DisGeM: Distractor Generation for Multiple Choice Questions with Span Masking

Devrim Çavuşoğlu [*] [†] [‡] and **Seçil Şen** [*] [†] [§] and **Ulaş Sert** [†]
[†] OBSS AI
[‡] Middle East Technical University
[§] Bogazici University
first-name.last-name@obss.tech

Abstract

Recent advancements in Natural Language Processing (NLP) have impacted numerous sub-fields such as natural language generation, natural language inference, question answering, and more. However, in the field of question generation, the creation of distractors for multiple-choice questions (MCQ) remains a challenging task. In this work, we present a simple, generic framework for distractor generation using readily available Pre-trained Language Models (PLMs). Unlike previous methods, our framework relies solely on pre-trained language models and does not require additional training on specific datasets. Building upon previous research, we introduce a two-stage framework consisting of candidate generation and candidate selection. Our proposed distractor generation framework outperforms previous methods without the need for training or fine-tuning. Human evaluations confirm that our approach produces more effective and engaging distractors. The related codebase is publicly available at https://github.com/obss/disgem.

1 Introduction

Multiple-choice cloze tests are a prevalent form of assessment that not only evaluates a student's reading comprehension but also challenges their ability to deduce the most fitting option from a set of alternatives. Rooted in the concept of the traditional cloze test, where specific words are omitted from a passage and students are required to fill in the blanks with appropriate terms, the multiple-choice variant enhances the testing methodology by presenting a selection of potential options for each blank. This approach adds an element of complexity, requiring students to not only comprehend the context but also discern the most contextually appropriate answer among the provided choices. Central to the construction of multiple-choice cloze tests are the distractors – options deliberately crafted to divert students away from the correct answer. The creation of these distractors involves a careful balance of linguistic nuances while also appearing plausible enough to challenge the analytical skills of test-takers. In the realm of education, these tests serve as valuable tools for educators to gauge students' reading comprehension, critical thinking, and inference abilities, offering a holistic assessment of their grasp on the subject matter. We provided an example of a multiple-choice cloze style question in Table 1 from the CLOTH dataset (Xie et al., 2018).

Stem	If you are grateful, you naturally _____ yourself up to receive all kinds of blessings and good things in life.	
Options	**A. open**	→ Answer
	B. make	→ Distractor
	C. stand	→ Distractor
	D. take	→ Distractor

Table 1: A Cloze Test Example from CLOTH Dataset: The challenge of multiple-choice cloze test generation pertains to the creation of both plausible and reliable distractors.

Convincing distractor generation for short-form extractive multiple-choice questions (MCQ) in NLP remains an active research area, offering potential applications in educational assessments and question-generation tasks (Agarwal and Mannem, 2011). This paper presents a novel approach to generating distractors by leveraging the capabilities of Pre-trained Language Models (PLMs), specifically, those based on the Transformer architecture (Vaswani et al., 2017). Our proposed technique aims to generate distractors that closely resemble correct answers while maintaining semantic dissimilarity, all while utilizing publicly available pre-existing models like

[*]Equal contribution.

Findings of the Association for Computational Linguistics: EACL 2024, pages 9714–9732

BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019).

Unlike prior approaches that rely on parts-of-speech, recurrence, WordNet, or semantic analysis (Madri and Meruva, 2023), our approach utilizes Transformer-based PLMs to generate automated, diverse, and plausible distractors for short-form extractive MCQs without the need for additional model training. We make use of PLMs trained with the Masked Language Modeling (MLM) paradigm and Natural Language Inference (NLI), tasks for which extensive pre-trained models exist (Devlin et al., 2019; Liu et al., 2019; Dagan et al., 2005). MLM training allows these models to comprehend contextual information and generate coherent text, while NLI helps in maintaining semantic dissimilarity, creating distractors that closely mirror correct answers but differ in meaning.

MLM naturally aligns with distractor generation, as it draws inspiration from the Cloze task (Taylor, 1953). The Cloze task involves completing a text with omitted words based on contextual understanding, paralleling our use of MLM for distractor generation. By aligning these two concepts, we leverage MLM's innate strength in contextual completion to generate more relevant and coherent distractors.

Furthermore, we tackle a challenge presented by Transformer-based PLMs: their limitation in generating content within pre-existing text. While these models excel at MLM tasks, they require a predetermined token count for the masked region, lacking flexibility in dynamically determining the length of the generated text. This limitation poses a challenge since excessively long or short distractors may adversely impact their plausibility. To overcome this, we propose a system that allows for the generation of distractors of variable lengths, ensuring they remain within an acceptable range.

This study builds on the CDGP framework (Chiang et al., 2022) by introducing a new n-gram token generation procedure and refining the distractor selection method, outlined in Figure 1. The key contributions of our work are threefold. Firstly, we demonstrate that our system can produce a diverse set of distractors that exhibit similar characteristics to the correct answers, enhancing the sophistication of short-form extractive MCQs. Secondly, we emphasize that our method requires no specific training, making it easily applicable to various domains

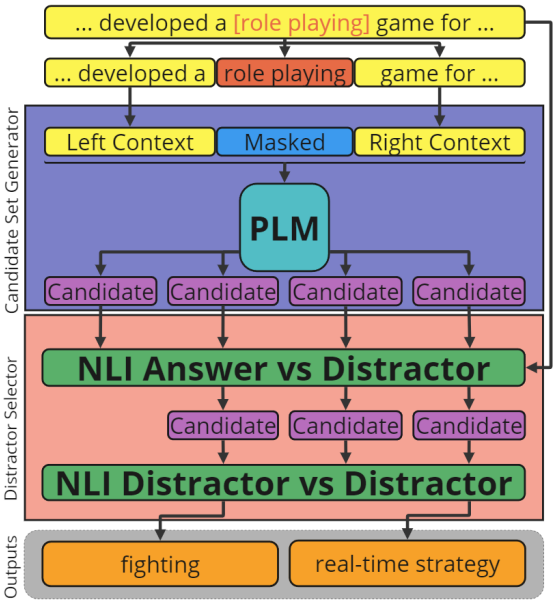

Figure 1: The overall architecture of DisGeM. Pre-trained Language Model generates candidates, which are then filtered with two NLI models to ensure consistency among the correct answer and distractors.

and languages without the need for extensive data collection. Thirdly, our framework offers a structured approach that can be readily extended or modified to accommodate different requirements, promoting adaptability and scalability. While our performance on automated metrics is comparable to the previous work, from the human evaluations we observe a clear preference toward our framework.

2 Related Work

Early NLP research focused on automatic question generation, with particular attention to Multiple-Choice Questions (MCQs), addressing the challenge of distractor generation, i.e. generating plausible distractors alongside correct answers (CH and Saha, 2020). Traditional methods, including semantic analysis and ontologies, were initially employed for distractor generation (Kumar et al., 2023; Ha and Yaneva, 2018; Faizan and Lohmann, 2018). Recent approaches leverage ontologies to select distractors based on semantic relationships (Ren and Q. Zhu, 2021; Liang et al., 2018).

Advancements in deep learning introduced end-to-end frameworks such as Bi-LSTM and sequence-to-sequence models for distractor generation (Qiu et al., 2020). Transformer models further improved distractor generation, with approaches like round-

trip neural machine translation (Panda et al., 2022) and one encoder, three decoders architecture (Maurya and Desarkar, 2020). Transformer fine-tuning strategies have also been applied (Chiang et al., 2022; Offerijns et al., 2020; Chung et al., 2020).

Two noteworthy frameworks emerged, both featuring a candidate set generator and a distractor selector (Ren and Q. Zhu, 2021; Chiang et al., 2022). Ren and Q. Zhu (2021) integrates knowledge bases, ensuring semantic and grammatical relatedness, and uses a learning-to-rank model for distractor selection. Building on this, Chiang et al. (2022) proposes a pre-trained language model-based distractor generator that outperforms the former. However, fine-tuning limitations and computational costs are noted. In contrast, our model harnesses pre-trained masked language models without requiring fine-tuning, offering a simpler and more practical solution.

