Generation and Evaluation of Multiple-choice Reading Comprehension Questions for Swedish

Dmytro Kalpakchi, KTH Royal Institute of Technology, Stockholm, Sweden dmytroka@kth.se
Johan Boye, KTH Royal Institute of Technology, Stockholm, Sweden jboye@kth.se

Abstract Multiple-choice questions (MCQs) provide a widely used means of assessing reading comprehension. The automatic generation of such MCQs is a challenging language-technological problem that also has interesting educational applications. This article presents several methods for automatically producing reading comprehension questions MCQs from Swedish text. Unlike previous approaches, we construct models to generate the whole MCQ in one go, rather than using a pipeline architecture. Furthermore, we propose a two-stage method for evaluating the quality of the generated MCQs, first evaluating on carefully designed single-sentence texts, and then on texts from the SFI national exams. An extensive evaluation of the MCQ-generating capabilities of 12 different models, using this two-stage scheme, reveals that GPT-based models surpass smaller models that have been fine-tuned using small-scale datasets on this specific problem.

1 Introduction

In several educational stages, multiple-choice questions (MCQs) provide a widely used means of assessing reading comprehension (OECD, 2021). Tests that consist of MCQs have the very appealing property of allowing swift, automatic, and objective grading. However, creating such tests is far from being swift or automatic, but rather is time-consuming and requires a great deal of expertise (Haladyna, 2004). In this work, we analyze to what extent the creation of MCQ tests for reading comprehension in Swedish could be automated using publicly available language models (both closed-source models via public APIs, and open-source ones).

The focus of this article is on MCQs aimed at assessing reading comprehension of second-language learners of Swedish, specifically aimed at the Swedish for Immigrants courses (SFI). Our contributions[1] are:

- we propose a number of methods that can generate a number of distinct reading comprehension MCQs from a given text;
- we propose a two-stage method for evaluating the quality of the generated MCQs, first evaluating on carefully designed single-sentence texts, and then on a small corpus of texts[2] from the SFI national exams (for the D-level course);
- we compare our proposed methods with the state-of-the-art GPT-3 and ChatGPT models, as well as some baselines.

An MCQ consists of a question proper (the *stem*), the correct answer (the *key*), and a number of wrong but plausible options (the *distractors*). We will refer to the key and distractors together as *alternatives* (ALT). Contrary to prior work (Majumder and Saha, 2015; Araki et al., 2016; Guo et al., 2016; Zhou et al., 2020; Kalpakchi and Boye, 2021), we do NOT split the problem of generating stem and key from the problem of generating distractors. Instead, we aim to generate the whole "package" at once and be able to offer more than one MCQ per text. We assume that the texts are already given, and that they are on the appropriate level for testing reading comprehension, e.g., they are of the appropriate length, split into paragraphs, use the vocabulary of the appropriate complexity for the second-language learners, etc. In this article we do NOT conduct any assessment on how appropriate the given texts are for testing reading comprehension. The interested reader is referred to (OECD, 2019, 2021) for more discussion on this matter.

In the aforementioned prior work, researchers have tried a two-stage approach to MCQ generation, first generating a stem-key pair using one method and then generating the distractors using another method. Such approaches have a number of advantages, e.g., the key can be extracted directly from the text and thus guar-

[1] The source code and the Plugga corpus will be freely available upon the publicaiton of the article.
[2] We call this corpus *Plugga* and make it available online

anteed to be correct, or the stem/key formulation can be edited to (hopefully) get higher quality distractors. Nevertheless, generating the whole "package" at once, as we attempt to do in this article, has also its advantages. Our motivation is twofold. The first reason is that a stem-key pair produced at stage 1 might not necessarily allow for good distractors to be produced at stage 2. The hope is that when generating the whole MCQ in one go, the model will learn to only generate stems that have reasonable alternatives. The second reason is speed of execution, meaning that it is more resource-efficient to run one model and directly obtain the entire MCQ, instead of running several models that generate each part of the MCQ separately.

2 Related work

Automatic question generation from text has been studied before, mainly for the English language. Results obtained up to 2020 are summarized in the survey article by Zhang et al. (2021). However, very little work has been done on generating whole MCQs (rather than just the questions), although some researchers have focused on generating distractors separately using large language models (Qiu et al., 2020; Zhou et al., 2020; Offerijns et al., 2020; Chung et al., 2020; Kalpakchi and Boye, 2021).

To the best of our knowledge, there have been no prior attempts on generating *the whole reading comprehension MCQs* for Swedish using fully open-source and free-to-use models. The only prior attempt in this direction was by Kalpakchi and Boye (2023a), where they generated MCQs using OpenAI's GPT-3 (Brown et al., 2020), which is neither open-source nor free to use.

However, there have been attempts at generating parts of MCQs for Swedish. Kalpakchi and Boye (2022a) generated stems and keys separately using a data-driven rule inductor based on Universal Dependencies for creating templates for question-answer pairs and then attempting to apply them to single sentences during the generation process.

Kalpakchi and Boye (2021) experimented with a method based on the Swedish BERT (Malmsten et al., 2020) for generating distractors given the text, the stem, and the key.

3 Data

3.1 Training

In this work we experimented with two previously released datasets of Swedish MCQs for reading comprehension. The first one, SweQUAD-MC (Kalpakchi and Boye, 2021), contains MCQs on texts scraped from the websites of Swedish public authorities. Following the definition of OECD (2019), all texts have a *continuous* format, which means they have no internal structure beyond being organized into sentences and paragraphs. The MCQs were created by paid linguistics students and required both the key and distractors either to appear in the text directly, or be a grammatical reformulation of a phrase present in the text. Most MCQs in SweQUAD-MC contain three alternatives, but some include more (up to six alternatives). Nevertheless, it is guaranteed that exactly one of the presented alternatives is correct.

The second dataset, Quasi (Kalpakchi and Boye, 2023a), consists of 90 texts collected from the SFI[3] national examinations, along with 317 MCQs synthetically generated by GPT-3 and manually curated. The texts in Quasi are of different genres, e.g., news articles, ads, e-mails, blog posts, etc. The majority of the texts are either partially or fully *non-continuous* which, by the definition of OECD (2019), means that they have some internal structure that helps (or is necessary for) understanding their content. For instance, e-mails contain the addresses of the sender and receiver in certain places, recruitment ads feature the employer company name and contact details, and posts contain the name, and possibly also the age, of the author. All MCQs in Quasi contain four alternatives, of which exactly one is correct.

3.2 Evaluation

Evaluation sets for reading comprehension questions in Swedish are scarce. SweQUAD-MC does have a validation and test set, but the texts are not verified to be suitable for testing reading comprehension (e.g., some of them might use too complex language, especially for 2nd language learners). The texts in Quasi are taken from national SFI examinations and are thus suitable for assessing reading comprehension. However, Quasi provides only a small set of MCQs, which is impractical to split further into training and test sets.

It is also worth noting that there are currently no reliable methods to automatically evaluate the quality of MCQs. Evaluation methods like BLEU, ROUGE, and METEOR, which are based on word overlap between the generated result and the "gold" MCQ in a test set, will not give much information due to the open-ended nature of the task – from any non-trivial text, a very large number of MCQs are possible. Hence, there is no real benefit in having a test set of MCQs. Instead, one should have a test set of texts suitable for reading comprehension, and with a degree of variation in multiple aspects, such as length, genre, formatting, etc.

In this work, we have adopted the latter approach

[3] Swedish for Immigrants, the national Swedish course curriculum for 2nd language learners.

and have collected a corpus of 10 texts for evaluating reading comprehension in Swedish, a corpus which we will refer to as *Plugga*. Similarly to designers of Quasi, we took materials from the previous national exams for SFI (but made sure that there is *no overlap* between Quasi and Plugga) by running the Tesseract OCR engine[4], and manually correcting the outputs. The sources and genres of texts in Plugga are distributed as follows:

- 2 newspaper articles, shortened and simplified by the SFI test constructors;
- 1 shortened and simplified yearly report from the public authority (Statistics Sweden, SCB);
- 1 short text with tips when to ring the emergency telephone number 112 from SOS Alarm (the company running 112);
- 2 compiled short answers to a given question by multiple people;
- 2 e-mails;
- 1 short forum thread discussing a given issue;
- 1 detailed program to an event.

The first three sources are continuous texts, divided into paragraphs, whereas the last four sources are either fully or partially non-continuous.

4 Method

In this work, we have experimented with fine-tuning two publicly available large language models: Swedish BERT (Malmsten et al., 2020), and SweCTRL-Mini (Kalpakchi and Boye, 2023c). We compared the fine-tuned models to two baselines described in Section 4.4, as well as GPT-3 (Brown et al., 2020), specifically *text-davinci-003*, and ChatGPT[5], specifically *gpt-3.5-turbo-0301*. We did not use GPT-4[6] in this work, because at the time of writing, access to its API is limited for the general public. The same goes for GPT-SW3[7].

4.1 Models based on KB/BERT

Swedish BERT, later referred to as *KB/BERT*, is a discriminative model that has been previously used by Kalpakchi and Boye (2021) for generating distractors with relative success. In this work, we also use KB/BERT, but attempt to generate *whole MCQs* and not only distractors. Following Kalpakchi and Boye (2021), we frame the problem as an auto-regressive generation in two different ways: left-to-right and arbitrary-order[8]. The training procedure for both cases is summarized in Table 1. In both cases, each MCQ (here with 2 distractors) is represented as follows (later referred to as an *MCQ sequence*):

```
T [SEP] Q [SEP] A [SEP] D1 [SEP] D2
```

where `Q` denotes all the tokens of the stem (the question proper), `A` the words in the key (the correct answer), and so on, and `[SEP]` is the special separator token in BERT. Each `[SEP]`-separated part of the MCQ sequence except `T` will be referred to as an *MCQ sequence item*. For the sake of brevity, we will only use two distractors `D1` and `D2` in the examples. In general, we will use D to denote the set of distractors.

