input, the model learns to rank the candidate distractors according to their importance.

We represent the input to the model in terms of the following four tuples $< q_i, k_i, s_i, D_i >$ where, the *stem* (q), the key (k), supporting text (s) and a candidate distractor set D_i. Our task is to define a ranking function $R : (q_i, k_i, s_i, D_i) \rightarrow$ [0,1] such that *t* distractors in D_i are ranked higher than those in DD_i. Accordingly, we first convert each element of the tuples into a semantic representative vector, using the multichannel CNN+BiLSTM model. Then, an inverted pair wise semantic vector is computed by taking the element-wise inverted difference of each vector in the word sequence representations. The resulting difference is the discriminating representative vector of each of the q_i, k_i, s_i, D_i sentence pairs. This semantic vector is used as an additional feature vector for learning the inverted similarity between the two pairs of sentences. In addition to this coarse-level semantic information, we extract more fine-grained important information using a similarity matrix which contains word-to-word similarity quantification. Further convolutions are applied over the pair-wise similarity matrix to learn the similarity patterns between the words in the pair of sentences. The aim of the convolution function is to extract more fine-grained similarity features. Finally, a three layer fully connected neural network is used to produce the ranking from this concatenated feature vector. The first layers are activated by the ReLU [23] function, while we use the softmax link-function to the rest of the layers. We train the model to optimize binary cross-entropy. The overall architecture of the proposed model is depicted in Figure 1. In the following subsections we will discuss about the different modules of the proposed architecture.

Word Embedding Models

For the present work we have used different types of word-level vector representation models. Traditional methods like, word2vec (Mikolov et al., 2013), GloVe (Pennington et al., 2014) and FastText (Bojanowski et al., 2016) attempts to learn a n-dimensional representation for a word ignoring the surrounding context and the semantic sense. On the other hand, recent approaches like ELMo (Peters et al., 2018) and BERT (Devlin et al., 2018) have tried to overcome the aforementioned challenges by attempting to directly model the word's contextual sense into the vector representation. The pre-trained context-sensitive embedding of each word will be fed into the downstream neural network architecture, which is the distractor ranking in our case. Given a new do-