3 Methodology

Our approach, building upon the CDGP framework (Chiang et al., 2022), is a dual-stage procedure: first, a candidate set generator (CSG) creates potential options, followed by a distractor selector (DS) that finalizes the distractor set. See Figure 1 for our framework's structure. Notably, instead of requiring a training regimen, our methodology leverages a pre-trained language model to produce candidates. In the second stage, these candidates undergo a meticulous two-step elimination process to select the final distractors. For a detailed view of the distractor generation pipeline and the algorithm, we refer readers to Appendix B.

3.1 Candidate Set Generator (CSG)

The first phase of our framework, the generation phase, generates distractor candidates using the source context S and the answer a and $a = [a_1||a_2||...||a_r]$ with a_i being the i^{th} token string of the answer string a, and $[\cdot||\cdot]$ denotes concatenation of strings. In this framework, the answer is assumed to be a span/substring of the given context, i.e. $a \in S$. To generate candidates we use any PLM trained with an MLM objective (Devlin et al., 2019; Liu et al., 2019; Lan et al., 2019; He et al., 2020). With a chosen model, this phase involves two steps: first, masking the tokens of the answer a, and second, generating multiple candidates from these inputs. This phase corresponds to the distractor generation stage represented by the blue area in Figure 1. Although similar to the CSG phase in the foundational CDGP framework (Chiang et al., 2022), DisGeM's CSG exhibits two noteworthy advantages: **(i)** it doesn't require fine-tuning, but allows it as an option for tailored applications (e.g., domain adaptation), and **(ii)** it can generate multi-word candidates, in contrast to CDGP's capacity limited to single-word candidates.

Unlike CDGP, our CSG is capable of generating multi-word/token candidates. However, a challenge arises when attempting to predict all mask tokens simultaneously. The model might produce suboptimal results because it lacks awareness of the best predictions for the nearby masked tokens. To address this, we patch an auto-regressive generation strategy and incorporate decoding techniques such as left-to-right (L2R) and right-to-left (R2L) decoding (Watanabe and Sumita, 2002). For example, with L2R decoding, tokens are generated in a left-to-right sequence one-by-one, so that every other token would be aware of the surrounding context and conditioned on the previous generations.

We employ a beam search (Graves, 2012) alike algorithm in our proposed decoding methods, we refer to it as the pseudo beam search. We use the term "pseudo" because we restrict initial predictions to the n most probable predictions, instead of spanning the entire vocabulary and for upcoming token predictions we restrict it to only top-1 prediction. For initial mask predictions, we constrain the search space to $(k \times m_s) \ll |V|$ which is much smaller than the entire vocabulary. For subsequent mask token predictions, we focus on the most probable outcomes. The source text S undergoes pre-processing, as detailed in Equation (1),

$$\begin{aligned} S &= [t_1, t_2, ..., a_1, a_2, ..., a_r, ..., t_{n-1}, t_n] \\ S' &= [t_1, t_2, ..., m_1, m_2, ..., m_r, ..., t_{n-1}, t_n] \end{aligned} \quad (1)$$

where $m_i = m \ \forall i = 1, 2, ..., r$ and $m =$ <*mask*> is the mask token. The generation/decoding strategy is an iterative process, and pseudo-formally, the generation of a candidate c given the source S and the answer a is given in Equation (2),

$$P(c_i|S, a) = \begin{cases} \underset{\pi}{\text{argmax}}\, P(m_1|S') & \text{if } i = 1 \\ \underset{\pi}{\text{argmax}}\, P(m_i|S'') & \text{otherwise} \end{cases} \quad (2)$$

where π denotes the probability distribution over the vocabulary, and S'' is S' where m_j is replaced by c_j at each step, i.e. $m_j := c_j \; \forall j < i$. Note that $c_{j<i}$ is the case for L2R decoding.

In the generation phase, we propose two generation hyper-parameters; *dispersion* and n_{mask} (optional). The n_{mask} parameter dictates the number of mask tokens that replace the removed answer tokens, in practice we used the number of answer tokens as n_{mask} by default. The *dispersion* parameter aims to enhance the diversity of generations by defining an interval for randomizing the number of mask tokens. In the pre-processing step (masking of the answer) we take *dispersion* into consideration and we define an interval $[\max(n_{mask} - dispersion, 1), n_{mask} + dispersion]$ from which random numbers are drawn for multiple times. This results in contexts with varying numbers of replacement mask tokens, producing potentially multiple input variations.

By unmasking the mask tokens sequentially, we condition each subsequent token prediction on previously generated token(s) along with the context, due to S'' being updated at each step. While auto-regressive models generate predictions sequentially, MLMs are designed to predict each token simultaneously — a key aspect of their pretraining objective. In our approach, we adapt an auto-regressive generation schema for masked Language Models (LM). This ensures that each generation is conditioned on previously generated **context**, leading to more natural and semantically coherent outputs. Upon multiple generations from multiple contexts (different mask token sizes), we rank these generations by their overall score which is the product of the probabilities along the generation, the score function is given in Equation (3).

$$T = \prod_{i=1}^{r} P(m_i|S'') \tag{3}$$

Ranking all the candidates by Equation (3) is, however, not viable since the token size r varies across generations (e.g. one context may use 3 mask tokens and another one may use 4). Applying the product of probabilities unfairly compares generations of different token lengths. To address this, we introduce a heuristic ranking score for candidates of any length using their probability scores: $T_{rank} = \sqrt[r]{T}$, which is a geometric mean of the probabilities of each token (in a step) along the generation. We opted for geometric mean as we have a multiplicative relation and it is more robust to outliers. In practice, we also performed experiments with harmonic mean, and harmonic mean, alternatively, can also be used in place of a geometric mean. The generation results by both averaging techniques are given in Table 2. The observations reveal slight differences between the averaging techniques. The final ranking of generated candidates is determined by T_{rank} with higher values indicating better candidates.

3.2 Generation Strategies

In the candidate generation step, we introduced the L2R and R2L decoding/generation strategies (Watanabe and Sumita, 2002). Utilizing different decoding strategies enables the diversification of the candidate set outputs.

While these strategies are intuitive choices for generations, we recognize the potential for further generation strategies. Masked LMs consider the entire context during the generation phase, encompassing both the left and right surrounding contexts of the mask tokens. Building on this, we introduce the *cocktail shaker decoding* (CTL) strategy, inspired by the cocktail shaker sort (Knuth et al., 1973)). Notably, the CTL decoding strategy differs from the bidirectional decoding strategy proposed in (Watanabe and Sumita, 2002). In particular, CTL alternates decoding the mask tokens with L2R and R2L step-by-step, rather than decoding L2R until the midway $\lceil \frac{m}{2} \rceil$ and R2L until the midway $\lfloor \frac{m}{2} \rfloor$ in reverse. For the detailed view, we refer readers to Appendix B.

We also provided qualitative examples of generations from a SQuAD (Rajpurkar et al., 2016) sample, showcasing different decoding strategies in Table 2. It can be easily seen that the differences between the decoding strategies are more significant compared to the averaging techniques.