For the **left-to-right variant** (LRV), both training and generation is designed to proceed from left to right both when producing the whole MCQ sequence, and when generating each MCQ sequence item. **At training time**, each MCQ from the training data is represented as an MCQ sequence, which is then converted into multiple datapoints. This conversion is obtained by building the MCQ sequence one token at a time, masking the last token, and attempting to predict it. An example of such conversion is given in the top subtable of Table 1. In row 1, we started building the MCQ sequence, which consists of a single token. This token gets masked, and the task is to predict the correct token `Q1` from the Target column. Then, we add the next token (row 2) and mask the last token in the partial MCQ sequence, and again attempt to predict that token (`Q2` in this case), and so on. When we finished generating the stem (row 4), we add the `[SEP]` token, which now becomes the last token of the sequence and hence is also masked, requiring the model to be able to also predict `[SEP]` tokens correctly for learning to finish each sequence item. In this manner, each MCQ is converted into $|\texttt{Q}|+|\texttt{A}|+\sum_{\texttt{DX}\in D}|\texttt{DX}|+|D|+2$ training datapoints. **At generation time** the process is similar to the training time, but without knowing the target tokens. Specifically, we input the text `T`, followed by a `[SEP]` token, and append a `[MASK]` token at the end. Then we unmask the `[MASK]` token, by sampling from the provided distribution, and append a new `[MASK]` token. This process is repeated until we have generated $|D|+2$ `[SEP]` tokens, in which situation we assume that the stem, the key, and all distractors have been generated. We have enforced a hard limit of 30 tokens for the stem and 20 tokens for each alternative.

For the **arbitrary-order variant** (AOV-A), both training and generation is designed to proceed in an

[4] Freely available at https://github.com/tesseract-ocr/tesseract

[5] https://openai.com/blog/chatgpt

[6] https://openai.com/gpt-4

[7] https://www.ai.se/en/node/81535/gpt-sw3

[8] Referred to as the "u-PMLM variant" in the original article.

#	Input for left-to-right KB/BERT variant (LRV)	Target
1	[M]	Q1
2	Q1 [M]	Q2
3	Q1 Q2 [M]	Q3
	...	
4	Q1 Q2 ... QK [M]	[S]
5	Q1 ... QK [SEP] [M]	A1
6	Q1 ... QK [S] A1 [M]	A2
7	Q1 ... QK [S] A1 A2 [M]	[S]
8	Q1 ... QK [S] A1 A2 [S] [M]	D11
9	Q1 ... QK [S] A1 A2 [S] D11 [M]	D12
10	Q1 ... QK [S] A1 A2 [S] D11 D12 [M]	[S]
11	Q1 ... QK [S] A1 A2 [S] D11 D12 [S] [M]	D21
12	Q1 ... QK [S] A1 A2 [S] D11 D12 [S] D21 [M]	D22
13	Q1 ... QK [S] A1 A2 [S] D11 D12 [S] D21 D22 [M]	D23
14	Q1 ... QK [S] A1 A2 [S] D11 D12 [S] D21 D22 D23 [M]	[S]

#	Input for arbitrary-order KB/BERT variant (AOV-A)	Target(s)
1	Q1 [M] Q3 ... QK	Q2
2	[M] Q2 [M] ... QK	Q1, Q3
3	[M] [M] Q3 ... [M]	Q1, Q2, QK
	...	
4	Q1 ... QK [S] [M] A2	A1
5	Q1 ... QK [S] [M] [M]	A1, A2
6	Q1 ... QK [S] A1 [M]	A1
7	Q1 ... QK [S] A1 A2 [S] [M] D12	D11
8	Q1 ... QK [S] A1 A2 [S] D11 [M]	D12
9	Q1 ... QK [S] A1 A2 [S] [M] D12	D11
10	Q1 ... QK [S] A1 A2 [S] D11 D12 [S] D21 D22 [M]	D23
11	Q1 ... QK [S] A1 A2 [S] D11 D12 [S] [M] D22 [M]	D21, D23
12	Q1 ... QK [S] A1 A2 [S] D11 D12 [S] D21 [M] D23	D22

#	Input for arbitrary-order-all-at-once KB/BERT variant (AOV-B)	Target(s)
1	Q1 [M] Q3 Q4 [B] [S] [M] A2 [B] [S] D11 D12 [B] [S] D21 D22 D23	Q2, A1
2	Q1 Q2 [M] Q4 [M] [S] A1 [M] [B] [S] D11 [M] [B] [S] D21 D22 D23	Q3, [B], A2, D12
3	[M] Q2 Q3 [M] [B] [S] [M] [M] [B] [S] D11 D12 [B] [S] D21 D22 D23	Q1, Q4, A1, A2
4	Q1 Q2 Q3 Q4 [M] [S] A1 A2 [M] [S] D11 D12 [M] [S] D21 D22 D23	[B], [B], [B]
5	[M] Q2 Q3 Q4 [B] [S] [M] A2 [B] [S] D11 [M] [B] [S] D21 D22 D23	Q1, A1, D12
6	Q1 Q2 Q3 Q4 [B] [S] A1 [M] [B] [S] D11 D12 [B] [S] D21 D22 D23	A2
7	Q1 [M] Q4 [M] [S] A1 A2 [B] [S] D11 D12 [B] [S] D21 D22 D23	Q2, [B]
8	Q1 Q2 Q3 Q4 [B] [S] A1 A2 [B] [S] D11 [M] [B] [S] [M] D22 D23	D12, D21
9	Q1 Q2 Q3 Q4 [B] [S] A1 A2 [M] [S] D11 D12 [B] [S] D21 D22 D23	[B]

Table 1: Example datapoints extracted from *one* MCQ for training the model capable of left-to-right (top table) or arbitrary-order generation (bottom table). **Observe that all inputs are also prefixed with a string** [CLS] T [SEP], **which is omitted in this table for the sake of brevity**. [M] and [S] denote BERT's [MASK] and [SEP] tokens respectively. [B] denotes a special padding token [BLANK] introduced by us. In the example for the AOV-B variant, the question was padded to the maximum of 5 tokens, and each alternative was padded to the maximum of 3 tokens.

arbitrary order only when generating each MCQ sequence item, whereas the whole MCQ sequence is still produced from left to right. **At training time**, in contrast to LRV, we pad the stem and each alternative with a specially introduced token `[BLANK]` to obtain a fixed width of 30 tokens for the stem and 20 tokens for each alternative (the same as the hard limits for LRV). Again we convert each MCQ into multiple datapoints, but in a different way. We start with the leftmost MCQ sequence item, and *corrupt* it K times, as follows: We first draw a masking probability r from the uniform distribution, and attempt to mask each token of the MCQ sequence item with this probability. For instance, in rows 1–3 of the middle sub-table from Table 1, we deal with corrupting first MCQ item, the stem (question proper). Note that the masking in each row corresponds to a *different* value of r (and there are K such values in total). Once we have corrupted one MCQ sequence item K times, we append its non-corrupted version to the partial sequence, and proceed with corrupting the next sequence item, the key, while keeping the preceding MCQ sequence items non-corrupted. We proceed in the same manner until the whole MCQ sequence has been processed. Following (Kalpakchi and Boye, 2021), and contrary to LRV, we don't train the model to unmask the `[SEP]` token. **At generation time**, we provide a fixed number of `[MASK]` tokens (30 for the stem, and 20 for each alternative). Following Kalpakchi and Boye (2021), we unmask the token at the position where the model is most confident. However, rather than selecting the token with the maximum probability, we sample, in order to be able to generate more than one MCQ per text.

Additionally, we introduce another arbitrary-order setup, where we perform exactly the same procedure as for AOV-A, but on all MCQ sequence elements at once (as demonstrated by the bottom sub-table of Table 1). We refer to this setup as AOV-B. **At generation time**, we add $30 + 20 \cdot (|D| + 1)$ `[MASK]` tokens separated by `[SEP]` tokens. Apart from this, the unmasking procedure is the same as for AOV-A.

Because of the elaborate unmasking scheme, the generation phase of AOV-A and AOV-B takes somewhat longer time per token than the LRV setup.

4.2 Models based on SweCTRL-Mini

SweCTRL-Mini is a generative model capable of generating texts one token at a time, left to right. We fine-tune it similarly to left-to-right generation for KB/BERT, except that we don't use `[MASK]` and `[SEP]` tokens, as they are BERT-specific. Instead, we introduce a new *control code pair* consisting of the *opening control code* `:mcq:`, and its corresponding *ending control code* `:mcq:$`. The key is always the first of the four alternatives and is prefixed by `a)`, whereas all distractors are prefixed by the letters after "a", i.e., `b)`, `c)`, etc. Hence the structure of a datapoint for fine-tuning SweCTRL-Mini would look as follows:

```
T :mcq:  Q a) A b) D1 c) D2 :mcq$
```

We then train the model to predict one token at a time, left to right, for all tokens except those in `T`. **At generation time**, the MCQs were sampled we append `:mcq:` after the text `T` and sample the new tokens left to right using the nucleus sampling with the threshold $p = 0.9$ until reaching `:mcq:$`.

4.3 Models based on GPT

We used both GPT-3 and ChatGPT in a zero-shot manner. Inspired by Kalpakchi and Boye (2023a), we input the following prompt to both models, in **Swedish**:

> Skriv N_q^T läsförståelsefrågor med alternativ (a, b, c, d, o.s.v.) och ge varje fråga en unik nummer (1, 2, 3, o.s.v.). Första alternativet (a) ska alltid vara rätt, medan de andra alternativen (b, c, d, o.s.v.) ska vara felaktiga, men troliga. Alla frågor måste kunna besvaras av den följande texten.

To aid the readers not speaking Swedish, we provide the prompt's English *translation* below:

> Write N_q^T reading comprehension questions with alternatives (a, b, c, d, and so on) and give each question a unique number (1, 2, 3, and so on). The first alternative (a) should be always correct, while the other alternatives (b, c, d, and so on) should be wrong, but plausible. All questions must be answerable by the following text.

However, we stress again that the prompt was fed directly *in Swedish*. We used nucleus sampling with the nucleus threshold $p = 0.9$, and limited the maximum number of tokens to 2048 to accommodate the larger texts in Plugga.

4.4 Baselines

4.4.1 Baselines using Universal Dependencies

For this baseline, we generate stems and keys first using Quinductor (Kalpakchi and Boye, 2023b, 2022a) and then use the extractive distractor generation baseline proposed by Kalpakchi and Boye (2021). For both of these, we have used the provided official implementations, available on GitHub. All of these methods rely on Universal Dependencies (Nivre et al., 2020) and require the following resources:

- a pre-trained dependency parser compliant with Universal Dependencies;

- a dataset of texts and question-answer pairs (QA-pairs) for automatically inducing the Quinductor templates;
- a corpus of texts to extract the distractors from (referred to as *DIS-corpus*).

As a minimal example of a Quinductor template, consider the sentence "Tim plays basketball", with an associated question "What does Tim play?". The parser would conclude that the sentence "Tim plays basketball' consists of a verb (the root node `r`), the subject (`r.nsubj`), and an object (`r.obj`). The question "What does Tim play?" can then be described as "What does `[r.nsubj] [r.lemma]`?". This template can then be used to generate a question from a new, previously unseen sentence with a parallel grammatical structure (e.g. "Sue likes spaghetti" yielding "What does Sue like?"). Note that such templates are induced (and can be used) only on single sentences in the text.