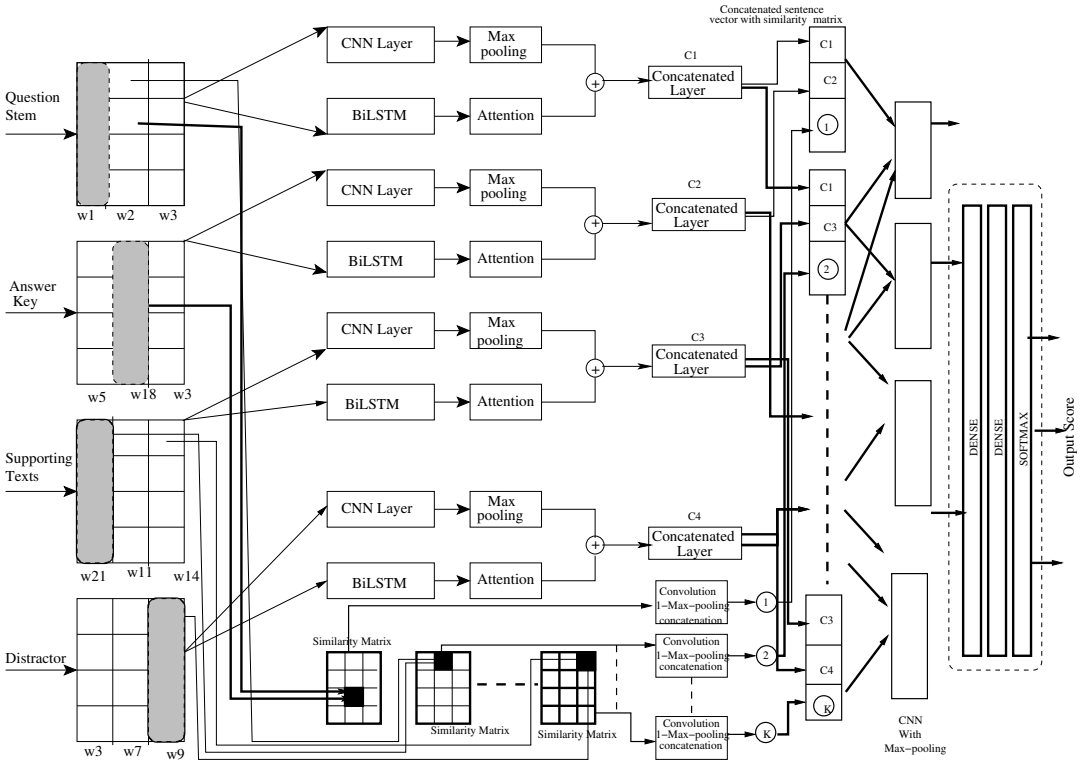

Figure 1: Overview of the Semantically informed multi-channel convolution recurrent neural network for distractor generation.

main of data, the downstream task tries to optimize the loss incurred during learning of the shared weights of the inner state of the pre-trained language model. All words that occur at least five times in the corpus are included and infrequent words are denoted as UNK. In our training dataset, we have encountered many such unknown words which are specific to some given domain such as Physics, Chemistry and Biology terms for which no word-level embeddings were available. To compensate for the loss we have also included the character-level embeddings.

In case of BERT we use the off-the-shelf pre-trained general domain $BERT_{BASE}$ model with the same additional CNN-LSTM network layers at the top of the architecture similar to the other embedding models. It has 12 layers of transformer blocks, 768 hidden units, and 12 self-attention heads. The pre-trained model are fine-tuned on all the three training dataset available to us.

Sentence modelling with CNN and LSTM

Once we generate both the sentence and word level embeddings, they form a one-dimensional array of representation. Each of the convolution network assumes a pre-defined sequence of n-gram words as input and perform a series of linear transformation. We represent every sentence using our joint CNN and LSTM architecture. The CNN is able to learn the local features from words to phrases from the text, while the LSTM learns the long-term dependencies of the text. The main goal of this step is to learn good intermediate semantic representations of the sentences, which are further used for the semantic similarity task. Let us assume a sequence of word $W = w_1, w_2...w_n$ where each w_i is represented with a word embedding of dimension d. Therefore, a sequence of n-words is represented by a $1D$ convolution network. The width of the convolution is k over the sentence. We apply the same convolution filter to each such window of size k in the sequence. This is done by computing a dot-product between the concatenation of the embedding vectors in a given window and a weight vector u. Accordingly, the convolution layer applies a linear transformation to all K windows in the given sequence of vectors. In order to ensure a uniform dimension over all sequences of input and output vectors, we perform a zero padding.

In the present distractor generation task, the neighbouring words in the form of context plays an important role. Occurrence of one word w_i largely depends upon the occurrence of its neighbouring words $w_{i-k}, w_{i-k-1}...w_{i-1}$. These factors can easily be handled by using the LSTM network structure. However, typical LSTMs allow the preceding elements to be considered as context, on the other hand, we have used the Bidirectional LSTM that considers both forward and backward sequence of elements (Zhou et al. 2016), (Hochreiter and Schmidhuber 1997). Thus, corresponding to each vector output generated from the preceding CNN layer, we parallaly determine the sequence level output representation from the Bi-LSTM units. Eventually, the last latent layer of the LSTM is taken as the semantic representation of the sentence, and the difference between element-wise difference between these representations is used as a semantic discrep-

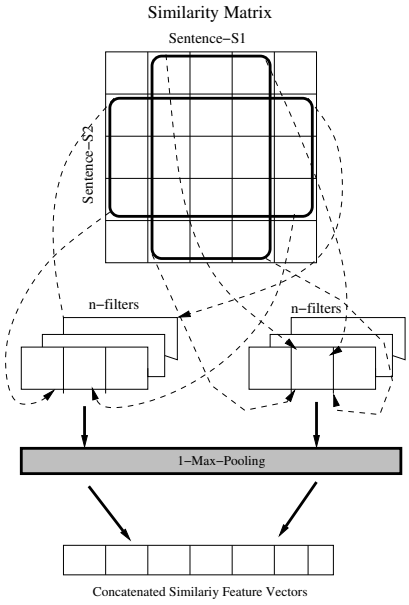

Figure 2: Generation of pair-wise word similarity matrix model

ancy measure at the level of the sentence pair.

