

---

# SweCTRL-Mini: a technical note on training and evaluation (v1.0.1)

---

Dmytro Kalpakchi  
dmytroka@kth.se

## Abstract

This document provides technical details on training and evaluation of SweCTRL-Mini. The document is not meant as an independent source of information and should be viewed as complementary to the original article. Terms from the original article might be used here without explanation.

## 1 Training data

For training we have used 2 data sources: the Swedish part of mC4 [Xue et al., 2021] and texts from Project Runeberg<sup>1</sup>. In both cases, however, we have chosen only a subset of data following the procedures outlined in the next section.

### 1.1 Filtering

#### 1.1.1 mC4

*mC4* is a subset of Common Crawl<sup>2</sup>, which provides the scraped Internet webpages, and their URLs. In order to navigate *the Swedish part of mC4* we have extensively used the provided URLs, which played a decisive role on whether a piece of text is to be included in the training data. We have excluded all pages from the websites that satisfy one of the following *exclusion criteria*:

1. contain personal information (e.g., `hitta.se`);
2. are hub-like, i.e., provide a collection of links to other websites (e.g., `pressen.se`);
3. most of the content is not in Swedish (e.g., personal blogs in Finnish or Arabic);
4. there seems to be an excessive number of page elements that are hard to clean (e.g., names of buttons like “Login”, “Show more comments”, “Like”, or links to media files);
5. the scraped texts include CSS/JavaScript, or those relying on dynamic content (e.g., quiz pages in the newspapers);
6. webpages for categories/tags containing links to all articles within this category/tag;
7. the Wikipedia utility pages (e.g., template pages or special pages);
8. certain types of URLs which contain repetitive text (e.g., webpages under paywall);
9. the text contains Unicode parsing errors (specifically, the unicode replacement character U+FFFD)

---

<sup>1</sup><http://runeberg.org/>

<sup>2</sup><https://commoncrawl.org/>

10. the text is less than 250 characters long.

Criteria 5 - 10 have been checked automatically, immediately excluding around 448 thousand documents (with distribution per exclusion criterion presented in Table 1).

Criterion	Before auto	After auto
JavaScript	130	177
CSS	40	45
Wiki-utility	4270	4344
Quiz URLs	238	262
Tags URLs	128338	146199
Hub URLs	738	774
Paywall URLs	77	91
Unicode errors	46100	48410
< 250 chars	268494	287677

Table 1: Filtering statistics before and after automatic classification

At the same time, criteria 1 - 4 could not be reliably checked automatically, hence we adopted another approach. For each non-excluded webpage we have extracted its network location (later referred to as *netloc*), which typically includes its domain name with subdomains. We have then ordered the extracted netlocs by the number of documents in the corpus, which in total amounted to around 1.2 million netlocs. For each of these netlocs (and better for each particular URL), we had to take 2 decisions: whether to include or exclude it based on the aforementioned criteria, and, if included, which control code it should be associated with. Even examining just all netlocs manually (let alone the URLs) was infeasible, hence we have opted for a hybrid approach by first manual, and then automatic categorization.

For a manual categorization, we took only the netlocs with at least 10000 documents (there were only 356 of those) and examined 10 random documents, which were the basis for providing a *default category* for all webpages having this netloc. The categories were created on the fly trying to accommodate all types of content as best we can. We ended up with 36 content categories (all categories in Table 1 of the original article, except *literature*), and one utility category “Other” for all excluded netlocs. Out of 356 manually examined netlocs, only 123 satisfied our inclusion criteria and have been assigned one of 36 content categories.

In the next step we attempted to make categorization more fine-grained by switching from categorizing netlocs to categorizing particular URLs for all webpages belonging to the 123 included netlocs. This was achieved by employing URL-based heuristics checking each URL against a number of regular expressions<sup>3</sup>, and using its default category as a fallback if none of the regular expressions have worked. At the end of this step, all webpages from 123 included netlocs got their own content category and the webpages from 233 excluded netlocs were assigned to a utility category “Other”.

In the next step, we have used the aforementioned dataset to train a 37-way text classifier that will help us classify all other websites (with less than 10000 documents) that we didn’t categorize manually. We specifically did not perform any cleaning on the texts, because we suspected that the parts that are typically cleaned (e.g., URLs) could provide valuable clues to the classifier. We have experimented with Logistic Regression and Naive Bayes classifiers, representing texts as TF-IDF vectors, and varying a number of hyper-parameters. After a grid search, we have concluded that a Logistic Regression classifier with a minimum document frequency of 100 provided the best results in terms of accuracy on

<sup>3</sup> for the exact list we refer to `cats_by_urls.py` in the associated GitHub repository

the validation set – 88.32%. For more details on this step, please consult Section 2. After applying the classifier on the remainder of the web pages, we got our filtered-out training data ready to be cleaned.

### 1.1.2 Project Runeberg

We first downloaded all books in Swedish from Project Runeberg published after 1950, which resulted in 41 books. Then we noticed that more than half of them (21 books) were written by one and the same author, Jan Myrdal. To decrease bias towards one particular author, we have removed 11 of 21 books from the training data. The remaining total of 30 books have been included in the training data and have been classified as literature, which was the 37th content category.

## 1.2 Cleaning

### 1.2.1 mC4

We have cleaned the *mC4* texts using specially crafted sets of regular expressions, specifically designed for texts in the coarse news and Wikipedia categories. We have also had two generic cleaning rules applied to all categories: removing the soft-hyphen U+00AD and latin1 symbol U+0096. For the news articles, the cleaning algorithm proceeded as follows:

- remove original URL, if present, e.g. “Artikeln ursprungsadress: <https://www.dn.se/arkiv/kultur/>”;
- remove recommended articles, e.g. “”LÄS MER/OCKSÅ” + INSÄNDARE or — 10 jun Artikel 50 av 50\nArboga — 30 apr Artikel 49 av 50\nArboga — 23 apr Artikel 48 av 50\nVästerås —”;
- remove e-mails with unknown top-level domains (TLD);
- remove text related to downloads/images, e.g. “\n1,51 MB • 1428 x 2000 px\n2,94 MB • 4500 x 3000 px\n2,88 MB • 4500 x 3000 px”;
- remove date picker data, e.g. “januari 2000 — augusti 2010 — september 2010 — oktober 2010”
- remove logging URLs;
- remove URLs between two “\n”, as those are probably also recommendations.