When all the resources are in place and the Quinductor templates have been induced, the generation of an MCQ with K alternatives based on the previously unseen text `T'` should proceed as follows:

1. Using the Quinductor method, generate and rank the QA-pairs using previously induced templates.
2. Select the desired number $N_q^{T'}$ of highest-ranked QA-pairs
3. For each QA-pair, extract distractors from the DIS-corpus by searching for the first $K-1$ syntactic structures similar to that of the key.

4.4.2 Zero-shot SweCTRL-Mini

Recall that SweCTRL-Mini is a generative model, in contrast to KB/BERT. Hence, it is possible that it could be able to produce MCQs (fully or partially) in a zero-shot manner. In this work we experiment with the following setup (later referred to as simply *Zero-shot*):

- Input the text `T` (for longer texts we follow the procedure outlined in Section 5).
- First generate a stem by using the prompt "`T Fråga:`" and keep generating until a "?" symbol is produced (separately or as part of another token).
- Then, use the generated stem and attempt to generate the key with the prompt "`T Fråga: Q Svar:`". Proceed until generating a full stop (.).
- Finally, use the generated stem and key and attempt to generate three distractors with the prompt "`T Fråga: Q a) A b)`". Terminate generation when reaching either the string "`e)`" or the string "Fråga".

At all stages of the generation process, we imposed a hard limit of 30 tokens.

5 Experimental setup

In this work we fine-tuned models for the 4 proposed methods (KB/BERT LRV, KB/BERT AOV-A, KB/BERT AOV-B, and SweCTRL-Mini). Each of these models was trained on two datasets (SweQUAD-MC, and Quasi), resulting in $4 \cdot 2 = 8$ models.

Each fine-tuned model was trained for 10 epochs using the AdamW optimizer (Loshchilov and Hutter, 2019) with the default Huggingface training parameters: initial learning rate 5×10^{-5}, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1 \times 10^{-8}$, without learning rate scheduling or weight decay. The gradients were clipped to the norm of 1. The training was conducted on a single NVIDIA 3090 GPU with 24GB of VRAM using a batch size of 8 for models based on SweCTRL-Mini, and a batch size of 16 for those based on KB/BERT. The checkpoints for each model were saved for each epoch, resulting in 10 checkpoints per fine-tuned model.

The major challenge with both KB/BERT and SweCTRL-Mini is their limited and rather small context window size (512 tokens for KB/BERT, and 256 tokens for SweCTRL-Mini). At all times the context window should accommodate both the text and the MCQ. To ensure that, we limited the number of text-related tokens to at most L_T tokens. The exact value of this limit was model-specific, namely $L_T = 441$ for KB/BERT LRV, $L_T = 384$ for KB/BERT AOV-A and AOV-B, and $L_T = 192$ for SweCTRL-Mini. However, if the MCQs in the training data could not be fit in the remainder of the context window, we automatically decreased these limits[9]. At all times we ensure that the basis for the correct answer from the text (the information that is provided by both datasets) is included in the context window.

For the UD-based baseline, we relied on the training set of SweQUAD-MC for generating both QA-pairs[10], and distractors[11]. For this work, we opted out of inducing such templates on Quasi, because most of the texts in Quasi are (partially) non-continuous texts. Since Quinductor was designed to work on single sentences from continuous texts, it is therefore unlikely that templates induced on Quasi will end up being generalizable (or will be induced at all).

The zero-shot SweCTRL-Mini baseline did not require any specific further training (by the nature of being zero-shot). This brings the total number of models to ten.

[9] For more information on this heuristic we refer to the source code accompanying the article.

[10] Using Quinductor templates provided by Kalpakchi and Boye (2022a) in the associated GitHub repository

[11] Using only raw texts from the training set of SweQUAD-MC

When evaluating, similarly to Kalpakchi and Boye (2023a), the longer the text T, the more MCQs we attempted to generate. More specifically, we requested N_q^T MCQs for each T using the following formula:

$$N_q^T = \left\lceil \frac{C_T}{\bar{W} \cdot \bar{C}} \right\rceil, \quad (1)$$

where C_T is the number of characters in T, $\bar{W} = 12.78$ ($\bar{C} = 4.81$) is the average number of words (characters) per sentence. These quantities were calculated as a weighted average across corpora from (Kalpakchi and Boye, 2023a, Table 1) with the weights being the relative sizes of the corpora in words.

6 Model selection

The goal for this section is to select the best checkpoint for each of the eight fine-tuned models. Since there are ten checkpoints per model, this step requires evaluating the quality of 80 checkpoints, which is prohibitively expensive to do using human evaluation. Instead we resort to using metrics that could be calculated automatically. Furthermore, since there is no reliable way to estimate how good the MCQs are using the automatic metrics, we aim at estimating how many *definitely* bad MCQs were generated by each checkpoint.

To define *definitely* bad MCQs we employed the following *badness metrics* (listed from the most to the least severe). In the list below, "MCQ%" means "percentage of MCQs", and "ALTs" means "alternatives" (the key and distractors together), whereas "ALT" means "an alternative" (either the key or any distractor). For all of the metrics below, *the lower, the better*.

1. *AltInStem*. MCQ% with any ALT being verbatim in the stem.
2. *AltAllSame*. MCQ% with all identical ALTs.
3. *StemTextRep*. MCQ% with the stem containing repetitive phrases contiguously (up to 10 tokens).
4. *AltAnyTextRep*. MCQ% with any ALT containing repetitive phrases contiguously (up to 10 tokens).
5. *AltAnySame*. MCQ% with ≥ 2 (but not all) identical ALTs.
6. *AltAnyEmpty*. MCQ% with ≥ 1 ALTs being an empty string[12].
7. *StemEmpty*. MCQ% with the stem being an empty string[12].
8. *StemNoQmark*. MCQ% with the stem not ending with a question mark.
9. *NoEndCode*. MCQ% where the generation was not finished with the appropriate control code `:mcq$:` (only for models based on SweCTRL-Mini).

We evaluated all checkpoints on the texts from the development set of SweQUAD-MC. The texts that are larger than L_T tokens are split into chunks of max L_T tokens. For each model, we have generated N_q^T MCQs, as calculated by Equation 1. Because generating MCQs using models based on KB/BERT is computationally heavy (and we need to generate MCQs for 80 checkpoints), we calculated the badness metrics only for the MCQs generated on the first 100 text chunks from the development set (when all the texts are sorted alphabetically).

Based on the badness metrics reported in Figure 1, we selected the checkpoint with the fewest and least severe errors for each model (recall that the introduced badness metrics are listed from the most to the least severe). Additionally, everything else being the same, we preferred earlier checkpoints to reduce the risk of overfitting (given that the training sets were quite small, especially for the SweCTRL-based models). The selected checkpoints per model (denoted by the number of training epochs) are reported in Table 2.

7 Human evaluation

For human evaluation, we compare eight best checkpoints selected in Section 6, two baseline models, and two GPT-based models, namely GPT-3 (*text-davinci-003*), and ChatGPT (*gpt-3.5-turbo-0301*).

In this section the evaluation begins with the following basic question for the set of produced MCQs per model per text:

Q0. Did the model produce the requested number N_q^T of MCQs?

Then each generated MCQ is evaluated separately on the following aspects:

Q1. Are the question (stem) and all alternatives grammatically correct?

Q2. Is the stem answerable by the text?

Q3. Are all alternatives relevant for the given stem?

Q4. Is the alternative (a) the only correct answer?

If the answer to Q1 is *No*, we report further whether stem, alternatives, or both are ungrammatical. Additionally, we add the category **gibberish** denoting cases when both stem and alternatives are ungrammatical and not formatted properly, or when at least one of

[12] After excluding the special tokens, e.g., `[SEP]`

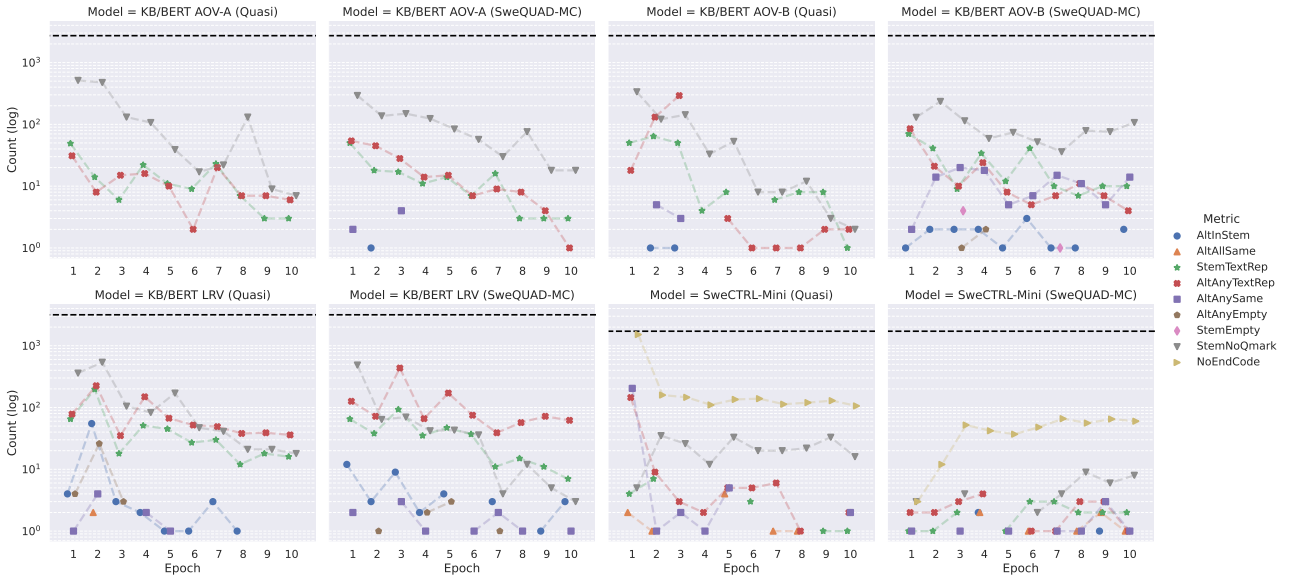

Figure 1: The automatically computed badness metrics for the saved checkpoints of all fine-tuned models. The plots are in *the logarithmic scale* on the y-axis.

	Trained on SweQUAD-MC				Trained on Quasi			
	KB/BERT			SweCTRL	KB/BERT			SweCTRL
	LRV	AOV-A	AOV-B		LRV	AOV-A	AOV-B	
Training epochs	8	10	9	5	8	10	6	9

Table 2: The selected checkpoints (denoted by the number of training epochs) based on the automatically calculated badness metrics.

them is not written in valid Swedish (e.g., there are some loose tokens or words that cannot be connected).