Pair-wise word similarity matching

Apart from generating the sentence, word and character level representation, we also maintain an inverted semantic distance between elements of the MCQs. A pair-wise inverted similarity matrix is construed by computing the similarity of each word in $< q_i, k_i, s_i D_i >$ to another word. Convolutions are used to construct similarity matrix to analyze patterns in the pair-wise word to word similarities. Figure 2 illustrates this process. It is intuitive that given two element pairs, semantic correspondence between words provide important semantic information for detecting similar sentences, and the pair-wise word similarity matching model learns the word-level similarity patterns between the two sentences. We compute the dot product as a similarity measure between all the word embeddings of all the pairs of the input tuple sequence. Next, we try to learn any semantic correspondence patterns between the tuples by applying the CNN onto the inverted similarity matrices. For every similarity matrices we convolve over two directions that is, both from left to right and from top to bottom. This gives two separate convolution results, ζ_1 and ζ_2. We apply a global max-pooling over the resultant convolution layer. This helps us to obtain the most informative feature vectors from both ζ_1 and ζ_2. Finally, the resultant vectors are concatenated to produce the output from this module.

Learning the Network: The features generated from the penultimate layer of the multichannel CNN-LSTM layers are first concatenated with the output of the CNN based similarity matrix. The final concatenated outputs are then passed to a three layer fully connected neural network. The output of the fully-connected layer is the probability distribution of the candidate distractors that is used to produce the ranking from this concatenated feature vector. The first layer is activated by the ReLU function, while we use the softmax link-function to the rest of the layers. We take cross-entropy as the loss function that measures the discrepancy between the real annotated output and the model output distribution of sentences in the corpora

Experimentation and Evaluation

For GLOVe, FastText and ELMo we use the following hyper-parameters setting: hidden unit dimension at 512, dropout probability at 0.5, learning rate at 0.001, learning rate decay at 0.9, and Adam as the optimization algorithm. Early stopping of training is set to 5 epochs without improvement to prevent over-fitting. window size of 15, minimum word count of 5, 15 iterations, and embedding size of 300 to match the off-the-shelf embeddings.

Fine-tuing BERT Recent popularity of BERT based models also motivated us to develop a classifier using the same architecture. We use off-the-shelf pre-train BERT model. But before that We need to have some finer adjustments in order to use the pre-trained BERT model on the specific distractor generation task. Xavier initialization is utilized (Glorot and Bengio, 2010), without which the BERT fine-tuning failed to converge. Further, we set the early stopping of fine-tuning to 800 steps to prevent over-fitting. Finally, post-processing steps are conducted to align the BERT output with the concept gold standard, including handling truncated sentences and word-pieced tokenization.

Experiments

Dataset We evaluate the proposed distractor generation models on the following three datasets:

- The SciQ dataset (Welbl et al., 2017) consisting of 13.7K science MCQs collected through crowd-sourcing and covering subjects like, biology, chemistry, earth science, and physics. The questions span elementary level to college introductory level in the US.
- RACE (Lai et al. 2017) dataset containing 27,933 articles with 97,687 questions from English examinations of Chinese students from grade 7 to 12.

For all the datsets we divide them into a ratio of 80:10:10 for training, validation and testing purpose. For SciQ, we follow the original train/valid/test splits. We have also filtered out the distractors such as "all of them/the above", "none of them/the above", "both A and B", and so on. We have used regular expressions to identify these patterns and keep questions with at least one distractor. We use all the keys and distractors in the dataset as candidate set D. For both the dataset we extract each data sample and represented them in the form of four tuple $< s, k, c, D >$ of question stems, answer key, supporting text,and set of distractors.

Based on the given datasets, we have conducted three different experiments to identify the best model architecture for our task. .

In **Experiment-I:** We take each of the individual datasets and we divided them into three groups 80%, 10% and 10% for training, validation, and testing respectively.

Table 1: Results of experiments demonstrating ranking performance (rounded off to next integer) in terms precision (P@3), mean average precision (MAP@3), normalized discounted cumulative gain (NDCG@10), and mean reciprocal rank (MRR) across the three datasets.