For Wikipedia articles, we have applied the following algorithm:

- remove meta-text, e.g. “Sidan redigerades/ändrades senast den 15 oktober 2018 kl. 13.48.”, “Läst 24 september 2012.” “Hämtad från <https://...>”;
- remove footnotes;
- remove info about categories and languages, e.g., “Navigeringsmeny”;
- remove references, e.g. “[1]”;
- remove bibliography entries, e.g. “^ Schmadel, Lutz D. (2007). Dictionary of Minor Planet Names – (2801) Huygens. Springer Berlin Heidelberg. sid. 229. ISBN 978-3-540-29925-7.”.

### 1.2.2 Project Runeberg

The downloaded books were in LXML format, which made it possible to split books into chapters. We have then processed each chapter line by line and removed all HTML tags from each line. Then we discarded all lines that:

- contain all uppercase letters;

	Raw	Cleaned
Documents	4.51M	4.51M
Words*	2.11B	2.08B
Characters	13.55B	13.24B

Table 2: Overall data statistics for the filtered Swedish part of mC4. Words here mean non-unique words, where each word is found by splitting a text by spaces.

- are likely to be a copyright notice;
- contain more than half of the words that could not be found in the wordlist, meaning this particular line probably contains lots of OCR errors.

Such approach might lead to the text being incoherent at times, but the hope was that it won't be frequent enough to make the text fully incomprehensible.

## 2 Details on training a text classifier

We specifically did not perform any cleaning on the texts, because we suspected that the parts that are typically cleaned (e.g., URLs) could provide valuable clues to the classifier. We have experimented with Logistic Regression and Naive Bayes classifiers, representing texts as TF-IDF vectors, and varying a number of hyper-parameters. After a grid search, we have concluded that a Logistic Regression classifier with a minimum document frequency of 100 provided the best results in terms of accuracy on the validation set – 88.32%. After applying the classifier on the remainder of the web pages, we got our filtered-out training data ready to be cleaned.

## 3 Experimental results of generation hyper-parameter search

SweCTRL-Mini, as the original CTRL, is an autoregressive model with left-to-right generation, meaning that text generation happens one token at a time, where at all times the model is given access to the previously generated tokens (called *context*). The next word is decided based on the softmax distribution for the next token provided by the model. Ideally, one would want to rely on the model, and simply take a token with the highest probability (greedy generation). However, such approach have been previously shown to lead to text stuck in loops, which we will refer to as *sampling loops*. The most frequently used approach to remedy this is to use sampling from the softmax distribution returned by the model.

Sampling can also be performed in a multitude of ways, where the resulting distribution is tweaked via *generation hyper-parameters* to increase/decrease certain probabilities or to limit the number of available tokens. In this paper, we attempted to perform a systematic search through some of these hyper-parameters (HP), specifically:

- the temperature  $T \in (0, 1]$ , that requires tweaking the softmax distribution as follows:

$$p_i = \frac{\exp(\frac{x_i}{T})}{\sum_k \exp(\frac{x_k}{T})}$$

- the repetition penalty  $r$ , proposed by [Keskar et al., 2019], that requires tracking the set of already generated tokens  $g$  and changing the sampling distribution to:

$$p_i = \frac{\exp(\frac{x_i}{I(x_i \in g)})}{\sum_k \exp(\frac{x_k}{I(x_k \in g)})},$$

where  $I(x_i \in g)$  is an indicator function that is equal to  $r$  if the token is in the set  $g$ , or 1 otherwise.

- the nucleus threshold  $p$ , introduced by [Holtzman et al., 2019], that samples among the subset of tokens  $V^{(p)}$ , known as nucleus, which is chosen such that:

$$\sum_{x_i \in V^{(p)}} p_i \geq p.$$

The new distribution over  $V^{(p)}$  is re-normalized, for the probabilities to sum to 1.

In all equations above  $x_i$  is a token that has index  $i$  in the vocabulary,  $p_i$  is the probability for  $x_i$  to be sampled.

We have employed a grid search over hyper-parameters in *pairs*:  $(p, r)$ , and  $(T, r)$ . The value grids were defined as follows:

- $p$  should range from 0.7 to 1 with a step of 0.05 (7 values in total);
- $T$  should range from 0 (in practice replaced by  $T = 1 \times 10^{-15}$ ) to 1 with a step of 0.2 (6 values in total);
- $r$  should range from 1 to 2 with a step of 0.2 (6 values in total).

The rationale behind such hyper-parameter pairing is that  $p$  and  $T$  attempt to achieve the same goal: denying sampling of highly improbable tokens (with a varying definition of “highly improbable”). On the other hand,  $r$  has a different goal: prevent sampling loops, and, as a by-product, increase lexical diversity.

For each configuration of hyper-parameters we have sampled 100 texts for each of the 11 major categories. Per category, we have sampled  $7 \cdot 6 \cdot 100 = 4200$  texts for  $(p, r)$ -search, and  $6 \cdot 6 \cdot 100 = 3600$  texts for the  $(T, r)$ -search, amounting to 7800 texts. In total, over 11 categories, we have sampled 85800 texts, using the same settings for all experiments:

- the prompt included only the opening control code (OCC), corresponding to the category;
- the generation process was stopped when reaching either the matching ending control code (ECC), or the length of 256 tokens in total (whichever comes first).

Given the large number of sampled texts, we have pre-analyzed the results automatically using the following automatic metrics.