If the answer to Q2 is *No*, we investigate the reasons behind the stems being unanswerable. Following Kalpakchi and Boye (2023a), we categorize such stems into **contradictive** (including presuppositions disagreeing with the text), **undiscussed** (inquiring about information not present in the text), or **ambiguous** (not providing enough information to choose one definite alternative).

If the answer to Q3 is *No*, we also investigate the reasons behind it. Following Kalpakchi and Boye (2023a), we categorize the problematic alternatives into **misfocused** or **heterogeneous**. The former category means that at least one of the alternatives does not provide the type of information requested in the stem. For instance, the stem "When was Alfred Nobel born?" with one of the alternatives being "Stockholm" is enough to classify such MCQ as having misfocused alternatives. The latter category means that one or more of the provided alternatives "stick out" and could provide a meta-clue to the students. For instance, the stem "When was Alfred Nobel born?" with the alternatives "21 October 1833", "1848", "1792", "1835" would be classified to have heterogeneous alternatives, since the first alternative (which happens to be the key) is more detailed than the others and thus "sticks out". Additionally, we introduce two new categories: **empty alternative(s)** (when at least one of the generated alternatives is an empty string), or **duplicate alternatives** (when there are at least two identical alternatives, lowering the effective number of alternatives).

If the answer to Q4 is *No*, we look into three sub-questions to understand why. If any sub-question gets a negative answer, we do not investigate the latter ones. The first sub-question is whether any alternative is the key (the correct answer), to begin with. The second sub-question is whether there is more than one alternative that could be considered to be the key, the case which is referred to as **overlapping alternatives**. The final sub-question is whether the only present key is actually the alternative (a).

Answering Q1 - Q4 required manual annotations, which we did ourselves using an iterative annotation process (annotating – discussing issues – reannotating). We used an instance of Textinator (Kalpakchi and Boye, 2022b) as the annotation tool. The annotation process was blind, meaning that the generated MCQs and their

texts were presented in a random order without any indication as to which model they were sampled from (all model-specific tokens and all question numbers were removed). After the evaluation was done, the separately generated key file (previously unseen by the annotator) was used to match the text annotations with their corresponding models.

7.1 Single-sentence texts

Before conducting evaluation on texts from Plugga, taken from the real-world reading comprehension examinations, we turn to a toy domain of extremely small texts consisting of only one sentence. We refer to texts from this toy domain as single-sentence texts (SSTs).

The rationale behind testing the models on SSTs is to facilitate a quick check whether the generated MCQs inquire *only* about the information given in the text. This concern arises from the well-known fact that large language models can generate pieces of text that sound plausible, but are either irrelevant to the given situation or simply false. In our early tests, we noticed that models tend to make up MCQs that are in line with the general topic of the text (e.g. about Sweden) but do not rely on the facts presented in the text. Such artifacts are absolutely unacceptable when producing reading comprehension tasks since the information necessary for answering a stem **must** be present in the text. While the aforementioned checks can be done on any corpus of texts (as we will do in Section 7.2), the idea with SSTs is to make such checks quick and simple. Another advantage with SSTs is that the evaluation becomes more controlled, as we can observe how well models react to slight changes of the text formulations, e.g. whether they are able to pick up slight changes or added information.

For the evaluation in this section we have created the SSTs presented in Table 3, which include 20 *core SSTs* and three *extra SSTs* (marked with asterisks). The extra SSTs contain a specific kind of grammatical errors, namely anglicisms. This is to check whether GPT-based models trained predominantly on English will borrow grammatical constructs from English, even when evaluated on Swedish texts. For every SST, we requested each model to generate **five MCQs** ($N_q^T = 5$) with the first alternative, (a), being correct.

7.1.1 Overview of the results

Our main finding was that two models produced substantially more MCQs without any problems at all, namely ChatGPT (50.43% problem-free MCQs), and GPT-3 (48.7% problem-free MCQs).

An overview of the found problems is presented in Figure 2. The problems in the legend are sorted by the level of their severity, i.e. the harder it is to fix the MCQ, the more severe the problem is. The histogram in Figure 2 accounts only for *the most severe problem* for each MCQ, meaning that the MCQ is guaranteed to not have problems higher in the list, but could still exhibit the problems lower in the list.

To address Q0, the number of results produced by the model, almost all models produced the requested $N_q^T = 5$ per SST. The only exception is the UD baseline that produced substantially fewer MCQs (producing none for the majority of SSTs).

Related to Q0, the models produced different number of alternatives, as reported in Figure 3. Note that vast majority of MCQs contain four alternatives, including both ChatGPT, and GPT-3 for which the number of requested alternatives was unspecified. The only two models with the substantial number of MCQs with other than four alternatives are SweCTRL-Mini trained on SweQUAD-MC (because the training data mostly contained three alternatives), and Zero-shot SweCTRL-Mini.

To address Q1, the issue of grammatical correctness, we look at the three most severe problems from Figure 2. The first and the most severe problem in the list is *gibberish* (red in Figure 2). Gibberish MCQs do not even provide a starting point for fixing an MCQ and require creating a new one altogether, which is why it is the most severe problem. The problem is rare among most of the models, except KB/BERT AOV-B trained on SweQUAD-MC (where it is present for the majority of MCQs).

The next two problems by severity are: *ungrammatical stems* (dark orange in Figure 2), and *ungrammatical alternatives* (light orange in Figure 2). These problems provide a starting point for fixing an MCQ, although still require re-writing major parts of the MCQ. We note that at least one of these two problems is present for every model. The least amount of ungrammatical MCQs were produced by ChatGPT, followed by GPT-3, which is in turn closely followed by KB/BERT AOV-A trained on Quasi.

Strikingly, KB/BERT AOV-B trained on SweQUAD-MC produced *all* MCQs exhibiting one of the three aforementioned problems. For this reason, the model is *excluded* from the further analysis.

Next, we address **Q2, whether or not all stems were answerable by the text**. In fact, the only model with all grammatical stems being answerable by the text is the UD baseline, but it has generated substantially fewer MCQs than the other models. Otherwise, the most frequent reason was that the stems were *undiscussed*, i.e., the answer to the question was not present or inferrable from the text (dark purple in Figure 2). The model with the smallest number of undiscussed stems was GPT-3, followed by ChatGPT. *Contradictive* stems (light purple in Figure 2) and *ambiguous* stems

SST ID	Swedish	English
SST-1	Stockholm är Sveriges huvudstad.	Stockholm is Sweden's capital.
SST-2	Kyiv är Ukrainas huvudstad.	Kyiv is Ukraine's capital.
SST-3	Skranos är Alpongwas huvudstad.	Skranos is Alpongwa's/Alpongwas' capital.
SST-4	Stockholm är Sveriges huvudstad och den största staden i landet.	Stockholm is Sweden's capital and the largest city in the country.
SST-5	Stockholm är huvudstaden och den största staden i Sverige.	Stockholm is the capital and the largest city in Sweden.
SST-6	Kyiv är Ukrainas huvudstad och den största staden i landet.	Kyiv is Ukraine's capital and the largest city in the country.
SST-7	Kyiv är huvudstaden och den största staden i Ukraina.	Kyiv is the capital and the largest city in Ukraine.
SST-8	Skranos är Alpongwas huvudstad och den största staden i landet.	Skranos is Alpongwa's/Alpongwas' capital and the largest city in the country.
SST-9	Skranos är huvudstaden och den största staden i Alpongwa.	Skranos is the capital and the largest city in Alpongwa.
SST-10	Engelska är svårt.	English is difficult.
SST-11	Bengt tycker att engelska är svårt.	Bengt thinks that English is difficult.
SST-12	Anna berättar att Bengt tycker att engelska är svårt.	Anna tells that Bengt thinks English is difficult.
SST-13	Anna är 20 år, Bengt är dubbelt så gammal.	Anna is 20 years old, Bengt is twice as old.
SST-14	Anna är 20 år, Bengt är dubbelt så ung.	Anna is 20 years old, Bengt is twice as young.
SST-15	Bengt är 20 år, Anna är dubbelt så gammal.	Bengt is 20 years old, Anna is twice as old.
SST-16	Bengt är 20 år, Anna är dubbelt så ung.	Bengt is 20 years old, Anna is twice as young.
SST-17 – SST-20 are the same as SST-13 – SST-16, but with the number 20 replaced by 38.		
SST-1*	Stockholm är huvudstaden av Sverige.*	Stockholm is the capital of Sweden.
SST-2*	Kyiv är huvudstaden av Ukraina.*	Kyiv is the capital of Ukraine
SST-3*	Skranos är huvudstaden av Alpongwa.*	Skranos is the capital of Alpongwa.

Table 3: The single-sentence texts (SSTs) used for human evaluation, along with their English translations. Extra SSTs are denoted by asterisks (*).

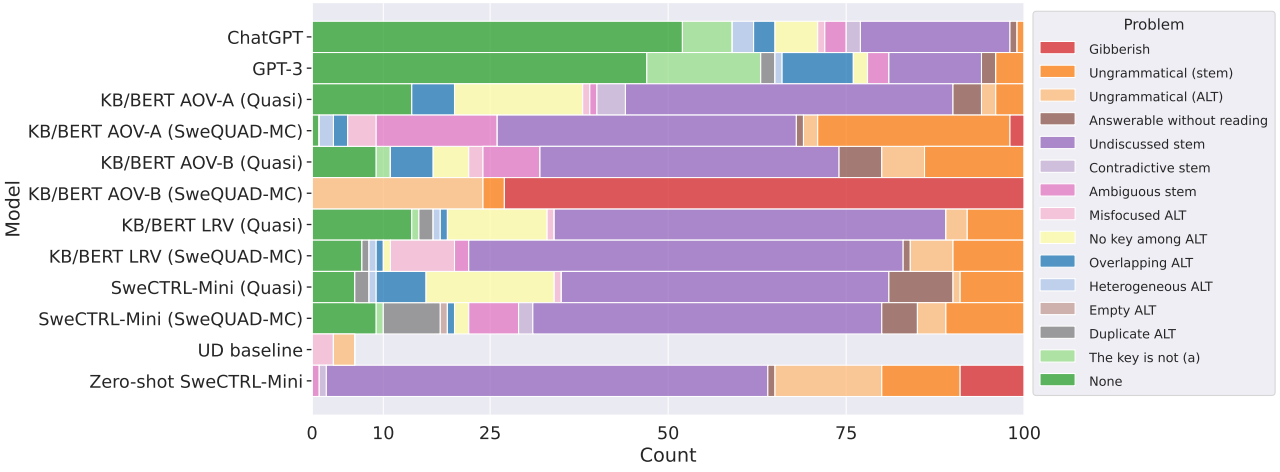

Figure 2: The distribution of problems in the MCQs generated by the 12 evaluated models on the *core* SSTs. The problems are sorted by the severity from the most to the least severe, with *None* (in dark green) indicating the number of MCQs without any aforementioned problems. ALT stands for "alternative(s)".