3.3 Distractor Selector (DS)

After receiving the outputs from CSG, the pipeline proceeds with the equally crucial Distractor Selector (DS) phase. This phase aims to refine the candidate set by eliminating undesired candidates. While the CSG phase is adept at generating a diverse set of candidates, it doesn't guarantee the suitability of these candidates as true distractors. There are primarily two main reasons behind that:

Passage	Tesla was born on 10 July [O.S. 28 June] 1856 into a Serb family in the village of Smiljan, Austrian Empire (modern-day Croatia). His father, Milutin Tesla, was a Serbian Orthodox priest. Tesla's mother, Đuka Tesla (née Mandić), whose father was also an Orthodox priest,:10 had a talent for making home craft tools, mechanical appliances, and the ability to memorize Serbian epic poems. Đuka had never received a formal education. Nikola his eidetic memory and creative abilities to **his mother's genetics** and influence. Tesla's progenitors were from western Serbia, near Montenegro.:12		
Question	Who did Tesla credit for his abilities?		
Geometric Mean	**L2R**	**R2L**	**CTL**
1	the family's wealth	enhance his sense of power	his own ageing, knowledge
2	his mother's education	have all his knowledge, power	his Serb religious upbringing
3	his own personal experience	preserve the Serbian tradition	his own Serbian language skills
4	his family tradition knowledge	be a combination of energy	the influence of Serbian culture
Harmonic Mean	**L2R**	**R2L**	**CTL**
1	the family's wealth	enhance his sense of power	his Serb religious upbringing
2	his own personal experience	have all his knowledge, power	his own ageing, knowledge
3	his family tradition knowledge	increase both his knowledge	his own Serbian language skills
4	his mother's education	be his sources of knowledge	the influence of Serbian culture

Table 2: Outputs from different decoding strategies and different averaging techniques. The answer is marked in bold font.

- The candidates might mirror the ground truth answer, either verbatim or in essence. This overlap will render a question invalid, as introduces multiple correct answers.
- The candidates themselves might mirror each other, such as presenting analogous distractors. While this doesn't outright invalidate a question, it undermines the quality of the distractors as a whole. By assuming only a single choice is the correct answer, a respondent can easily deduce duplicate choices as incorrect.

To address these challenges, our DS employs a language model fine-tuned on a downstream task of Natural Language Inference (NLI), sometimes also referred to as Recognizing Textual Entailment (RTE) (Dagan et al., 2005)). This approach is markedly different from the DS in CDGP (Chiang et al., 2022), which relies on FastText (Bojanowski et al., 2017) word embedding model. For our experiments, we used a publicly available BART model[1] (Lewis et al., 2020) fine-tuned on the NLI task. While this model is a three-way classifier for *entailment*, *neutral* and *contradiction*, a two-way NLI model that distinguishes between *entailment* and *contradiction* could also be used.

The elimination process consists of two steps, directly addressing the two aforementioned challenges regarding the generated candidates. This phase is illustrated in Figure 1, highlighted with a red background.

[1]https://huggingface.co/geckos/bart-fined-tuned-on-entailment-classification

For the first step, the aim is to eliminate the candidates that share identical or similar meanings with the ground truth answer. To achieve this, we compare the source context S (containing the ground truth answer) against the modified context where the ground truth answer is replaced with a candidate. Using the chosen NLI model, we then eliminate any candidate if the model's output is *entailment*. This ensures that only candidates that either contradict or remain neutral in meaning compared to the ground truth answer are retained.

The second stage of the elimination process takes the filtered outputs of the first step as its starting point. In this phase, our goal is to ensure diversity among the candidates, eliminating any that share similar meanings, as this similarity would be undesirable within the final set of distractors. To tackle this challenge, we once again employ the NLI model. However, the difference this time is that we compare contexts populated with different candidates against each other. If the NLI model's result for any pair of contexts is *entailment*, we discard the candidate with the lower score.

Following the second step of elimination, we are left with our finalized distractors. It is crucial to note that the NLI model we employed is trained on sentences rather than passages. Therefore, the input is formatted as " $< sentence_A >< sentence_B >$ ". To ensure thoroughness, we utilize a two-way entailment classification in our approach. That is, we provide the model with permutations of input: " $< sentence_A >< sentence_B >$ " and " $< sentence_B >< sentence_A >$ ". In this setup, an

entailment is only when both results are *entailment*.

4 Experiments

We conducted two primary evaluations to assess our distractor generation method: Instruction-following Large Language Models (LLMs) evaluations and human evaluations. For LLM evaluations, we used SQuAD context/answer pairs to test different generation hyperparameters and compared the best to the previous work. For human evaluations, we used questions from the CLOTH dataset, comparing gold distractors, our generated distractors, and those from previous work, and measured human accuracy and quality ratings. Additional quantitative results from experiments on the CLOTH dataset are provided in Appendix C.

4.1 LLM Evaluations

To evaluate the effectiveness of our distractor generation method, we used 100 random context/answer pairs from the SQuAD dataset. This dataset provides a diverse set of context passages and corresponding answers, which we utilized to create question/answer pairs. These pairs were then used to generate distractors with our method under different hyperparameter settings and compared against distractors generated by previous methods.

The evaluation involved the following steps:

1. Question Selection: We randomly selected 100 pairs from the SQuAD dataset to ensure a representative sample. Using these pairs, we created fill-in-the-blank questions. This question set has been kept constant across different experiments.

2. Distractor Generation: We generated distractors using our method with various hyperparameter settings (n_{mask}, $dispersion$, decoding strategy) and compared them between each other and against distractors generated by previous methods.

3. Best Set Selection by LLM: We employed ChatGPT-4o (OpenAI, 2024) to select the best distractor set from the given sets (including ours and previous methods). Ties were allowed if the best options were close in quality.

Parameter	Score
Number of Mask Tokens	
$n_{\text{mask}} = 0$	**57**
$n_{\text{mask}} = 1$	53
Dispersion Parameter	
$dispersion = 0$	47
$dispersion = 1$	**56**
$dispersion = 2$	50
Decoding Strategy	
L2R (Left-to-Right)	44
R2L (Right-to-Left)	29
CTL (Cocktail Shaker)	**52**

Table 3: Scores for various hyperparameters used in distractor generation: the number of mask tokens (n_{mask}), $dispersion$ parameters, and decoding strategies. Each score reflects the preference of the referee LLM, ChatGPT-4o, for that hyperparameter in direct comparisons.

4.1.1 Hyperparameter Experiments

To identify the optimal settings for generating distractors, we experimented with different hyperparameters by systematically varying one parameter at a time while keeping the others constant. For distractor generation, we utilized RoBERTa$_{\text{LARGE}}$ (Liu et al., 2019), a widely recognized pre-trained transformer model.

- First, we experimented with the number of mask tokens. We tested with $n_{\text{mask}} = 1$ and $n_{\text{mask}} = 0$ (where the number of mask tokens matched the number of answer tokens). For these experiments, the $dispersion$ parameter was set to 0, and the decoding strategy was left-to-right (L2R).

- Next, we explored the impact of the $dispersion$ parameter. We experimented with $dispersion$ values of 0, 1, and 2. In these experiments, we kept $n_{\text{mask}} = 0$ and continued using the L2R decoding strategy.

- Finally, we evaluated different decoding strategies to determine their effect on distractor quality. We compared left-to-right (L2R), right-to-left (R2L), and cocktail shaker (CTL) strategies. For these experiments, we set $n_{\text{mask}} = 0$ and $dispersion = 1$.

4.1.2 Results

The results indicated several key findings, as summarized in Table 3.

Method	Score
CDGP	12
DisGeM	**88**

Table 4: Final scores for distractor generation methods: CDGP vs. DisGeM. Each score reflects the preference of the referee LLM, ChatGPT-4o, in direct comparisons.

Distractors	CDGP	DisGeM	GOLD
Average	81.00	63.62	76.28
Median	83.33	66.67	76.47
Min.	55.56	38.89	47.06
Max.	94.44	83.33	100.00
Std. Dev.	12.81	10.91	11.11

Table 5: Statistics for the correctness of the evaluators. “GOLD" represents questions with ground truth distractors.

Experimenting with the number of mask tokens showed that using zero mask tokens, where the number of mask tokens matched the number of answer tokens, generally led to better distractor quality compared to using a single mask token.

When varying the dispersion parameter, we found that a moderate level of randomness in the number of mask tokens improved distractor quality. A dispersion parameter of 1 was more effective than having no dispersion or a higher level of dispersion.

Among the different decoding strategies, the Cocktail Shaker (CTL) strategy performed the best, followed by the Left-to-Right (L2R) and then the Right-to-Left (R2L) strategy. This indicates that the CTL strategy is more effective in generating high-quality distractors.