1. **Presence of ECC** in Figures 1 - 11). The idea with ECC is that SweCTRL-Mini should learn how to start and end texts in each genre appropriately, hence when the OCC and ECC do not match, this might potentially indicate the topic/genre shift, which is undesirable.
2. **Size of sampling loops** (subplots (b) and (f) in Figures 1 - 11). We have calculated the number of contiguously repeating phrases (up to the length of 5 tokens). Evidently, no such phrases are desirable at all, except, if for the case when *all* repeated tokens are numerals (such as “22” being tokenized as double “2”), which was excluded from the analysis.
3. **Number of generated tokens** (subplots (c) and (g) in Figures 1 - 11). There are no clear-cut rules for this metric, but our goal was to get as large variance as possible on this category to be able to generate texts of varying length.
4. **BLEU-4** (subplots (d) and (h) in Figures 1 - 11), introduced by [Papineni et al., 2002], to measure the lexical diversity of the generated texts (contrary to the typical use of BLEU-based metrics, *the lower, the better*, in this context).

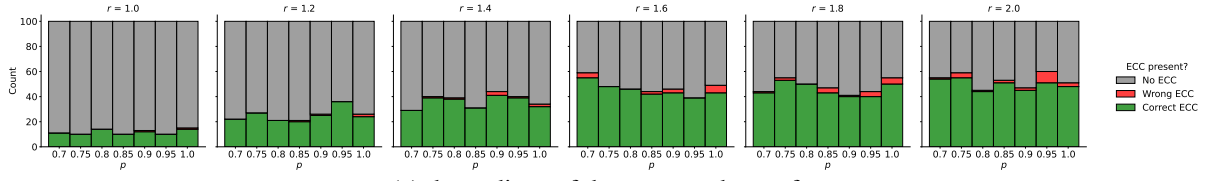
The initial trends observed in Section 5.3 of the original article are based on Figures 1 - 11 of this section.

#### **4 Human evaluation guidelines**

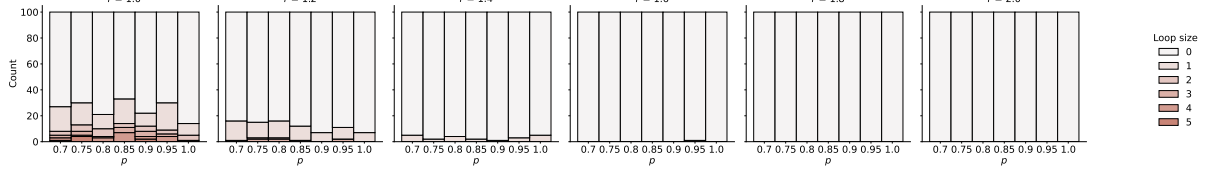
The original guidelines for human evaluation in Swedish are provided in Figure 12.

#### **References**

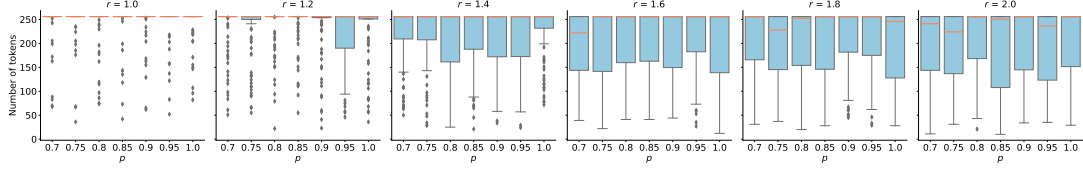
- [Holtzman et al., 2019] Holtzman, A., Buys, J., Du, L., Forbes, M., and Choi, Y. (2019). The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751*.
- [Keskar et al., 2019] Keskar, N. S., McCann, B., Varshney, L. R., Xiong, C., and Socher, R. (2019). Ctrl: A conditional transformer language model for controllable generation. *arXiv preprint arXiv:1909.05858*.
- [Papineni et al., 2002] Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.
- [Xue et al., 2021] Xue, L., Constant, N., Roberts, A., Kale, M., Al-Rfou, R., Siddhant, A., Barua, A., and Raffel, C. (2021). mt5: A massively multilingual pre-trained text-to-text transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 483–498.



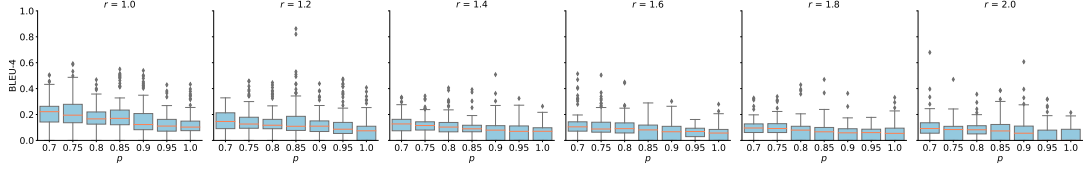
(a) the endings of the generated texts for



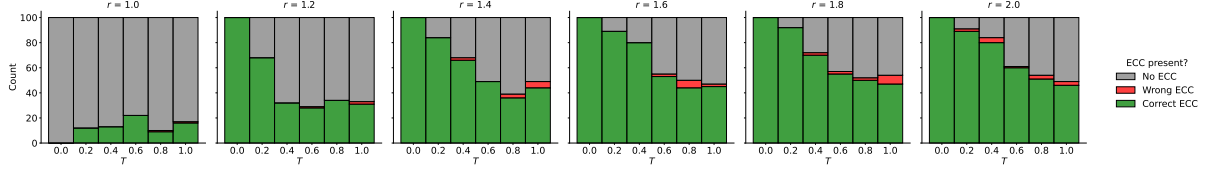
(b) the sampling loop size (up to 5 tokens) for the nucleus threshold  $p$