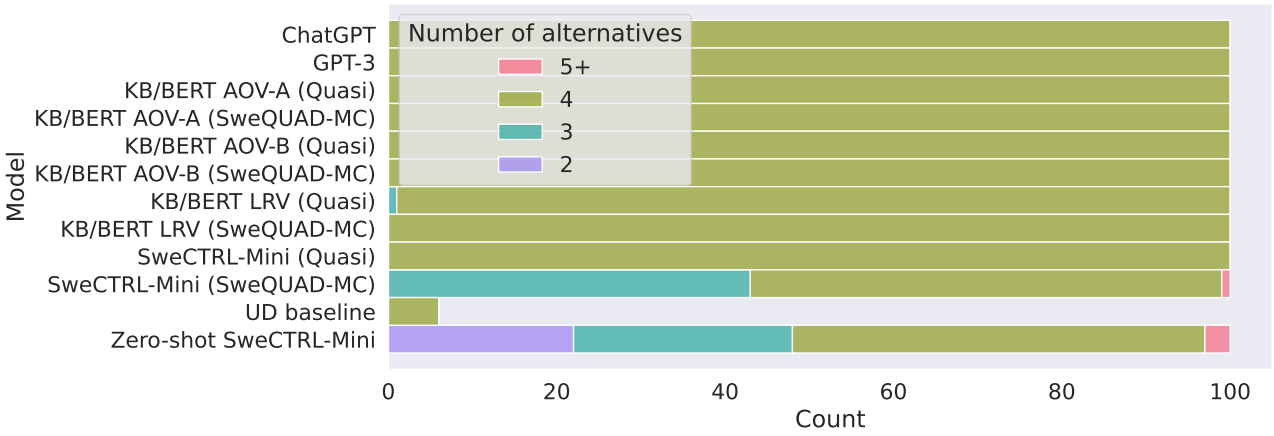

Figure 3: The number of alternatives in the MCQs generated by the 12 evaluated models on the *core* SSTs.

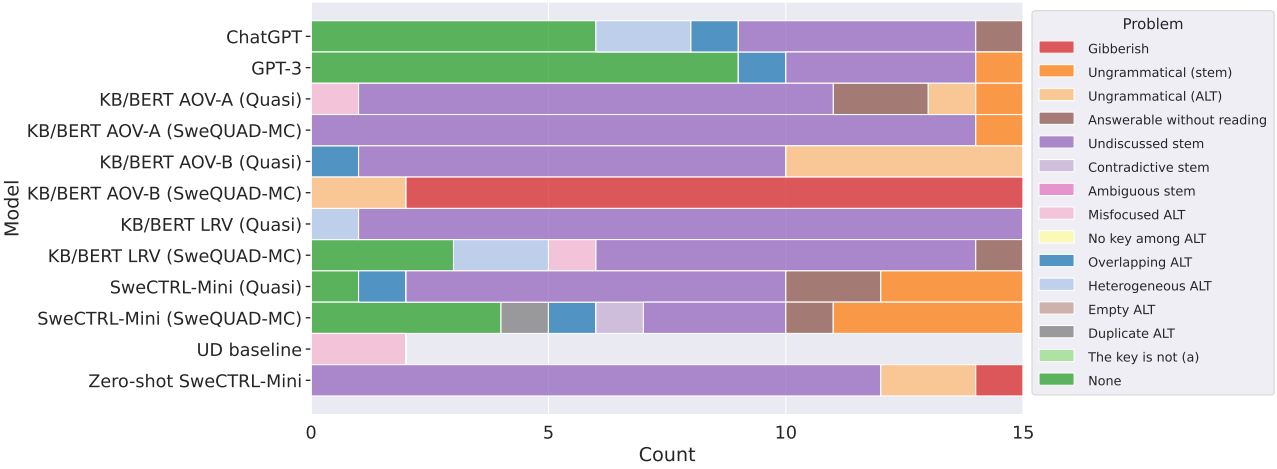

Figure 4: The distribution of problems in the MCQs generated by the 12 evaluated models on the *extra* SSTs (SST-1*, SST-2*, and SST-3*). Gibberish, ungrammatical stems and alternatives do not account for the anglicisms introduced on purpose. ALT stands for "alternative(s)"

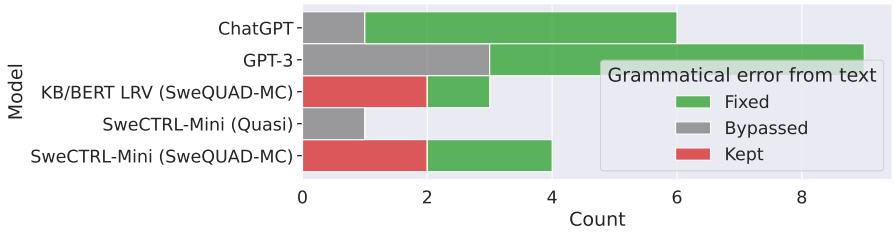

Figure 5: The distribution of *acceptable* MCQs generated by the 12 evaluated models on the *extra* SSTs (SST-1*, SST-2*, and SST-3*) based on whether the introduced anglicisms were fixed, kept or bypassed (by using other formulations).

(dark pink in Figure 2) were much less frequent in comparison (for all models).

Related to Q2, all models produced some MCQs that were answerable without reading the text (dark brown in Figure 2) with SweCTRL-Mini trained on Quasi producing the most such MCQs (by a substantial margin).

To address Q3, whether or not all answer alternatives were relevant for each stem, we note that such problems were infrequent compared to the stem-related problems discussed above. The model with the most MCQs with duplicate alternatives (by a substan-

tial margin) is SweCTRL-Mini trained on SweQUAD-MC, which is also the only model that generated empty alternatives (although for negligibly few MCQs).

Q4 concerns the number of correct answers, of which there should be exactly one, and preferably, this should be alternative (a). This was not always the case: All models except the baselines produced some MCQs with no correct answer at all among the alternatives (light yellow in Figure 2), with the only exception being KB/BERT AOV-A trained on SweQUAD-MC (which had more severe problems for the majority of its MCQs). The two models that produced the most MCQs without a correct answer (in roughly equal amounts) were trained on Quasi, namely KB/BERT AOV-A, and SweCTRL-Mini.

All the models except the baselines also produced MCQs with more than one correct alternative (dark blue in Figure 2). The model with the most such MCQs is GPT-3, whereas the runner-up (with substantially fewer MCQs) is SweCTRL-Mini trained on Quasi.

Finally, as the third and final sub-question of Q4, we checked whether or not the MCQs with exactly one correct alternative indeed had alternative (a) as the key. This was the case for most of the models. Two notable exceptions are GPT-3, and ChatGPT, producing substantially more MCQs with (a) not being the correct alternative (light green in Figure 2).

7.1.2 Error analysis

The distribution of non-problematic MCQs across the single-sentence texts (SSTs) is reported in Figure 6 (only for models that produced at least one such MCQ). Two best-performing models, ChatGPT and GPT-3, have generated at least one acceptable MCQ for almost every SST (except SST-11 for GPT-3). In contrast, generated MCQs for all the other models are distributed more sparsely among the SSTs. The non-GPT model with the best coverage across the SSTs is KB/BERT LRV trained on Quasi with 9 out of 23 SSTs receiving at least one generated MCQ. The very same model is also the model that generated the most acceptable MCQs (14) among non-GPT models. The other model with 14 MCQs is KB/BERT AOV-A trained on Quasi, but it has much worse coverage of only 3 SSTs.

The only two models that generated fully identical MCQs were KB/BERT AOV-A, and KB/BERT LRV, both trained on Quasi. Strikingly, neither ChatGPT, nor GPT-3 produced any fully identical MCQs, despite the fact that the prompt did not require the generated questions to be unique.

Looking closer at the SSTs themselves, we note that **SST-1 to SST-3** follow the same structure "X is Y's capital". The number of acceptable MCQs is the same across these three SSTs for GPT-3, while differs for all the other models. One curious case is the following MCQ generated by ChatGPT based on the SST-1:

> *Vilken stad i Sverige är känt som "Venedig i Norden"?* (*Which city in Sweden is known as "Venice of the North"?*)
> a) *Stockholm*
> b) *Malmö*
> c) *Göteborg*
> d) *Sundsvall*

This is a fully valid MCQ, with (a) being the correct answer, but it has absolutely nothing to do with the actual text of SST-1, which is why it was categorized as having undiscussed stem. This example shows that stems can be undiscussed in many different ways. Sometimes they can relate to the broader topic of the text (e.g., about Stockholm/Sweden) and be entirely valid MCQs in isolation, such as the example above. On the other hand, sometimes they can be completely off topic, such as the MCQ below generated also based on SST-1, but by KB/BERT AOV-B fine-tuned on Quasi.

> *Hur många tandläkare finns det här?*
> (*How many dentists are there here?*)
> a) 2
> b) 3
> c) 5
> d) 10

SST-3 involves using made-up toponyms, namely the country Alpongwa, and its capital Skranos. While both ChatGPT, and GPT-3 managed to produce acceptable MCQs, most other models did not. Interestingly, both ChatGPT, and GPT-3 produced acceptable MCQs that used made-up toponyms that sound plausible in their alternatives, as in the MCQ below produced by GPT-3.