Overall, the experiments demonstrated that the optimal configuration for generating high-quality distractors involves using a matching token count, a dispersion parameter of 1, and the Cocktail Shaker (CTL) decoding strategy. This configuration was compared to the previous work CDGP (Chiang et al., 2022), using their experiment setup from the CDGP codebase[2]. We conducted a final LLM comparison with their distractors versus our distractors on the question set. The results, shown in Table 4, indicate that our method, DisGeM, significantly outperformed CDGP.

4.2 Human Evaluation

As previous works (Ren and Q. Zhu, 2021; Chiang et al., 2022) have done, we also conducted human evaluation. For the evaluation process, we recruited 30 human evaluators, and we asked evaluators to take a cloze exam and rate the questions. Our evaluation process follows the evaluation setup of Chiang et al. (2022) with the following differences,

- We prepared a cloze test with three passages randomly chosen from the CLOTH dataset, each passage is evenly split into three parts for ground truth, CDGP and DisGeM distractors.
- Unlike CDGP’s ordering where the first 5 questions were ground truth and the last 5 questions were CDGP generations, we randomly shuffled the order of the questions to eliminate any possible bias from human evaluators.
- We included 17 questions in total for each distractor source (ground truth, CDGP, DisGeM) in three passages.
- We incorporated a simultaneous cloze test where we also gathered feedback regarding the quality and difficulty after each question. In this context, "difficulty" pertains to the extent of uncertainty or wavering among multiple answer choices (i.e. how well the distractors distract the evaluator). Here, we compounded the quality and difficulty assessment as a single feedback and asked the evaluators to rate on a Likert scale ranging from 1-5.

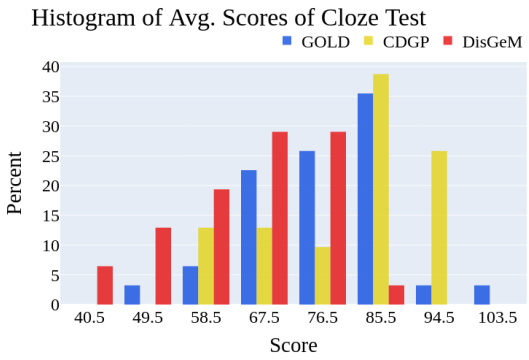

Figure 2: Average human correctness of the cloze test grouped by the frameworks, “GOLD" represents the questions with ground truth distractors.

4.2.1 Results

We illustrate the histograms[3] of the results of the cloze test and the results of the ratings by evaluators

[2]https://github.com/AndyChiangSH/CDGP

[3]We used the Plotly package (Inc., 2015) for illustrations.

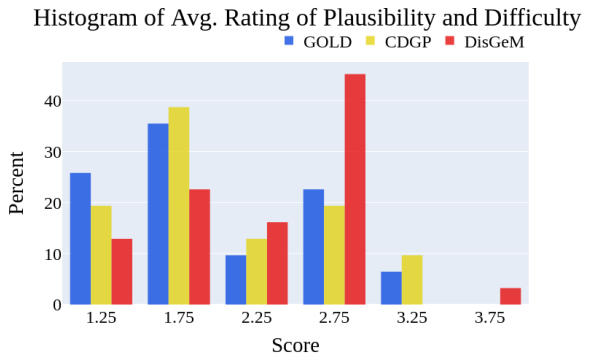

Figure 3: Average rating of the quality and difficulty of the questions by the evaluators, “GOLD" represents the questions with ground truth distractors.

grouped by the three groups in Figure 2 and in Figure 3 respectively. Detailed statistics are given in Table 5. The histograms reveal a noticeable trend in the rate of correctness of evaluators is comparatively lower on DisGeM’s questions than on those from GOLD and CDGP. Also, more evaluators voted higher for DisGeM in quality and difficulty rating of questions/distractors compared to GOLD and CDGP.

For the assessment of the statistical significance we also conduct a Student’s t-test to show that the question group of DisGeM is slightly harder than that of GOLD (i.e. the ground truth distractors). Chiang et al. (2022) has already stated in their human evaluation that their questions/distractors are slightly easier from the GOLD. Our human evaluation results are also parallel to this conclusion. Hence, we moved forward to compare DisGeM and GOLD groups. We conducted Student’s t-test (Student, 1908) under the null hypothesis $\mu_{disgem} - \mu_{gold} < 0$, and get the p-value 0.999, and thus we have no significant evidence to reject the null hypothesis. Also, the extremity of the p-value suggests that the null hypothesis is likely to be the case.

5 Discussion

The evaluation results from both LLM and human evaluations provide significant insights into the effectiveness of our distractor generation method, DisGeM, compared to previous methods. In the LLM evaluations, we observed that the optimal configuration of a matching token count, a dispersion parameter of 1, and the Cocktail Shaker (CTL) decoding strategy showed a significant improvement over CDGP, as indicated by the final comparison scores.

In human evaluations, the results were similarly encouraging. The cloze test scores revealed that evaluators had a slightly lower correctness rate on DisGeM’s questions compared to GOLD and CDGP. This lower correctness rate indicates that DisGeM’s distractors were more effective at introducing uncertainty and challenging the evaluators. The quality and difficulty ratings from evaluators were also higher for DisGeM, suggesting that our method produces distractors that are perceived as more engaging and effective at creating uncertainty.

6 Conclusion

We propose a training-free dual-stage distractor generation framework, DisGeM. Our framework possesses two main features extending previous works. DisGeM in particular,

- is a training-free framework and does not necessitate fine-tuning on a particular dataset though one can opt for various reasons (e.g. domain adaptation).
- has a novel multi-word distractor generation algorithm.
- has far less amount of generation hyper-parameters that need to be tuned by users/researchers.
- incorporate strategies (e.g. decoding, multi-token generation) to achieve natural variation in the generations.

Also, due to the training-free nature of our framework can directly be used on any passage without the effort of additional training/fine-tuning, and hence saves a lot of time. Additionally, it alleviates the heavy hyper-parameter tuning stage.

Our evaluations with LLM and human participants demonstrate DisGeM’s effectiveness. LLM evaluations showed DisGeM outperforms previous methods. Human evaluations confirmed this with lower correctness rates and higher quality and difficulty ratings, indicating more effective and engaging distractors.

Limitations

We have so far laid out results along with a discussion on them. Although the results were promising there are several limitations to the

proposal. Like its predecessors (Ren and Q. Zhu, 2021; Chiang et al., 2022), DisGeM is also proposed as a distractor generation framework for extractive MCQs and for English only, and experiments are conducted according to the extractive MCQ setup. The framework on its own is not capable of working on an abstractive MCQ setting, where the answer is not necessarily a span/substring in the context, as it depends on the masking of the answer span in the context. Nonetheless, this might be achieved in future research that enhances the CSG phase.

Another limitation of DisGeM similar to the prior studies (Ren and Q. Zhu, 2021; Chiang et al., 2022) is that there is no control of the difficulty level for generated distractors. Yet, this might be a good research direction for related future studies as it may have a positive impact on real-life quiz applications.

Finally, we would like to point out that we did not conduct any comparison with the instruction following LLMs or PLMs having parameters many times more than our PLMs in experiments (e.g. (Brown et al., 2020; Ouyang et al., 2022; Touvron et al., 2023; OpenAI, 2024)). The primary reason for not comparing with those kinds of models was the absence of a standardized methodology for employing such models as baseline performance. The performance of these models can depend on prompt engineering, model versions, and whether the approach is zero-shot or fine-tuned. Also, general models like those may generate 'hallucinated' content, which requires additional layers of validation. Furthermore, a kind of comprehensive evaluation is needed to compare against those models which requires significant time investment.

Apart from the limitations discussed, there are also potential risks that could primarily affect education. While the proposed framework may be promising in experiments, these experiments can not provide a full-picture overview. Using this framework in real-life quizzes and tests may not give a correct assessment of the test takers. Thus, the framework should be used under the guidance of a human expert (e.g. teacher) in those scenarios. This is further amplified by our lack of control over how difficult the distractors are.