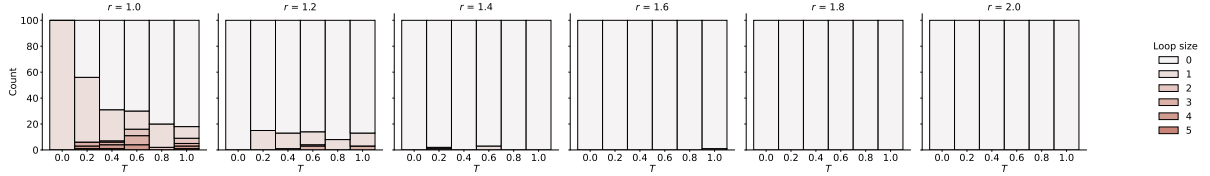
(c) the number of generated tokens for the nucleus threshold  $p$



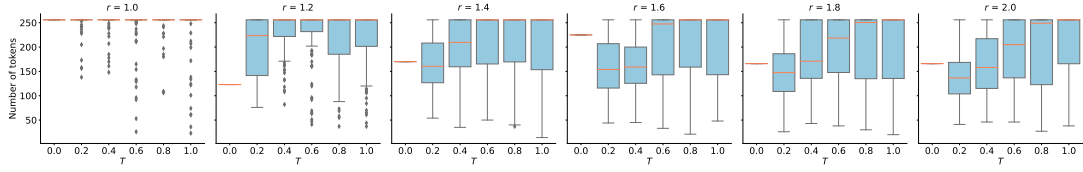
(d) BLEU-4 scores for the nucleus threshold  $p$



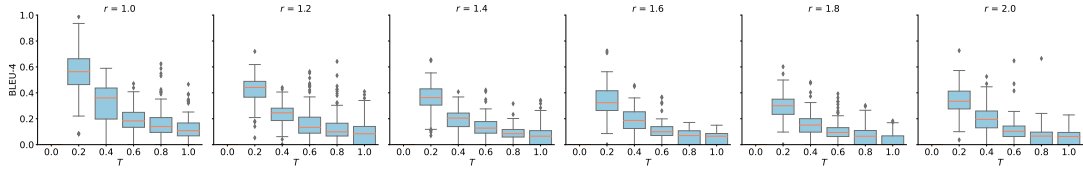
(e) the endings of the generated texts for



(f) the sampling loop size (up to 5 tokens) for the temperature  $T$



(g) the number of generated tokens for the temperature  $T$



(h) BLEU-4 scores for the temperature  $T$

Figure 1: Automatic metrics for the generation HP search for the “admin” category

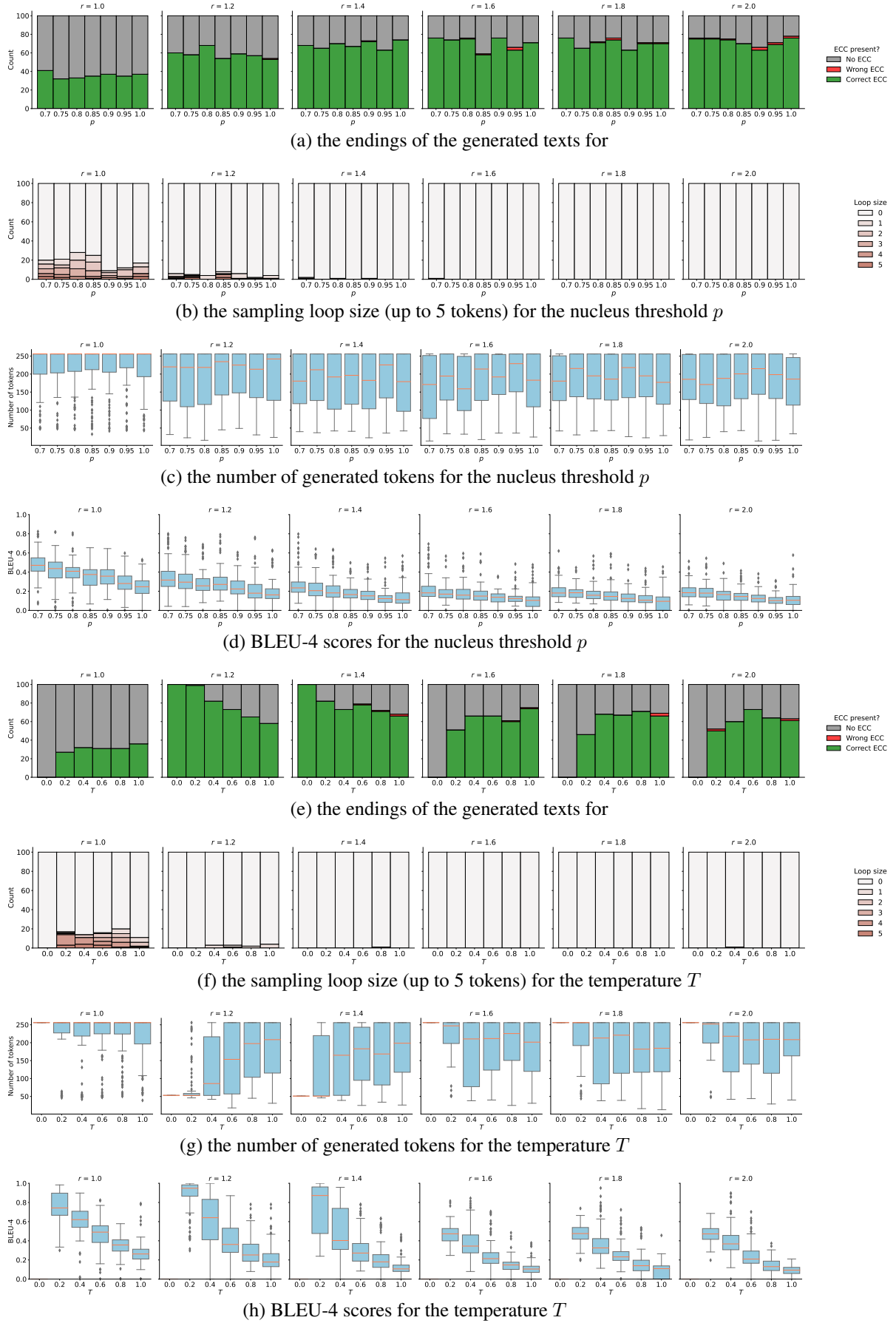
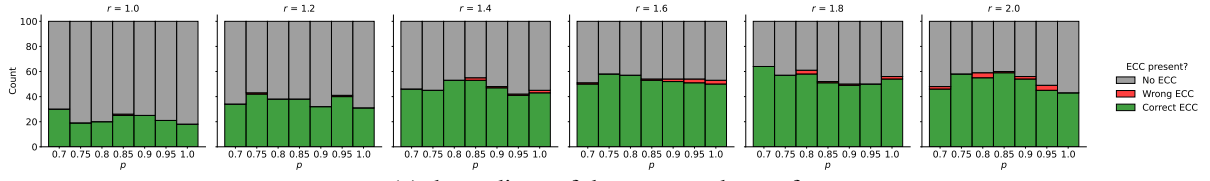
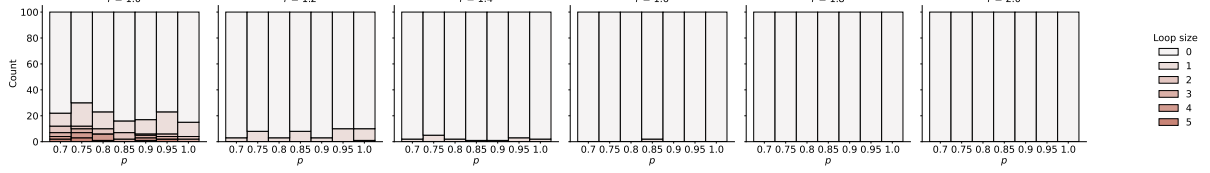