> *Vad är huvudstaden i Alpongwa?*
> (*What is the capital in Alpongwa?*)
> a) *Skranos*
> b) *Pangea*
> c) *Malvin*
> d) *Rislanda*

By contrast, the only acceptable MCQ based on SST-3 produced by a non-GPT model, namely KB/BERT LRV trained on Quasi, did not use made-up toponyms:

> *Var ligger Alpongwas huvudstad?*
> (*Where is the capital of Alpongwa?*)
> a) *Skranos*
> b) *Oslo*
> c) *Göteborg*
> d) *Stockholm*

The next batch, **SST-4 to SST-9**, extend SST-1 through SST-3 with one more piece of information. The new structural templates are "X is the capital and the

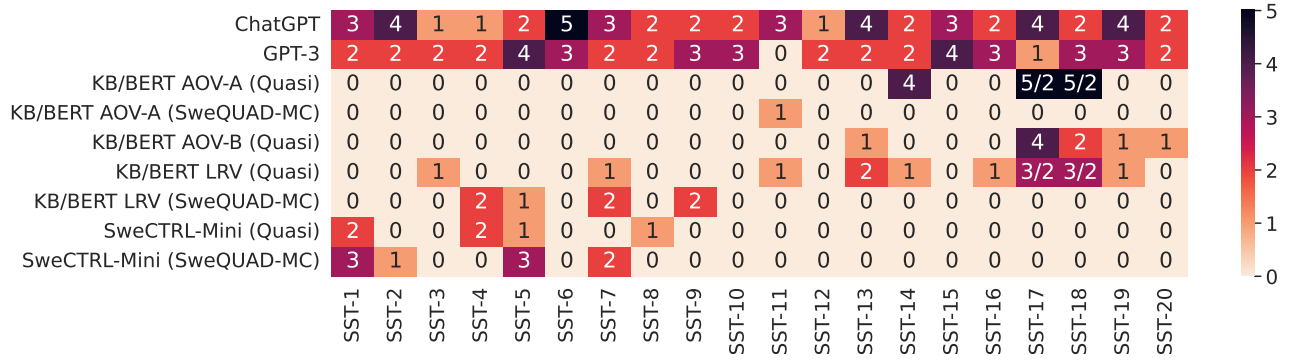

Figure 6: The distribution of the generated MCQs with no problems (dark green in Figure 2). The cells with slashes ("/") indicate cases with fully identical MCQs, the format reads "total MCQs with no problems / of which unique".

largest city in Y" for the evenly numbered SSTs, and "X is Y's capital and the largest city in the country" for oddly numbered SSTs. For these examples we note that the number of acceptable MCQs differs between the pairs of reformulations (i.e. SST-4 and SST-5, or SST-6 and SST-7, or SST-8 and SST-9) for all models. Similarly, GPT-based models managed to produce more MCQs overall, although some fine-tuned models perform on-par on these SSTs, except SST-6 and SST-8.

The following MCQ produced by ChatGPT based on SST-4 is an interesting example of an MCQ with heterogeneous alternatives:

> *Vilken stad är större än Stockholm i Sverige?*
> (*Which city is larger than Stockholm in Sweden?*)
> a. *Ingen, Stockholm är den största staden.*
> (*None, Stockholm is the largest city.*)
> b. *Göteborg.*
> c. *Malmö.*
> d. *Uppsala.*

While the alternative (a) is the key, it is clearly longer than all the others. If formulated simply *Ingen* (*None*), then the problem would have disappeared. However, similarly, to undiscussed stems, there are many ways in which the alternatives can be heterogeneous. Additionally, there are different number of alternatives that can "stick out". For instance, in the following MCQ based on SST-9 produced by SweCTRL-Mini trained on Quasi, two alternatives are heterogeneous:

> *Var ligger staden Skranos?*
> (*Where is the city of Skranos?*)
> a) *I Alpongwa* (*In Alpongwa*)
> b) *I huvudstaden* (*In the capital*)
> c) *I Skranos* (*In Skranos*)
> d) *I den kinesiska huvudstaden* (*In the Chinese capital*)

Note that this question is heterogeneous because alternatives (c) and (d) do not use proper names, and the alternative (d) is longer than all the others. Nevertheless, this MCQ was marked as answerable without reading, because the alternative (c) is a correct common-sense alternative simply after reading the stem. Note, however, that the alternative (c) is unlikely to be used in the real reading comprehension tests.

The very same model, SweCTRL-Mini trained on Quasi, produced one of the few acceptable MCQs based on SST-8, which did not use proper names as alternatives.

> *Vilken stad är Alpongwas huvudstad?*
> (*Which city is Alpongwa's capital?*)
> a) *Den största staden* (*The largest city*)
> b) *Den minsta staden* (*The smallest city*)
> c) *Den största floden* (*The largest river*)
> d) *Den högsta punkten* (*The highest point*)

While both (c) and (d) do have nothing to do with cities (and could be viewed as misfocused), such MCQ might still be useful for people just starting to learn the language (which is why it is viewed as acceptable in this evaluation).

Another acceptable MCQ also based on SST-8 without proper names in alternatives was produced by ChatGPT:

> *Hur stor är Skranos jämfört med andra städer i Alpongwa?*
> (*How large is Skranos compared to other cities in Alpongwa?*)
> a. *Störst* (*The largest*)
> b. *Minst* (*The smallest*)
> c. *Andra störst* (*The second largest*)
> d. *Fjärde störst* (*The fourth largest*)

Both this and previous MCQs took advantage of the second clause added to SST-8 compared to SST-3.

Another interesting aspect concerns SST-2, SST-6, and SST-7, namely that the capital of Ukraine has two alternative spellings in English: *Kyiv*, and *Kiev*. All SSTs

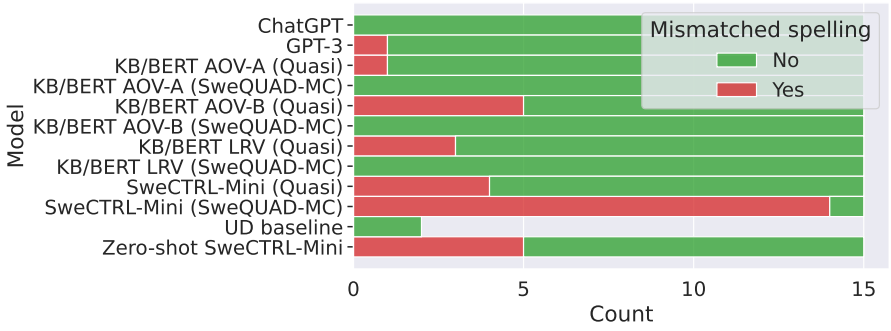

Figure 7: The distribution of the generated MCQs by the presence of mismatched spelling in SST-2, SST-2*, SST-6 or SST-7. The mismatch counts if the spelling of the capital of Ukraine used in the MCQ does not match that of the SST, i.e. Kiev instead of Kyiv (as used in the SSTs).

used the former spelling, but it is relevant to check how well the models comply with the spelling of the text. For that we have manually marked all MCQs that used spelling *Kiev* instead of *Kyiv* present in the SSTs. The distribution of such cases of **mismatched spelling** across the models for the four aforementioned SSTs is shown in Figure 7. We note that the models mostly complied with the spelling of the text, except SweCTRL-Mini trained on SweQUAD-MC which used the alternative spelling most of the time. Interestingly, one of the MCQs based on SST-6 produced by ChatGPT, explicitly asked about the spelling difference:

> *Vad är en annan beteckning för Kyiv?*
> (*What is another term for Kyiv?*)
> a) Kiev
> b) Krakow
> c) Vilnius
> d) Belgrad

Because the nature of the MCQ required the use of both spellings, this was not counted as a mismatch in spellings.

The next batch, namely **SST-10 to SST-12**, are opinion-based statements. Their aim is to test whether the models will add constructs like "according to X", or "as X thinks". We note that most of the models failed to produce an acceptable MCQ on any of these SSTs. Two notable exceptions are ChatGPT and GPT-3 (although GPT-3 also failed on SST-11). Single acceptable MCQs were also produced by two models based on KB/BERT, and one model based on SweCTRL-Mini. In this batch, SST-10 is the interesting one, because the statement sounds like a fact, but it clearly is an opinion, although without specifying whose. The best and only possible kind of MCQ for this SST was produced by ChatGPT:

> *Vilket påstående beskriver författarens åsikt om engelska?*
> (*Which statement describes the author's opinion about English?*)
> a) *Engelska är svårt.* (*English is hard.*)
> b) *Engelska är lätt.* (*English is easy.*)
> c) *Engelska är roligt.* (*English is fun.*)
> d) *Engelska är tråkigt.* (*English is boring.*)

A similar approach was taken by GPT-3 in the following MCQ:

> *Vilken beskrivning är riktig för texten?*
> (*Which description is correct for the text?*)
> a) *Texten beskriver att engelska är svårt*
> (*The text describes that English is hard*)
> b) *Texten beskriver att franska är svårt*
> (*The text describes that French is hard*)
> c) *Texten beskriver att spanska är svårt*
> (*The text describes that Spanish is hard*)
> d) *Texten beskriver att tyska är svårt*
> (*The text describes that German is hard*)

The acceptable MCQs generated on SST-11 and SST-12 took into account that the statements were opinions. For instance, as in the following (and only) acceptable MCQs generated based on SST-11 by KB/BERT LRV trained on Quasi:

> *Vad är det för språk som Bengt tycker är svårt?*
> (*What language does Bengt find difficult?*)
> a) Engelska (*English*)
> b) Svenska (*Swedish*)
> c) Italienska (*Italian*)
> d) Spanska (*Spanish*)

The next batch of SSTs, namely **SST-13 to SST-20**, have to do with counting (understanding who is twice as old, or twice as young). One of the most common mistakes in MCQs for these SSTs had to do with wrong counting. Notably, none of the models based on SweCTRL-Mini were able to produce any acceptable MCQ for this batch. This is also the only batch with absolutely identical acceptable MCQs produced

by KB/BERT AOV-A, and KB/BERT LRV, both trained on Quasi.

Additionally, this was the only batch including MCQs with hypothetical stems (all of which were produced by ChatGPT), for instance, the following MCQ based on SST-16:

> *Om Bengt fyller 25 år, hur gammal kommer Anna att vara då?*
> (*If Bengt turns 25, how old will Anna be then?*)
> a) *30 år* (*30 y.o.*)
> b) *15 år* (*15 y.o.*)
> c) *10 år* (*10 y.o.*)
> d) *40 år* (*40 y.o.*)

SST-16 tells us that Bengt is 20, and Anna is twice as young, meaning 10. Then if Bengt turns 25, Anna will also become five years older, meaning 15, hence the correct alternative is (b), and the MCQ was marked as not having (a) as the key. In fact, out of nine generated hypothetical MCQs, only one was acceptable, while three MCQs had ambiguous stems, the other three MCQs did not have (a) as the correct alternative, and the final two MCQs did not provide the key at all. Nevertheless, it is interesting to observe that current state-of-the-art models are capable of producing more challenging MCQs with hypothetical stems. However, one could argue that such MCQs test skills in mathematics, rather than in reading comprehension, a discussion that we will not develop further in this article.