		Dataset-1				Dataset-2			
	Embeddings	P@3	MAP@10	NDCG@10	MRR	P@3	MAP@3	NDCG@10	MRR
CNN	GLOVe	14	15	15	16	16	17	17	18
	FastText	17	18	19	19	18	20	20	21
	ELMo	18	18	19	21	18	16	17	21
BiLSTM	GLOVe	16	17	19	19	16	16	17	19
	FastText	19	19	22	22	17	17	18	20
	ELMo	34	37	38	37	18	17	19	21
CNN+ BiLSTM-att	GLOVe	27	27	29	27	27	26	28	32
	FastText	31	33	38	37	31	29	32	33
	ELMo	45	44	44	47	37	39	38	42
S-CNN+ BiLSTM-att	GLOVe	34	33	41	47	33	39	38	42
	FastText	47	45	45	51	37	39	46	48
	ELMo	53	47	49	60	49	53	56	58
BERT-base		51	43	45	57	45	51	55	55

Table 2: Results of Exp.-II combining Dataset-1 and 2.

	Embed-dings	P @3	MAP @10	NDCG @10	MRR
CNN	GLOVe	13	14	15	16
	FastText	17	14	16	19
	ELMo	16	0.41	16	16
BiLSTM	GLOVe	15	14	19	21
	FastText	21	26	26	29
	ELMo	20	24	023	25
CNN+ BiLSTM-att	GLOVe	19	23	25	27
	FastText	21	25	31	37
	ELMo	24	33	35	41
S-CNN+ BiLSTM-att	GLOVe	34	33	37	37
	FastText	44	43	45	47
	ELMo	**44**	49	48	49
BERT-base		47	**48**	**47**	**50**

In **Experiment-II:** We combine all the datasets together and formed a combined dataset of 117K questions. We then divided the entire corpus into 80%, 10% and 10% for training, validation, and testing respectively.

We use a host of standard neural network architectures like Convolutional Neural Network(CNN), Bi-directional LSTM (BiLSTM) and Generative Adversarial Network (GAN) as the baseline supervised models. Additionally, we have also used some unsupervised baseline models like, GLOVe, FastText and ELMo embedding distance scores. Each of the above neural network model were individually trained and tested with Glove vectors, FastText, ELMo and BERT embedding models. The evaluation is done based on the following parameters: precision (P@3), mean average precision (MAP@3), normalized discounted cumulative gain (NDCG@10), and mean reciprocal rank (MRR). Results of the experiments are reported in Table 1.

Results

In the case of **Experiment-I**, we found that throughout all the target classes the performance of the Semantically Aware CNN-BiLSTM with self Attention (S-CNN-BiLSTM-Att) model is significantly higher than the individual CNN, LSTM and the CNN-LSTM models. We have also explored the performance of our model with respect to the different embedding techniques. We have observed that context specific word embedding technique like, ELMO has been very effective in capturing contextual information. This is quite expected as , questions can be from multiple domains and therefore word/phrases from one domain may not be as important in another domain. Throughout all the models, we found that the performance of the S-CNN-BiLSTM-Att model along with the combined ELMo embedding is significantly higher than the rest of the existing models. During the analysis of the individual datasets we have observed that the S-CNN-BiLSTM-Att model for RACE (Dataset-2) has achieved the MRR of 58 as compared to the MRR of 60 for SciQ. One of the probable reason for such a low score can be due to the fact the the supporting texts for RACE are much larger in length as compared to the SciQ dataset. Consequently, semantic similarity score matrix between question, and answer pairs often gives incorrect results. This affects the overall ranking process. However, a detailed analysis of the difference in results are yet to be performed.

In **Experiment-II:** combining the two dataset together gives a slight edge of BERT based technique as compared to our proposed model. This is may be due to the diverse question answer domain where is BERT efficiently able to tune the parameter weights. Although the result is still below the highest scores achieved when the datasets are trained separately. Table 2 report the results obtained after combining all the training datasets together and testing the individual models.

Conclusion

Automatically ranking distractors for MCQs is still an open problem. Although a host of different linguistic as well as deep neural network based techniques have been applied to solve the problem, many of he fundamental issues still persists. Moreover, throughout the literature none of the existing systems are able to generate or rank the distractors with

very high accuracy. In this paper we have explored a number of deep neural network techniques and proposed a semantically aware multi-Channel network to rank distractors for MCQs. We have clearly observed that incorporating domain and semantic knowledge do help us to rank distractors better. Presently performance of our proposed model in terms of precision, MAP, NDCG and MRR exceeds that of the current baseline systems. However, a detailed analysis of the results as well as further improvement of the model remains a future work of this paper.