Figure 2: Automatic metrics for the generation HP search for the “ads” category

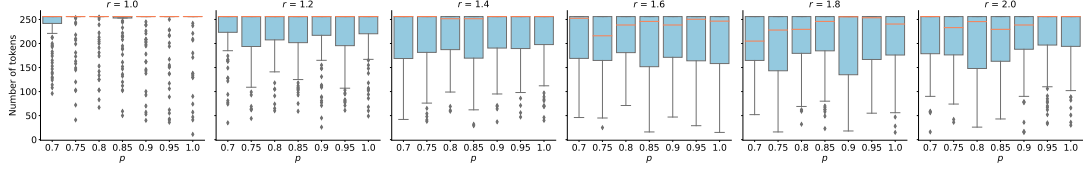




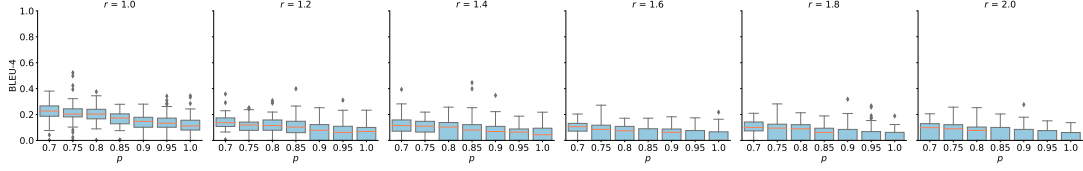
(a) the endings of the generated texts for



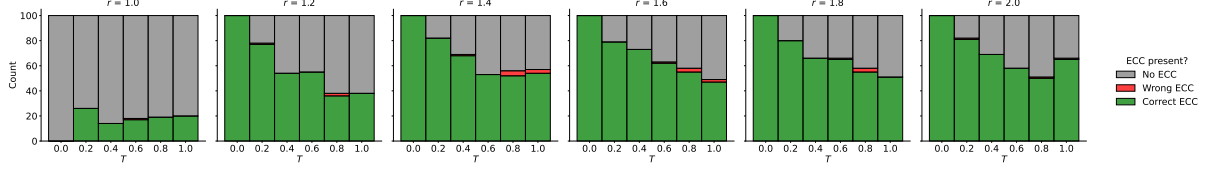
(b) the sampling loop size (up to 5 tokens) for the nucleus threshold  $p$



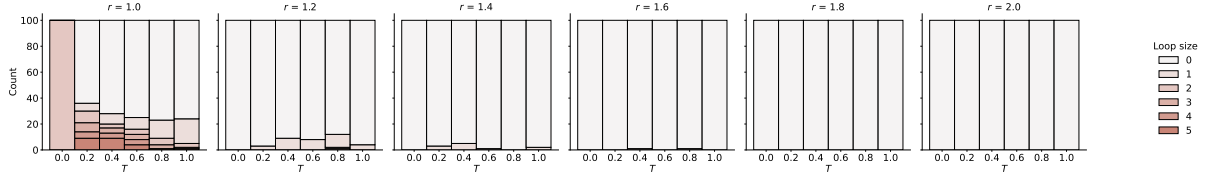
(c) the number of generated tokens for the nucleus threshold  $p$



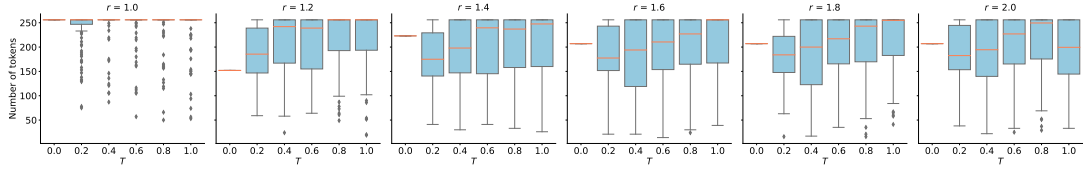
(d) BLEU-4 scores for the nucleus threshold  $p$