References

Manish Agarwal and Prashanth Mannem. 2011. Automatic gap-fill question generation from text books. In *Proceedings of the Sixth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 56–64, Portland, Oregon. Association for Computational Linguistics.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Dhawaleswar Rao CH and Sujan Kumar Saha. 2020. Automatic multiple choice question generation from text: A survey. *IEEE Transactions on Learning Technologies*, 13(1):14–25.

Shang-Hsuan Chiang, Ssu-Cheng Wang, and Yao-Chung Fan. 2022. CDGP: Automatic cloze distractor generation based on pre-trained language model. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 5835–5840, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Ho-Lam Chung, Ying-Hong Chan, and Yao-Chung Fan. 2020. A BERT-based distractor generation scheme with multi-tasking and negative answer training strategies. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4390–4400, Online. Association for Computational Linguistics.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The pascal recognising textual entailment challenge. In *Proceedings of the First International Conference on Machine Learning Challenges: Evaluating Predictive Uncertainty Visual Object Classification, and Recognizing Textual Entailment*, MLCW'05, page 177–190, Berlin, Heidelberg. Springer-Verlag.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Ainuddin Faizan and Steffen Lohmann. 2018. Automatic generation of multiple choice questions from slide content using linked data. In *Proceedings of the 8th International Conference on Web*

Intelligence, Mining and Semantics, WIMS '18, New York, NY, USA. Association for Computing Machinery.

Alex Graves. 2012. Sequence transduction with recurrent neural networks. *arXiv preprint arXiv:1211.3711*.

Le An Ha and Victoria Yaneva. 2018. Automatic distractor suggestion for multiple-choice tests using concept embeddings and information retrieval. In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 389–398, New Orleans, Louisiana. Association for Computational Linguistics.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2020. Deberta: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654*.

Plotly Technologies Inc. 2015. Collaborative data science.

Donald Ervin Knuth et al. 1973. *The art of computer programming*, volume 3. Addison-Wesley Reading, MA.

Archana Praveen Kumar, Ashalatha Nayak, Manjula Shenoy K., Shashank Goyal, and Chaitanya. 2023. A novel approach to generate distractors for multiple choice questions. *Expert Syst. Appl.*, 225(C).

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Chen Liang, Xiao Yang, Neisarg Dave, Drew Wham, Bart Pursel, and C. Lee Giles. 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*, pages 284–290, New Orleans, Louisiana. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Vijaya Raju Madri and Sreenivasulu Meruva. 2023. A comprehensive review on mcq generation from text. *Multimedia Tools and Applications*, pages 1–20.

Kaushal Kumar Maurya and Maunendra Sankar Desarkar. 2020. Learning to distract: A hierarchical multi-decoder network for automated generation of long distractors for multiple-choice questions for reading comprehension. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, CIKM '20, page 1115–1124, New York, NY, USA. Association for Computing Machinery.

Jeroen Offerijns, Suzan Verberne, and Tessa Verhoef. 2020. Better distractions: Transformer-based distractor generation and multiple choice question filtering.

OpenAI. 2024. Hello gpt-4o. Accessed: 2024-06-15.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.

Subhadarshi Panda, Frank Palma Gomez, Michael Flor, and Alla Rozovskaya. 2022. Automatic generation of distractors for fill-in-the-blank exercises with round-trip neural machine translation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 391–401, Dublin, Ireland. Association for Computational Linguistics.

Zhaopeng Qiu, Xian Wu, and Wei Fan. 2020. Automatic distractor generation for multiple choice questions in standard tests. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2096–2106, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Siyu Ren and Kenny Q. Zhu. 2021. Knowledge-driven distractor generation for cloze-style multiple choice questions. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(5):4339–4347.

Student. 1908. The Probable Error Of a Mean. *Biometrika*, 6(1):1–25.

Wilson L. Taylor. 1953. "cloze procedure": A new tool for measuring readability. *Journalism Quarterly*, 30(4):415–433.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Taro Watanabe and Eiichiro Sumita. 2002. Bidirectional decoding for statistical machine translation. In *COLING 2002: The 19th International Conference on Computational Linguistics*.

Qizhe Xie, Guokun Lai, Zihang Dai, and Eduard Hovy. 2018. Large-scale cloze test dataset created by teachers. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2344–2356, Brussels, Belgium. Association for Computational Linguistics.

A Experimental Setup

The hardware specification that is used for conducting experiments are given as follows:

- CPU: AMD Ryzen 9 7900X 12-Core Processor
- GPU: NVIDIA RTX 3090 Ti

B Pipeline

We present the detailed view and the algorithm on our distractor generation pipeline. We describe pseudo-code algorithms for both CSG and DS phases.

B.1 Candidate Set Generator (CSG)

Our CSG phase of the framework is detailed in Algorithm 1. This phase highlights how our CSG differs from the previous work, CDGP. The candidate set generation phase utilizes methods such as randomly sampling the number of tokens with provided hyper-parameters (e.g. $dispersion$, n_{mask}). Our pseudo-beam search operates with a fixed window size of 1, after the first step of $k \times m_s$ generations. In the algorithm, we designed the search multiplier m_s to allow users to adjust it based on their preferred trade-off between speed and fidelity. In practice, we set m_s to 10 for the single-mask cases (like CLOTH outputs) when there is only one token and set it to 7 other scenarios.

B.2 Distractor Selector (DS)

The pipeline details for the distractor selection phase can be found in Algorithm 2. As previously mentioned, our approach utilizes a two-way entailment check. In our context, checking for entailment from both the answer to the distractor and the distractor to the answer is undesirable for distractor generation.

B.3 Decoding Strategies

We have conducted experiments and discussed the outputs in Section 3. The schema in Figure 4 illustrates various generation strategies, including our newly proposed cocktail shaker (CTL) decoding strategy.

C Quantitative Results

Following the previous works (Ren and Q. Zhu, 2021; Chiang et al., 2022) we evaluated our framework with automated metrics. We used the experiment setup from CDGP (Chiang et al., 2022) codebase[4].

C.1 Dataset

Following (Chiang et al., 2022) we evaluated our framework on CLOTH dataset (Xie et al., 2018). We used the instances from the test split (high) of the CLOTH dataset to evaluate our framework. Since we did not conduct any fine-tuning, we only focused on the test split.

CLOTH is a dataset curated by teachers and consists of passages/paragraphs containing cloze-style questions where the answers are one word and with 4 options in MCQ style. The distractors/options in MCQ The statistics regarding the test split (high) of CLOTH are given in Table 6.

C.2 Evaluation Metric

We adhere to the evaluation metric setup established in (Ren and Q. Zhu, 2021; Chiang et al., 2022), assessing Precision (P@1), F1 score (F1@3), Mean Reciprocal Rank (MRR@10), and Normalized Discounted Cumulative Gain (NDCG@10).

C.3 Experimental Setup

We used $\text{BERT}_{\text{LARGE}}$ (Devlin et al., 2019) and $\text{RoBERTa}_{\text{LARGE}}$ (Liu et al., 2019) as PLMs for the CSG phase to conduct our experiments following (Chiang et al., 2022). Since CLOTH dataset

[4]https://github.com/AndyChiangSH/CDGP

Algorithm 1: Overall CSG pipeline. Indexing is assumed to be starting from 0.