7.2 Plugga

For texts in Plugga we requested N_q^T MCQs per text as calculated by the Equation 1. Both GPT-3 and ChatGPT were able to accommodate the whole input text at once. However, some texts were too long for the context windows of KB/BERT and SweCTRL-Mini (recall that the input for these approaches has to include the input text, *and* as many masked tokens as the final output will contain). In these cases we split the text into multiple parts of L_T tokens each. Often this would result into the last chunk of the text to be be left as a remainder with $< L_T$ tokens. This last chunk could even constitute one sentence, which might not always be enough to generate an MCQ. Hence we took the last L_T tokens as the last chunk meaning that the last and the penultimate pieces of text will overlap. To exemplify, if $L_T = 3$ and the text is `A B C D E F G`, we would split the text into the following chunks: (`A B C`), (`D E F`), (`E F G`). This, together with the fact that the Equation 1 includes rounding up, means that the smaller the L_T, the more MCQs the model will generate compared to the models with larger L_T on the same texts.

7.2.1 Overview of the results

In summary, the GPT-based models, ChatGPT and GPT-3, produced a substantially larger number of acceptable MCQs (63.7% and 37.1%, respectively), compared to the other models. This result is similar to the results of the SST-based evaluation.

An overview of the problems found in the output MCQs is presented in Figure 8, and, similarly to the evaluation on SST, the histogram accounts only for *the most severe problem* for each MCQ. For the sake of brevity, we will omit to list the datasets that the models were trained on in the remainder of the section, since both models trained on KB/BERT were trained on the Quasi dataset, and there is only one version of SweCTRL-Mini.

Similarly to the SST evaluation, we are interested in the very same Q0 - Q4 aspects. Recall that these aspects have been defined as follows:

- Q0. Did the model produce the requested number N_q^T of MCQs?
- Q1. Are the question (stem) and all alternatives grammatically correct?
- Q2. Is the stem answerable by the text?
- Q3. Are all alternatives relevant for the given stem?
- Q4. Is the alternative (a) the only correct answer?

To address Q0, the number of results produced by the model, only the fine-tuned models generated the requested number of MCQs (shown by the black dashed lines in Figure 8). ChatGPT was slightly short of the target (generating 91.8% of the requested MCQs), while GPT-3 was substantially off the target (generating 33.8% of the requested amount). Concerning the distribution of under-generated MCQs in Figure 10, we note the ChatGPT generated fewer MCQs only for one text due to it reaching the maximum context window size. In contrast, GPT-3 generated fewer MCQs on most of the texts without reaching the maximum context window size for any of the texts.

Related to Q0, the models produced different number of alternatives, as reported in Figure 9. Similarly to SST-based evaluation, vast majority of MCQs contain four alternatives except SweCTRL-Mini that mostly featured MCQs with three alternatives.

To address Q1, the issue of grammatical correctness, all models produced some MCQs with ungrammatical stems and/or alternatives. However, both ChatGPT and GPT-3 produced a very small number of such MCQs. All fine-tuned models generated a larger amount of ungrammatical MCQs compared to GPT-based models, with SweCTRL-Mini even producing a couple of gibberish MCQs. It should be noted that gibberish here included some loose tokens for one of the

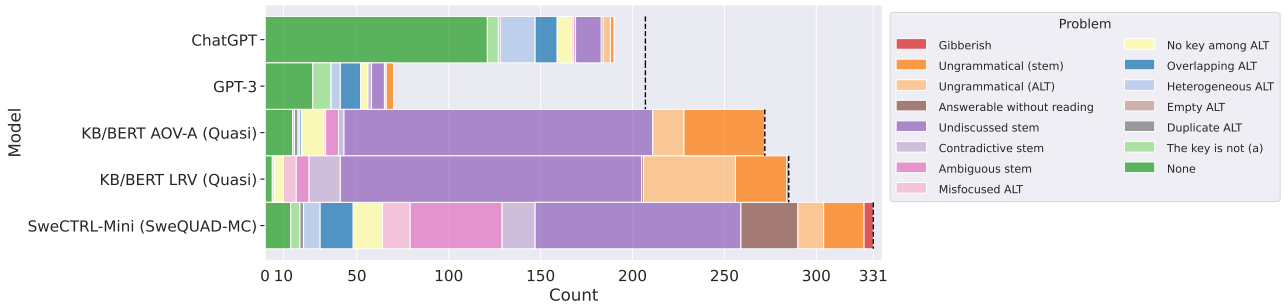

Figure 8: The distribution of problems in the MCQs generated by the TOP-5 best-performing models on SSTs on the texts from Plugga. Black dashed lines indicate the requested number of MCQs to be generated. The problems are sorted by the severity from the most to the least severe, with *None* (in dark green) indicating the number of MCQs without any aforementioned problems. ALT stands for "alternative(s)".

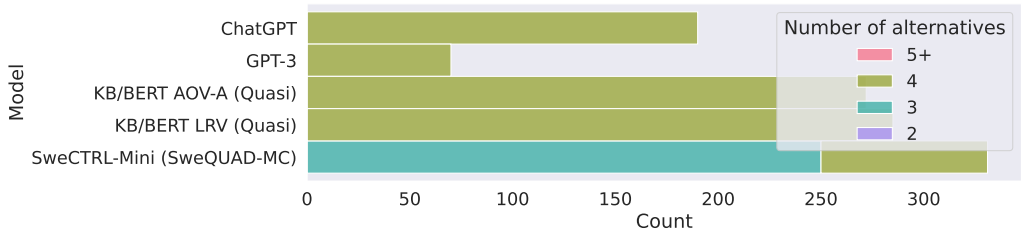

Figure 9: The number of alternatives in the MCQs generated by the 12 evaluated models on the texts from Plugga.

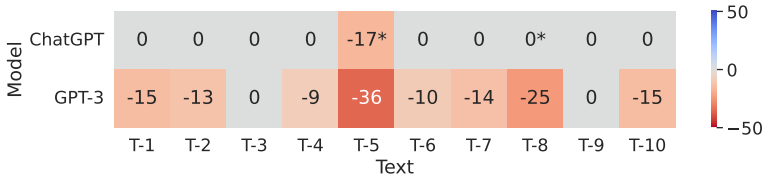

Figure 10: The difference between the actual number of generated MCQs and the requested ones (negative values mean under-generation, positive ones – over-generation, zeros – exactly on point) The cells with the asterisk (*) indicate the cases when the model stopped generation because it reached the maximum context window size.

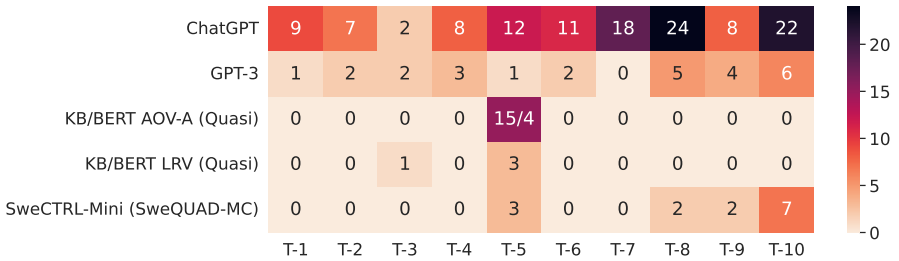

Figure 11: The distribution of the generated MCQs with no problems (dark green in Figure 2). The cells with slashes ("/") indicate cases with fully identical MCQs, the format reads "total MCQs with no problems / of which unique".

alternatives (while the rest of the MCQ is OK), which is radically different from most gibberish encountered in the evaluation on the SSTs previously.

Next, we address **Q2, whether or not all stems were answerable by the text**. In fact, not all stems were answerable by the text, with the most frequent reason (similar to SSTs) being that the stems were undiscussed (dark purple in Figure 8). The pattern is similar to the evaluation on SSTs with the least undiscussed stems being generated by GPT-3, fol-

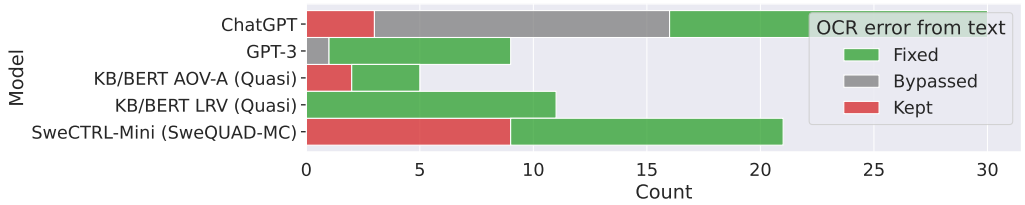

Figure 12: The distribution of *all* MCQs generated by the evaluated models on the Text 8 from Pluga based on whether the OCR error was fixed, kept or bypassed (by using other formulations).

lowed by ChatGPT. Interestingly SweCTRL-Mini generated fewer undiscussed MCQs than models based on KB/BERT, despite producing more MCQs in total. Both contradictive (light purple in Figure 8) and ambiguous (dark pink in Figure 8) stems were much more infrequent in comparison. The largest number of contradictive stems was generated by SweCTRL-Mini (18 MCQs), closely followed by KB/BERT LRV (17 MCQs). The model with a substantially larger amount of ambiguous stems compared to the other models is SweCTRL-Mini.

Related to Q2, one model, namely SweCTRL-Mini, produced substantially more MCQs that are answerable without reading the text (dark brown in Figure 8) compared to the other models. The other two models producing a single such MCQ each were KB/BERT LRV and ChatGPT.

To address Q3, whether all answer alternatives were relevant for each stem, we note that this was not always the case. Similarly to the SST-based evaluation, both misfocused and heterogeneous alternatives are infrequent compared to stem-related problems. The model with the largest number of misfocused alternatives (light pink in Figure 8) is SweCTRL-Mini. The model with the most heterogeneous alternatives (light blue in Figure 8) is still ChatGPT. Additionally, no model produced any empty alternatives, whereas all models except GPT-3 produced at most two MCQs with duplicate alternatives.

Q4 concerns the number of correct answers, of which there should be exactly one, and preferably, this should be alternative (a). This was not always the case: All models produced some MCQs with no correct answer at all among the alternatives (light yellow in Figure 8). The two models that produced the most such MCQs (with roughly equal amounts) were KB/BERT AOV-A and SweCTRL-Mini.

All the models, except KB/BERT LRV, produced MCQs with more than one correct alternative (dark blue in Figure 8).The model with the most such MCQs is SweCTRL-Mini, whereas the runner-ups (with a roughly equal number of such MCQs) are the GPT-based models.

Finally, as the third sub-question of Q4, all models, except KB/BERT LRV, produced some MCQs with the correct alternative not being (a) (and without any more severe problems).