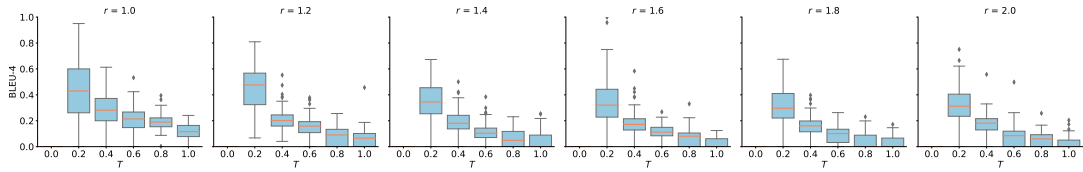
(e) the endings of the generated texts for



(f) the sampling loop size (up to 5 tokens) for the temperature  $T$

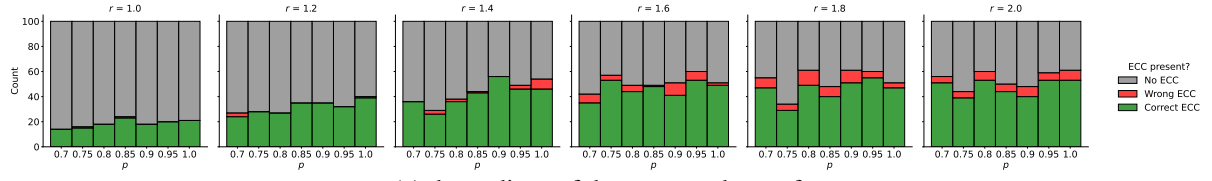


(g) the number of generated tokens for the temperature  $T$

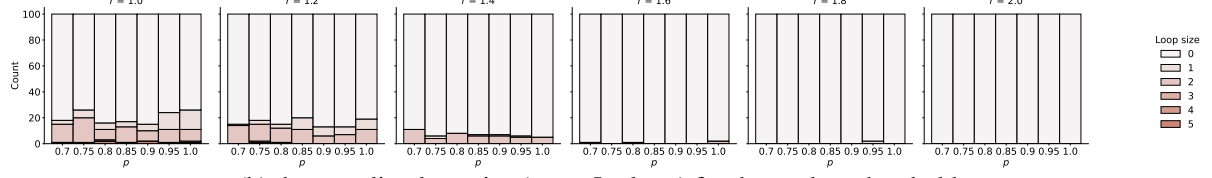


(h) BLEU-4 scores for the temperature  $T$

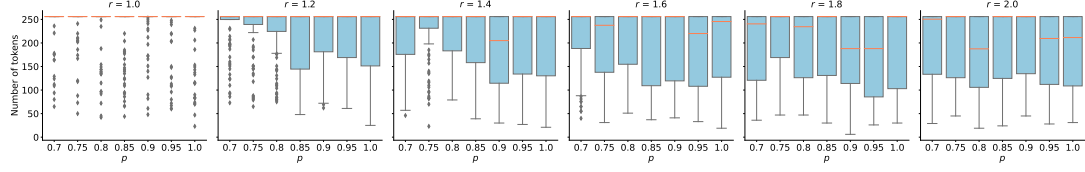
Figure 3: Automatic metrics for the generation HP search for the “blogs” category



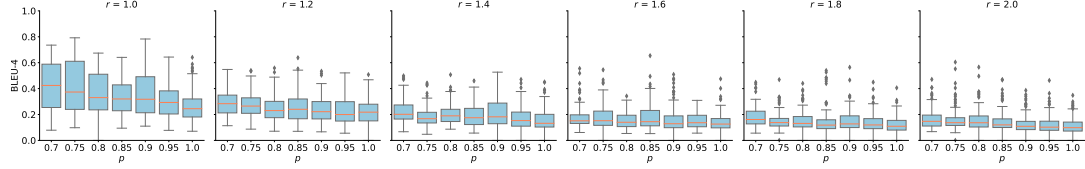
(a) the endings of the generated texts for



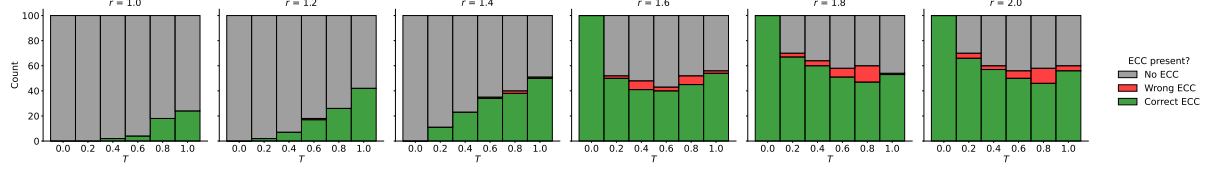
(b) the sampling loop size (up to 5 tokens) for the nucleus threshold  $p$



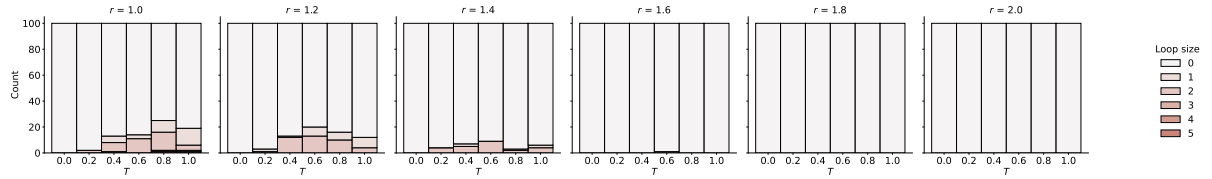
(c) the number of generated tokens for the nucleus threshold  $p$