Data: P passage, A answer, M_{PLM} Language Model (pretrained with MLM task), T_{PLM} tokenizer of M_{PLM}, k number of distractors, s decoding strategy, d dispersion, m_s search multiplier, n_{mask} number of mask tokens, avg averaging technique (geometric or harmonic)
Result: CS candidate set
$P_T \leftarrow T_{PLM}(P)$; // tokenize passage
if $n_{mask} == 0$ **then**
  $n_{mask} \leftarrow |A_T|$
end
$N \leftarrow draw\_three(a = \max(n_{mask} - dispersion, 1), b = n_{mask} + dispersion, replace = False)$;
$CS \leftarrow [\,]$;
$RS \leftarrow [\,]$;
for $i \leftarrow 0$ *to* $|N| - 1$ **do**
  $M_T \leftarrow [< mask >_0, \ldots, < mask >_{N_i - 1}]$;
  $P_T \leftarrow replace(A_T, M_T, P_T)$; // replace answer tokens by <mask> tokens
  $m \leftarrow k * m_s$;
  $C \leftarrow empty\_array(m, |M_T|)$;
  $R \leftarrow empty\_array(m, |M_T|)$;
  $P'_T \leftarrow tile\_array(P_T, m)$; // P'_T is a 2 dim array of size $(m, |P_T|)$
  for $j \leftarrow 0$ *to* $|M_T| - 1$ **do**
    if $j == 1$ **then**
      $t = m$;
    else
      $t = 1$;
    $C_{\cdot j}, R_{\cdot j} \leftarrow M_{PLM}(P'_T, token = M_{T_j}, top_k = t)$; // get t predictions for j^{th} token of M_T
    $P'_T \leftarrow replace(M_{T_j}, C_{\cdot j}, P'_T)$;
  end
  $C \leftarrow concat\_strings(C, axis = 1)$;
  $R_{prod} \leftarrow product(R, axis = 0)$;
  $C \leftarrow sort(C, by = R_{prod}, descending = True)$;
  $add(C, CS)$;
  $add(R, RS)$; // RS and CS are 2 dim arrays with shapes $(3m, |M_T|)$
end
$R_{avg} \leftarrow average(RS, type = avg, axis = 0)$;
$CS \leftarrow sort(C, by = R_{avg}, descending = True)$;
return CS;

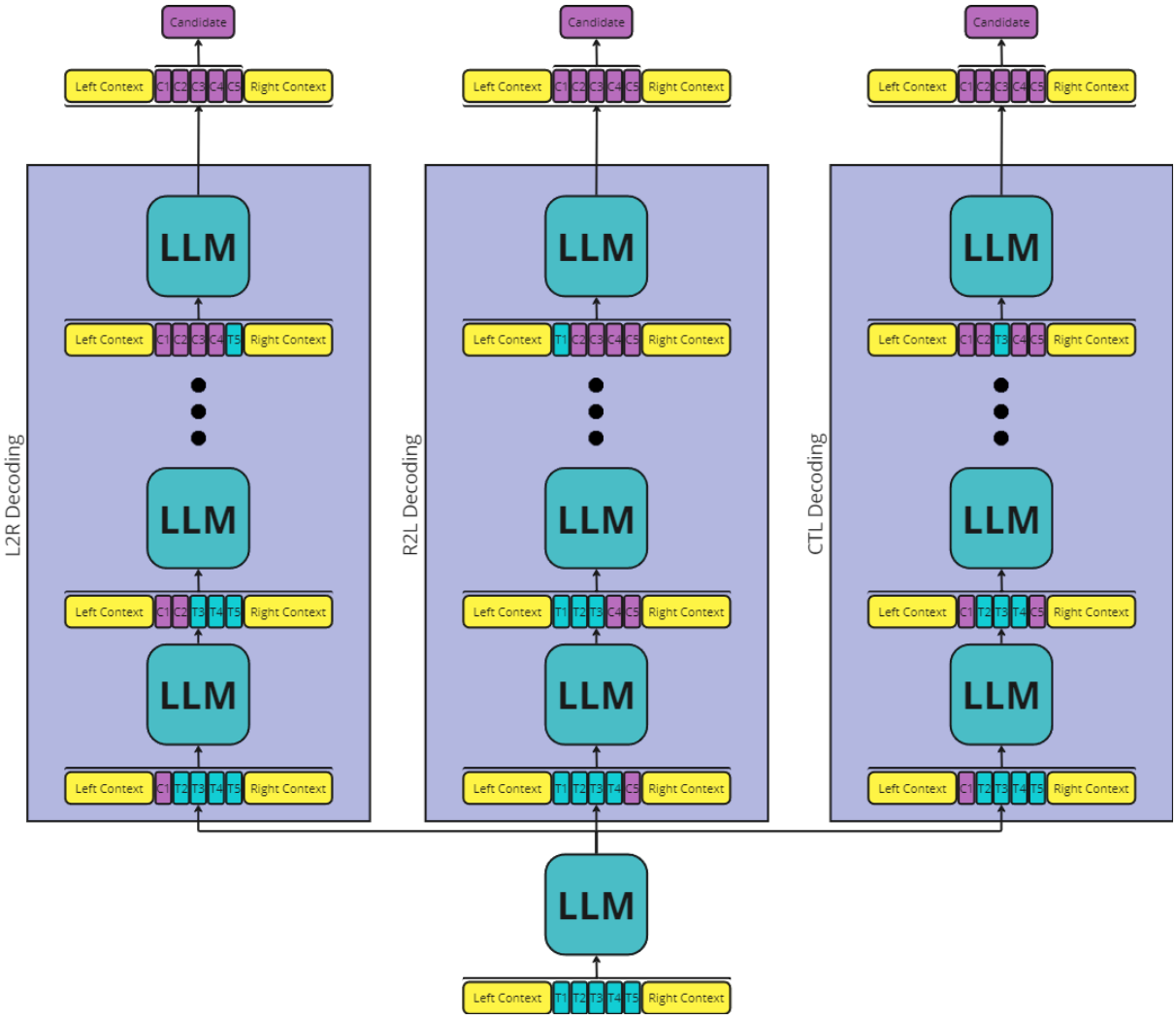

Figure 4: Different strategies proposed for generating candidates. The prediction orders for mask tokens are **(left) L2R** 1-2-3-4-5, **(middle) R2L** 5-4-3-2-1, **(right) CTL** 1-5-2-4-3. “T" in blue refers to mask tokens. “C" in purple refers to the candidate tokens.

Algorithm 2: Overall DS pipeline. Indexing is assumed to start from 0. $|\cdot|$ refers to the norm/length of a component.

Data: S sentence, A answer, C candidates by CSG, M_{NLI} NLI model, k number of distractors
Result: D distractors
for $n \leftarrow |C| - 1$ *to* 0 **do**
  // remove candidates entailing with the answer
  $P_d \leftarrow replace(C_n, A, P)$;
  $check \leftarrow (M_{NLI}(P_d, P), M_{NLI}(P_d, P))$; // two-way entailment check
  if $check == (entailment, entailment)$ **then**
    $remove\_item(C, index = n)$
  end
end
$increment \leftarrow True$;
$i_{kept} \leftarrow 1$;
while $i_{kept} < k < |C|$ **do**
  // remove candidates entailing within (among selected distractors)
  for $j \leftarrow 1$ *to* i_{kept} **do**
    $P_{d1} \leftarrow replace(C_j, A, P)$;
    $P_{d2} \leftarrow replace(C_{i_{kept}}, A, P)$;
    $check \leftarrow (M_{NLI}(P_{d1}, P_{d2}), M_{NLI}(P_{d2}, P_{d1}))$; // two-way entailment check
    if $check == (entailment, entailment)$ **then**
      $increment \leftarrow False$;
      $remove\_item(C, index = i_{kept})$;
      break
    end
    $increment \leftarrow True$;
  end
  if $increment$ **then**
    $i_{kept} + +$;
  end
end
return $get\_top\_k(C, k)$;

Dataset	CLOTH (High/Test)
# passages	478
Avg. # question per passage	17.41
Avg. # sentence	18.92
Avg. # words	365.1

Table 6: Statistics of the CLOTH dataset test split (high).

answers and distractors consist of a single word, we set $dispersion = 0$, $n_{mask} = 1$, $k = 10$ and $m_s = 7$. Note that forcing $n_{mask} = 1$ does not truly demonstrate the capabilities of our framework as our framework is capable of generating multi-word distractors. We set L2R decoding and used the geometric mean for averaging. Hardware specifications are detailed in Appendix A.