References

Agarwal, M., and Mannem, P. 2011. Automatic gap-fill question generation from text books. In *Proceedings of the 6th Workshop on Innovative Use of NLP for Building Educational Applications*, 56–64. Association for Computational Linguistics.

Al-Rukban, M. O. 2006. Guidelines for the construction of multiple choice questions tests. *Journal of family & community medicine* 13(3):125.

Araki, J.; Rajagopal, D.; Sankaranarayanan, S.; Holm, S.; Yamakawa, Y.; and Mitamura, T. 2016. Generating questions and multiple-choice answers using semantic analysis of texts. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, 1125–1136.

Gates, D. M. 2011. How to generate cloze questions from definitions: A syntactic approach. In *2011 AAAI Fall Symposium Series*.

Guo, Q.; Kulkarni, C.; Kittur, A.; Bigham, J. P.; and Brunskill, E. 2016. Questimator: Generating knowledge assessments for arbitrary topics. In *IJCAI-16: Proceedings of the AAAI Twenty-Fifth International Joint Conference on Artificial Intelligence*.

Hill, J., and Simha, R. 2016. Automatic generation of context-based fill-in-the-blank exercises using co-occurrence likelihoods and google n-grams. In *Proceedings of the 11th Workshop on Innovative Use of NLP for Building Educational Applications*, 23–30.

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Hoshino, A., and Nakagawa, H. 2007. Assisting cloze test making with a web application. In *Society for Information Technology & Teacher Education International Conference*, 2807–2814. Association for the Advancement of Computing in Education (AACE).

Jiang, S., and Lee, J. 2017. Distractor generation for chinese fill-in-the-blank items. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, 143–148.

Kumar, G.; Banchs, R.; and D'Haro, L. F. 2015. Revup: Automatic gap-fill question generation from educational texts. In *Proceedings of the Tenth Workshop on Innovative Use of NLP for Building Educational Applications*, 154–161.

Lai, G.; Xie, Q.; Liu, H.; Yang, Y.; and Hovy, E. 2017. Race: Large-scale reading comprehension dataset from examinations. *arXiv preprint arXiv:1704.04683*.

Liang, C.; Yang, X.; Wham, D.; Pursel, B.; Passonneaur, R.; and Giles, C. L. 2017. Distractor generation with generative adversarial nets for automatically creating fill-in-the-blank questions. In *Proceedings of the Knowledge Capture Conference*, 33. ACM.

Liang, C.; Yang, X.; Dave, N.; Wham, D.; Pursel, B.; and Giles, C. L. 2018. Distractor generation for multiple choice questions using learning to rank. In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, 284–290.

McMillan, J. H. 2001. Secondary teachers' classroom assessment and grading practices. *Educational Measurement: Issues and Practice* 20(1):20–32.

Mitkov, R.; Ha, L. A.; Varga, A.; and Rello, L. 2009. Semantic similarity of distractors in multiple-choice tests: extrinsic evaluation. In *Proceedings of the Workshop on Geometrical Models of Natural Language Semantics*, 49–56. Association for Computational Linguistics.

Pino, J., and Eskenazi, M. 2009. Semi-automatic generation of cloze question distractors effect of students' l1. In *International Workshop on Speech and Language Technology in Education*.

Sakaguchi, K.; Arase, Y.; and Komachi, M. 2013. Discriminative approach to fill-in-the-blank quiz generation for language learners. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 238–242.

Stasaski, K., and Hearst, M. A. 2017. Multiple choice question generation utilizing an ontology. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, 303–312.

Sumita, E.; Sugaya, F.; and Yamamoto, S. 2005. Measuring non-native speakers' proficiency of english by using a test with automatically-generated fill-in-the-blank questions. In *Proceedings of the second workshop on Building Educational Applications Using NLP*, 61–68. Association for Computational Linguistics.

Welbl, J.; Liu, N. F.; and Gardner, M. 2017. Crowdsourcing multiple choice science questions. *arXiv preprint arXiv:1707.06209*.

Zesch, T., and Melamud, O. 2014. Automatic generation of challenging distractors using context-sensitive inference rules. In *Proceedings of the Ninth Workshop on Innovative Use of NLP for Building Educational Applications*, 143–148.

Zhou, P.; Shi, W.; Tian, J.; Qi, Z.; Li, B.; Hao, H.; and Xu, B. 2016. Attention-based bidirectional long short-term memory networks for relation classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 207–212.