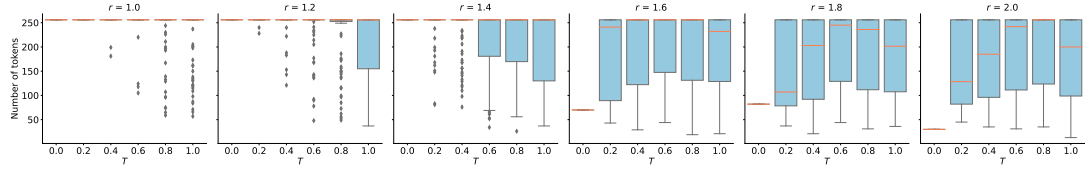
(d) BLEU-4 scores for the nucleus threshold  $p$



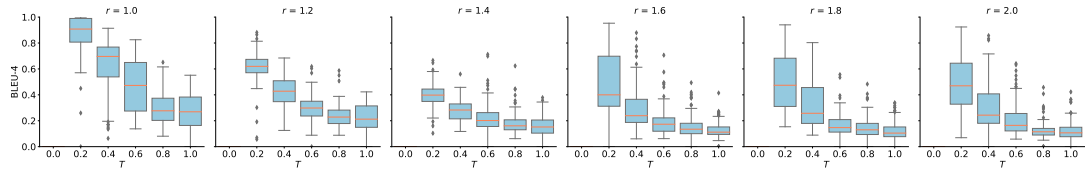
(e) the endings of the generated texts for



(f) the sampling loop size (up to 5 tokens) for the temperature  $T$

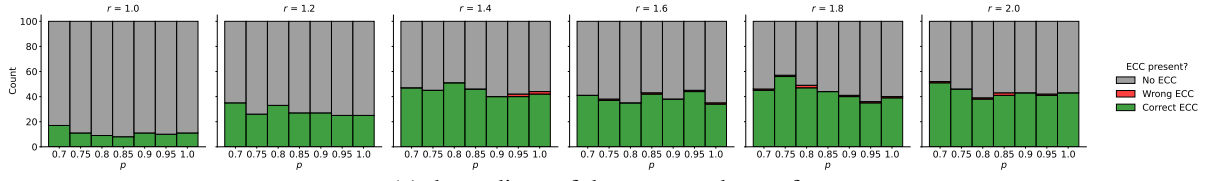


(g) the number of generated tokens for the temperature  $T$

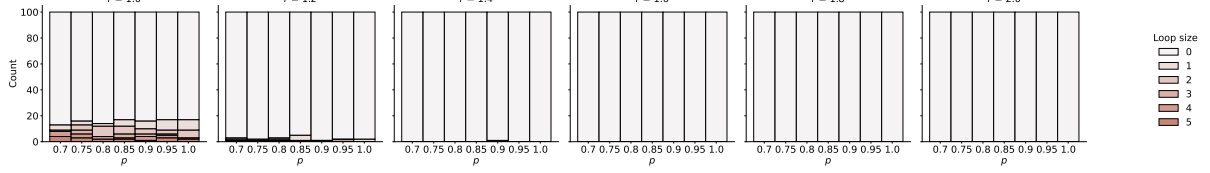


(h) BLEU-4 scores for the temperature  $T$

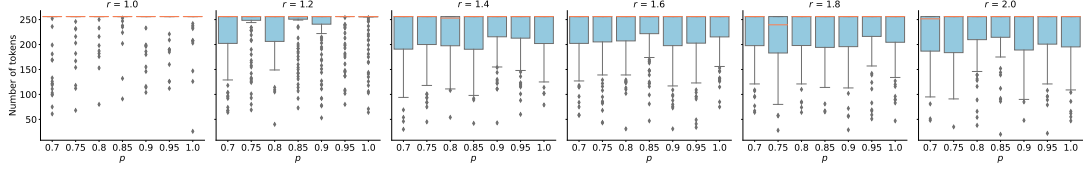
Figure 4: Automatic metrics for the generation HP search for the “debate” category



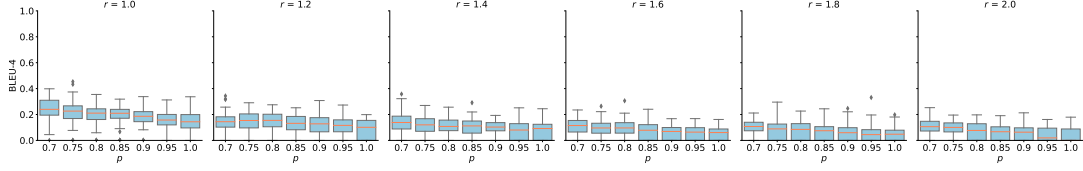
(a) the endings of the generated texts for



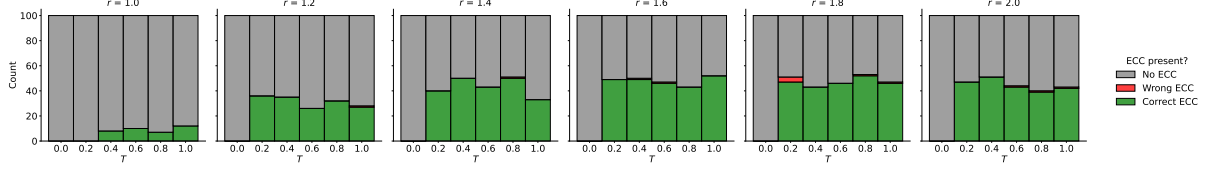
(b) the sampling loop size (up to 5 tokens) for the nucleus threshold  $p$



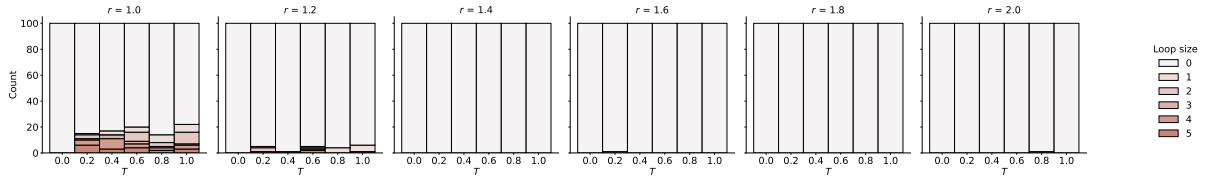
(c) the number of generated tokens for the nucleus threshold  $p$