C.4 Results

In this section, we lay out the results either quantitative or qualitative that are obtained from the experiments with aforementioned datasets.

The performance of the framework and comparison with CDGP on the CLOTH dataset is reported in Table 7. We followed the same experiment setup as CDGP. It can be seen that despite not using a fine-tuned PLM on CSG, DisGeM's results are compatible with CDGP's. Moreover, the trade-off for performance vs. fine-tuning may be worth using DisGeM (e.g. for the RoBERTa case).

We also conduct a study with different input settings whose results are reported in Table 8. We tried supplying the whole passage to the model and also supplying only the sentence with the question at hand. Probably due to the pre-training nature of BERT (Devlin et al., 2019), the generations with passages are closer to the ground truth distractors. Furthermore, we also tried pre-filling the blanks other than the question at hand with a PLM compared to pre-filling with the ground truth distractors. Interestingly enough, when the blanks are pre-filled with PLMs (we used RoBERTa-large (Liu et al., 2019)) the generations are closer to the ground truth. We did not conduct a pre-filling comparison for sentence inputs as the majority of the sentences have a single blank, and used ground truth answers only for those that have more than 1 blank.

Models	P@1	F1@3	MRR@10	NDCG@10
CDGP (BERT)	**18.50**	**13.80**	**29.96**	**37.82**
CDGP (RoBERTa)	10.50	9.83	20.42	28.17
DisGeM (BERT)	14.00	7.67	19.03	22.94
DisGeM (RoBERTa)	8.00	6.00	14.80	19.92

Table 7: Automated metrics from the evaluation on CLOTH dataset test split (high) and comparison with CDGP (Chiang et al., 2022).

Models	P@1	F1@3	MRR@10	NDCG@10
BERT (P/FM)	**14.00**	**7.67**	**19.03**	**22.94**
BERT (P/G)	12.00	6.67	16.90	20.89
BERT (S/G)	11.00	5.33	15.01	18.60

Table 8: Automated metrics from the evaluation on CLOTH dataset test split (high) utilizing different techniques on DisGeM. Model input is **(P)** the whole passage **(S)** the sentence containing the blank/question. If the input context has more blanks other than the question at hand **(FM)** they are pre-filled using a PLM, **(G)** they are pre-filled with gold answers.

C.5 Discussion

In the metric evaluations, the reported automated metrics either cannot reach SOTA or are comparable to some end for some variants (e.g. CDGP RoBERTa results in Table 7). These results are not surprising for several reasons. First of all, our framework is training-free and we do not fine-tune PLMs for specific datasets (e.g. CLOTH) whereas in CDGP's framework, the results are obtained by PLMs that are fine-tuned on the datasets. Hence, it's expected that CDGP's framework is better at matching the distractors of that particular domain. We do not want to state that CDGP's or DisGeM's better in comparison by only interpreting the automatic metrics on CLOTH datasets since this kind of evaluation measures that the system is capable of matching the gold distractors written by people for that dataset. As Chiang et al. (2022) also states, the current evaluation on the test dataset may not truly represent the quality of the distractor generation system as a mismatch with the ground truth distractors does not necessarily indicate the infeasibility of the generated distractor.

It's also worth mentioning that the CLOTH dataset consists of single-word answers and distractors, which somewhat restricts our system from demonstrating its capabilities to the fullest extent.

D Qualitative Analysis

In this section, we present qualitative samples from our framework's outputs, accompanied by commentary.

For the qualitative analysis, we have selected examples from the SQuAD dataset. It is important to note that the SQuAD dataset is primarily designed as an extractive question-answering dataset, where the answers are directly extracted as spans from the context. While primarily for question answering, it can be used for tasks other, such as question generation. Additionally, we'd like to emphasize that our framework, *DisGeM*, focuses solely on the provided context and not any specific question.

Our distractor generation results, obtained using parameters $dispersion = 1$, L2R decoding, and $n_{mask} = 0$ (i.e. equals the length of the answer tokens), are showcased in Table 9, Table 10 and Table 11 with comparison to CDGP results. To give a realistic representation of MCQs, we randomly positioned the correct answer among the choices without any particular order. Note that CDGP is primarily designed for the use case where the PLMs are first fine-tuned, and these fine-tuned models are then utilized to generate candidates. However, it is important to note that such fine-tuning is specific to multiple-choice question-answering datasets because the training objective relies on ground truth distractors. Contrastingly, the SQuAD dataset consists solely of contexts, questions and answers. To ensure a balanced comparison, we included results from two CDGP outputs: the BERT model fine-tuned on CLOTH (referred to as CDGP (F)) and the pre-trained BERT base (referred to as CDGP (P)), which we also use for DisGeM.

Our qualitative analyses affirm that the candidates generated using the entire passage (as with DisGeM) are significantly more effective than those produced when only the sentence is provided (as in the case of CDGP). In our analyses, we refer to this as “context awareness". Specifically, we use “shallower context awareness" (SCA) when the model input is limited to a sentence and “wider context awareness" (WCA) when it encompasses an entire passage.

D.1 Passage 1 - Super Bowl 50

On Table 9 we selected the passage from the “Super_Bowl_50" article[5]. The passage is about Super Bowl 50 in particular, mentioning related information such as the competing teams, and when and where the game is played.

The generated distractors on Question-1.1, DisGeM successfully generated distractors that are close to the gold answer, “2015". Given that the context mentions Super Bowl 50 was played in 2016, “2016" emerges as a plausible distractor. Options 2014 and 2017 are also closer years to 2015. While the outputs from CDGP (F) closely resemble those of DisGeM, option (A) deviates slightly from other choices. Conversely, the quality of outputs from CDGP (P) is noticeably lower compared to both DisGeM and CDGP (F). This case stands as an effective illustration of the beneficial impact of WCA over SCA.

Upon examining the distractors for Question 1.2, it becomes evident that the distractors generated by CDGP (F) are less effective than the rest. It generated “denver" as a distractor, which is functionally identical to the correct answer, thus invalidating the question. Both DisGeM and CDGP (P) are comparable, though DisGeM’s distractors appear slightly more plausible, given its multi-word generation capability. Other than that both DisGeM and CDGP (P) successfully generated distractors of AFC teams.

For Question-1.3, DisGeM’s outputs outshine those from both CDGP (F) and CDGP (P). This time, unlike Question-1.2, one of the DisGeM’s distractors, “Miami Dolphins", is an AFC team, but not NFC. Both CDGP (F) and CDGP (P) generated undesirable distractors. For CDGP (F), there is only a single acceptable string as a candidate, but once again it is extremely similar to the correct answer. On the other hand, CDGP (P) generated “Denver" and “Broncos" as distractors which refer to the same team “Denver Broncos" making the distractors suboptimal.

D.2 Passage 2 - Nikola Tesla

On Table 10 we selected the passage from the “Nikola_Tesla" article[6]. The passage, Passage 2, is a paragraph about the work life of Nikola Tesla and the events that took place posthumously.

Analyzing the outcomes for Question-2.1, DisGeM’s distractors for the answer “mad scientist" aptly capture Tesla’s reputation in popular culture. Conversely, CDGP (F) and CDGP (P) offered less fitting alternatives, with “mad scientist" being the most appropriate choice among their options though DisGeM and CDGP (P) have a common plausible distractor, “inventor". Nonetheless, DisGeM’s “genius" and “electric genius" distractors are close to each other in meaning which may be undesirable. This underlines DisGeM’s ability to generate context-aware distractors.

Moving to Question 2.2, the comparison highlights both CDGP (F) and CDGP (P) as producing less plausible distractors. Remarkably, CDGP (F) generated “hotels" as a distractor, which directly agrees with the answer “New York hotels". Besides, CDGP (P) distractors are all very similar in meaning, significantly reducing the quality of the overall question and choices. In contrast, DisGeM yielded options that are more contextually feasible, especially option (A), “small retirement homes", which is quite relatable with the context of retirement.