7.2.2 Error analysis

The distribution of non-problematic MCQs across the texts in Plugga is presented in Figure 11. The two best-performing models, ChatGPT and GPT-3, have generated at least one acceptable MCQ for almost every text (except Text 7 for GPT-3). Similarly to the SST-based evaluation, the acceptable MCQs generated by all the other models are distributed more sparsely among the texts. The non-GPT model with the best coverage across the texts in Plugga is SweCTRL-Mini with 4 out of 10 texts resulting in at least one generated MCQ. The only model that generated several MCQs that were completely identical is KB/BERT AOV-A, as it also did for the SST-based evaluation. With this in mind, SweCTRL-Mini is also the best non-GPT model when it comes to the number of unique generated MCQs.

Most of the generated MCQs asked about facts or details presented in the text, with only two MCQs, both produced by ChatGPT, requiring high-level text-based inference. One such MCQ based on the Text 8 is presented below:

> *Vad är budskapet i denna historia?*
> (*What is the message in this story?*)
> a) *Det är viktigt att hjälpa andra* (*It is important to help others*)
> b) *Det är bäst att inte bry sig om andra* (*It is best to not care about others*)
> c) *Det är farligt att hjälpa främlingar* (*It is dangerous to help strangers*)
> d) *Det är bäst att vara egoistisk* (*It is best to be selfish*)

Finally, text 8 from Plugga (a news article about the boy, *Josef*, who helped to save the girl in a wheelchair from a snow trap), we purposefully kept one small OCR error that misspelled the boy's name to *Joset**. In contrast to the spelling example from the SST-based evaluation (*Kyiv vs Kiev*), here using the spelling from the

text is undesirable. Only three evaluated models managed to generate at least one MCQ for this text, and the distribution of all such MCQs (not only the acceptable ones) based on whether the OCR error was fixed is reported in Figure 12. Notably, the majority of MCQs either fixed or bypassed the misspelling problem by using different formulations. Three models that kept the erroneous spelling are SweCTRL-Mini (for around half of the *its* MCQs), ChatGPT, and KB/BERT AOV-A.

8 Discussion

We have observed a substantial performance difference between ChatGPT (gpt-3.5-turbo-0301) and all other models for the MCQ generation task. We conducted only very limited manual hyper-parameter tuning for the non-GPT models when fine-tuning, which means that there is a possibility that the performance of these models could be boosted further. However, as the datasets used for fine-tuning are quite small, it is unlikely that such a boost will be enough to reach the performance of ChatGPT, which produces acceptable MCQs at a rate at least three times higher.

Additionally, one should bear in mind that the focus of this evaluation was mainly on uncovering **foundational problems** with the generated MCQs. This means that even MCQs judged to be acceptable might not necessarily be of sufficient quality to be useful in a high-stakes testing scenario (such as national exams). For instance, the following MCQ produced by ChatGPT on text 8 was marked as acceptable.

> *Vilken funktion hade tjejens rullstol?* (*Which function did the girl's wheelchair have?*)
> a) *Den var elektrisk* (*It was electrical*)
> b) *Den var manuell* (*It was manual*)
> c) *Den var gjord av trä* (*It was made of wood*)
> d) *Den hade ingen funktion* (*It didn't have any function*)

While, formally, this MCQ did not have any of the problems brought up in this work, it asks about a very minor detail. The type of the wheelchair is mentioned twice in the text but really has nothing to do with the plot of the story. While this MCQ could be useful to test how attentive the reader was, it is not aimed at assessing reading comprehension skills but rather memory, which might also be desirable depending on the target audience for the test. Furthermore, alternative (c) is arguably not plausible, since being made of wood is not a function. Nevertheless, such an alternative might be deemed relevant depending on the target audience of the test (for instance, at the entrance-level classes).

Similarly, the use of "none of the above" alternatives, such as in the MCQ below generated by ChatGPT on Text 7, might not necessarily be desirable.

> *Vad hoppas Fredrika på angående sitt jobb?* (*What does Fredrika hope for regarding her job?*)
> a) *Att hon får ett jobb som känns meningsfullt, även om det inte är så bra betalt* (*That she will have a job that feels meaningful, even if it is not very well paid*)
> b) *Att hon får ett jobb som ger henne hög lön* (*That she will have a high-paying job*)
> c) *Att hon får ett jobb som hon inte behöver engagera sig i* (*That she will have a job that requires minimal effort*)
> d) *Ingenting nämns om vad hon hoppas på angående sitt jobb* (*Nothing is mentioned about what she hopes for regarding her job*)

In fact, Haladyna et al. (2002) report that opinions are split about using alternatives such as (d), which is why such MCQs might not necessarily be judged as acceptable in the larger-scale evaluation.

9 Conclusion

In this article, we have compared the MCQ-generating capabilities of 12 different models, eight models fine-tuned by ourselves, two baselines, as well as GPT-3 (*text-davinci-003*), and ChatGPT (*gpt-3.5-turbo-0301*).

The GPT-based models perform better than the rest of the models. ChatGPT performs substantially better than *all* other models in *all* evaluation settings. GPT-3 performs substantially better than non-GPT models when tested on single-sentence texts, whereas the performance gap on the texts from SFI national tests (the Plugga corpus) is less pronounced in comparison to other models (excluding ChatGPT).

Additionally, we noticed that GPT-based models have an inductive bias for producing MCQs with four alternatives, even when the number of alternatives is unspecified in the prompt.

In our limited experiments with the introduced grammatical errors, we noticed that GPT-based models avoid using anglicisms even if they were introduced in the original text, while other models tend to stick to the text much more frequently. The behavior of the models was less conclusive for the introduced OCR error, except for GPT-3 (which produced substantially fewer MCQs than requested though) and KB/BERT LRV trained on Quasi that did not use the formulation with the OCR error.

At the same time, when sticking to the text was necessary, such as in the example with the alternative spellings of the capital of Ukraine, most models used the spelling from the text with ChatGPT, and some models based on KB/BERT taking the lead. All models

based on SweCTRL-Mini were lagging behind and used the formulation that was more frequent in their training data (148 thousand occurrences of the word *Kiev* vs 2052 occurrences of the word *Kyiv*)[13].

In summary, all conducted evaluations point to the fact that the best model for generating MCQs in Swedish is ChatGPT, followed by GPT-3, followed by SweCTRL-Mini fine-tuned on the SweQUAD-MC corpus. The results based on the toy domain of single-sentence texts (SSTs) closely resemble those on the larger-scale texts from the SFI national exam. This indicates that SST-based evaluation might be a viable lower-cost alternative to the full-scale human evaluation, and warrants more extensive studies on the strength of the correlation, and the extent of the time savings.

References

Araki, Jun, Dheeraj Rajagopal, Sreecharan Sankaranarayanan, Susan Holm, Yukari Yamakawa, and Teruko Mitamura. 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*, pages 1125–1136, Osaka, Japan. The COLING 2016 Organizing Committee.

Brown, Tom, 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.

Chung, Ho-Lam, 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.

Guo, Qi, Chinmay Kulkarni, Aniket Kittur, Jeffrey P Bigham, and Emma Brunskill. 2016. Questimator: Generating knowledge assessments for arbitrary topics. In *IJCAI-16: Proceedings of the AAAI Twenty-Fifth International Joint Conference on Artificial Intelligence*.

Haladyna, Thomas M. 2004. *Developing and validating multiple-choice test items*. Routledge.

Haladyna, Thomas M, Steven M Downing, and Michael C Rodriguez. 2002. A review of multiple-choice item-writing guidelines for classroom assessment. *Applied measurement in education*, 15(3):309–333.

Kalpakchi, Dmytro and Johan Boye. 2021. BERT-based distractor generation for Swedish reading comprehension questions using a small-scale dataset. In *Proceedings of the 14th International Conference on Natural Language Generation*, pages 387–403, Aberdeen, Scotland, UK. Association for Computational Linguistics.

Kalpakchi, Dmytro and Johan Boye. 2022a. Automatically generating question-answer pairs for assessing basic reading comprehension in Swedish. *arXiv preprint arXiv:2211.15568*.

Kalpakchi, Dmytro and Johan Boye. 2022b. Textinator: an internationalized tool for annotation and human evaluation in natural language processing and generation. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 856–866, Marseille, France. European Language Resources Association.

Kalpakchi, Dmytro and Johan Boye. 2023a. Quasi: a synthetic question-answering dataset in Swedish using GPT-3 and zero-shot learning. In *Proceedings of the 24th Nordic Conference on Computational Linguistics (NoDaLiDa)*, pages 477–491, Tórshavn, Faroe Islands. University of Tartu Library.

Kalpakchi, Dmytro and Johan Boye. 2023b. Quinductor: A multilingual data-driven method for generating reading-comprehension questions using universal dependencies. *Natural Language Engineering*, page 1–39.

Kalpakchi, Dmytro and Johan Boye. 2023c. SweCTRL-Mini: a data-transparent Transformer-based large language model for controllable text generation in Swedish. *arXiv preprint arXiv:2304.13994*.

Loshchilov, Ilya and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Majumder, Mukta and Sujan Kumar Saha. 2015. A system for generating multiple choice questions: With a novel approach for sentence selection. In *Proceedings of the 2nd Workshop on Natural Language Processing Techniques for Educational Applications*, pages 64–72, Beijing, China. Association for Computational Linguistics.

Malmsten, Martin, Love Börjeson, and Chris Haffenden. 2020. Playing with Words at the National Library of Sweden – Making a Swedish BERT.

[13] Checked using the publicly available website for SweCTRL-Mini: https://swectrl.dev/data

Nivre, Joakim, Marie-Catherine de Marneffe, Filip Ginter, Jan Hajič, Christopher D. Manning, Sampo Pyysalo, Sebastian Schuster, Francis Tyers, and Daniel Zeman. 2020. Universal Dependencies v2: An evergrowing multilingual treebank collection. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 4034–4043, Marseille, France. European Language Resources Association.

OECD. 2019. *PISA 2018 Assessment and Analytical Framework*. OECD Publishing, Paris.

OECD. 2021. *PISA 2025 Foreign Language Assessment Framework*. OECD Publishing, Paris.

Offerijns, Jeroen, Suzan Verberne, and Tessa Verhoef. 2020. Better distractions: Transformer-based distractor generation and multiple choice question filtering. *arXiv preprint arXiv:2010.09598*.

Qiu, Zhaopeng, 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.

Zhang, Ruqing, Jiafeng Guo, Lu Chen, Yixing Fan, and Xueqi Cheng. 2021. A review on question generation from natural language text. *ACM Transactions on Information Systems (TOIS)*, 40(1):1–43.

Zhou, Xiaorui, Senlin Luo, and Yunfang Wu. 2020. Co-attention hierarchical network: Generating coherent long distractors for reading comprehension. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 9725–9732.