In the case of Question 2.3, the results show that DisGeM’s distractors capture the nuances of the given answer, contrasting with the more generalized options from CDGP (F) and CDGP (P). This allows the distractors to compete more effectively with the answer “SI unit of magnetic flux density". Both CDGP (F) and CDGP (P) distractors substantially differ from the correct answer. Placing a lengthy, detailed answer next to short and general distractors significantly compromises the integrity of the question. Conversely, DisGeM showcases its proficiency by creating distractors that resonate with the answer well, although Option (D), “first “universal" electric car", introduces an element of whimsy that feels slightly out of place.

[5]https://rajpurkar.github.io/SQuAD-explorer/explore/1.1/dev/Super_Bowl_50.html

[6]https://rajpurkar.github.io/SQuAD-explorer/explore/1.1/dev/Nikola_Tesla.html

Passage-1	Super Bowl 50 was an American football game to determine the champion of the National Football League (NFL) for the **__1__** season. The American Football Conference (AFC) champion **__2__** defeated the National Football Conference (NFC) champion **__3__** 24–10 to earn their third Super Bowl title. The game was played on February 7, 2016, at Levi's Stadium in the San Francisco Bay Area at Santa Clara, California. As this was the 50th Super Bowl, the league emphasized the "golden anniversary" with various gold-themed initiatives, as well as temporarily suspending the tradition of naming each Super Bowl game with Roman numerals (under which the game would have been known as "Super Bowl L"), so that the logo could prominently feature the Arabic numerals 50.		
Question-1.1	Super Bowl 50 decided the NFL champion for what season?		
Choices	DisGeM	CDGP (F)	CDGP (P)
	A. 2016 B. 2014 C. 2017 **D. 2015**	A. 2010 B. 2014 C. 2017 **D. 2015**	A. 1950 B. 1949 C. 1951 **D. 2015**
Question-1.2	Which NFL team represented the AFC at Super Bowl 50?		
Choices	DisGeM	CDGP (F)	CDGP (P)
	A. Patriots **B. Denver Broncos** C. Miami Dolphins D. Houston Texans	A. denver **B. Denver Broncos** C. eventually D. .	A. Patriots **B. Denver Broncos** C. Steelers D. Colts
Question-1.3	Which NFL team represented the NFC at Super Bowl 50?		
Choices	DisGeM	CDGP (F)	CDGP (P)
	A. Carolina Panthers B. 49ers C. Philedelphia Eagles D. Miami Dolphins	**A. Carolina Panthers** B. carolina C. , D. .	**A. Carolina Panthers** B. Denver C. Broncos D. Colts

Table 9: Several SQuAD examples with context, question, answer and generated distractors. For CDGP outputs are with **(F)** BERT model fine-tuned on the CLOTH dataset **(P)** pre-trained BERT model.

Passage-2	Tesla was renowned for his achievements and showmanship, eventually earning him a reputation in popular culture as an archetypal **__1__** . His patents earned him a considerable amount of money, much of which was used to finance his own projects with varying degrees of success. He lived most of his life in a series of **__2__** , through his retirement. Tesla died on 7 January 1943. His work fell into relative obscurity after his death, but in 1960 the General Conference on Weights and Measures named the **__3__** the tesla in his honor. There has been a resurgence in popular interest in Tesla since the 1990s.		
Question-2.1	What was Tesla's reputation in popular culture?		
Choices	DisGeM	CDGP (F)	CDGP (P)
	A. inventor B. genius **C. mad scientist** D. electric genius	A. actor B. artist **C. mad scientist** D. writer	A. figure B. hero **C. mad scientist** D. inventor
Question-2.2	Where did Tesla live for much of his life?		
Choices	DisGeM	CDGP (F)	CDGP (P)
	A. small retirement homes B. homes worldwide **C. New York hotels** D. hospitals	A. hospitals B. parks **C. New York hotels** D. hotels	A. houses B. homes **C. New York hotels** D. apartments
Question-2.3	What was named "The Tesla" in his honor?		
Choices	DisGeM	CDGP (F)	CDGP (P)
	A. new precision measurement device B. new "universal motor" **C. SI unit of magnetic flux density** D. first "universal" electric car	A. model B. unit **C. SI unit of magnetic flux density** D. field	A. device B. instrument **C. SI unit of magnetic flux density** D. asteroid

Table 10: Several SQuAD examples with context, question, answer and generated distractors.For CDGP outputs are with **(F)** BERT model fine-tuned on the CLOTH dataset **(P)** pre-trained BERT model.

D.3 Passage 3 - Nikola Tesla

On Table 11 we selected the passage from the “Nikola_Tesla" article[7] like Passage 2. This passage touches on the early life and background of Nikola Tesla. It mentions his birthdate, family background, and some of the influences and factors that contributed to his development and abilities.

When analyzing the results for Question-3.1, DisGeM stands out for its ability to craft distractors closely linked to the answer “Croatia", capturing the essence of the modern-day country of Tesla’s birth. However, CDGP (F) seems to focus narrowly on the geographical proximity, offering choices like “france" and “hungary". While they are geographically related, they don’t resonate with contextual clues like “Austrian Empire", “Smiljan", “Serbian", and “Serb family". Meanwhile, CDGP (P) offers distractors that closely align with the context, but critically includes “Croatia", which is the actual answer, rendering the question invalid.

Turning to Question-3.2, DisGeM’s distractors align well not only with the context and the flow but also with the answer “his mother’s genetics". In contrast, the distractors generated by both CDGP (F) and CDGP (P) are not as contextually fitting.

D.4 Conclusion

In summary, our qualitative analyses, supported by numerous examples, underline DisGeM’s proficiency in generating contextually relevant and well-aligned distractors. DisGeM notably outperforms both CDGP (P) and CDGP (F), enhancing the quality of distractor generation, especially in scenarios where single-word distractor generation is inadequate. We highlighted several limitations of the CDGP framework, the previous SOTA, such as its strong reliance on fine-tuning, its tendency to produce distractors semantically close to the answer, and its inability to generate multi-word distractors. These insights reinforce the benefits of adopting the DisGeM framework, especially when high precision and relevance in distractor generation are paramount. Overall, DisGeM’s predominant advantage stems from its capacity for multi-word generation.

[7] https://rajpurkar.github.io/SQuAD-explorer/explore/1.1/dev/Nikola_Tesla.html

Part	SQuAD Samples		
Passage-3	Tesla was born on 10 July [O.S. 28 June] 1856 into a Serb family in the village of Smiljan, Austrian Empire (modern-day **__1__**). His father, Milutin Tesla, was a Serbian Orthodox priest. Tesla's mother, Đuka Tesla (née Mandić), whose father was also an Orthodox priest,:10 had a talent for making home craft tools, mechanical appliances, and the ability to memorize Serbian epic poems. Đuka had never received a formal education. Nikola credited his eidetic memory and creative abilities to **__2__** and influence. Tesla's progenitors were from western Serbia, near Montenegro.:12		
Question-3.1	What modern-day country was Tesla born in?		
Choices	DisGeM	CDGP (F)	CDGP (P)
	A. Serbia B. Montenegro **C. Croatia** D. Bosnia Herzegovina	A. france B. china **C. Croatia** D. hungary	A. Croatia B. Serbia **C. Croatia** D. Slovenia
Question-3.2	Who did Tesla credit for his abilities?		
Choices	DisGeM	CDGP (F)	CDGP (P)
	A. the family's wealth B. is own personal experience **C. his mother's genetics** D. his mother's education	A. study B. change **C. his mother's genetics** D. adapt	A. him B. inspiration **C. his mother's genetics** D. success

Table 11: Several SQuAD examples with context, question, answer and generated distractors. For CDGP outputs are with **(F)** BERT model fine-tuned on the CLOTH dataset **(P)** pre-trained BERT model.