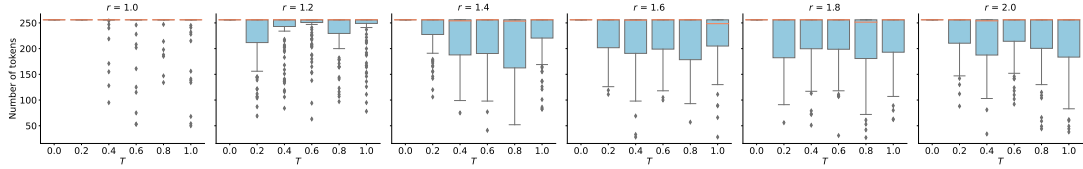
(d) BLEU-4 scores for the nucleus threshold  $p$



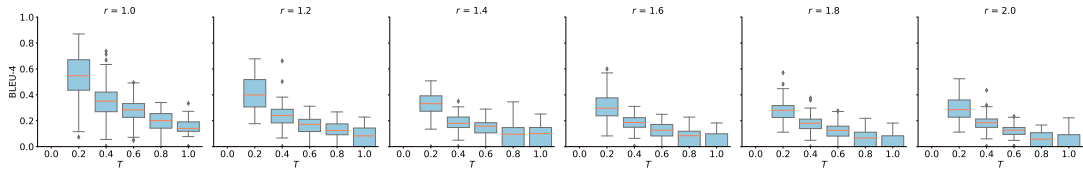
(e) the endings of the generated texts for



(f) the sampling loop size (up to 5 tokens) for the temperature  $T$



(g) the number of generated tokens for the temperature  $T$



(h) BLEU-4 scores for the temperature  $T$

Figure 5: Automatic metrics for the generation HP search for the “forum” category

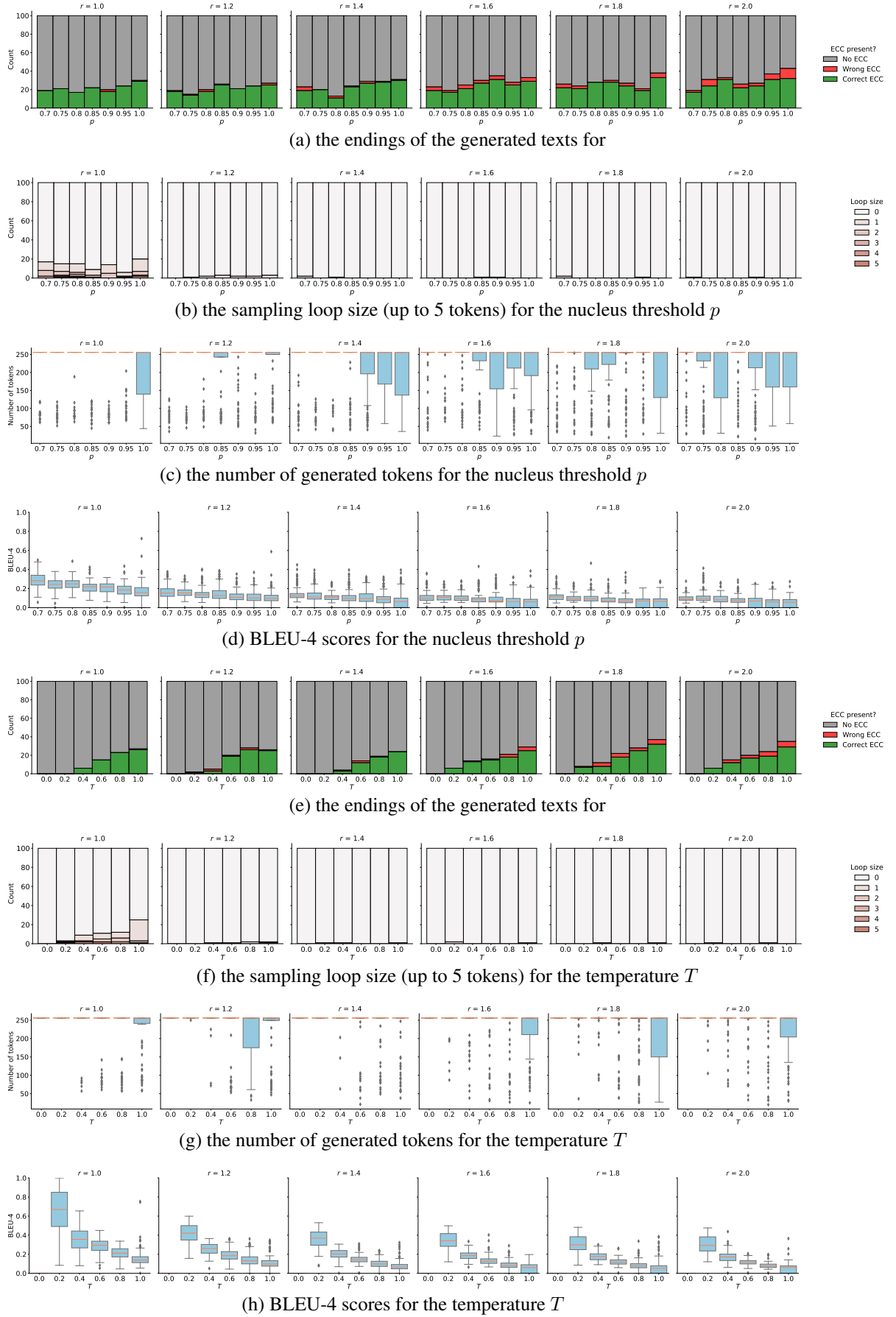


Figure 6: Automatic metrics for the generation HP search for the “info” category

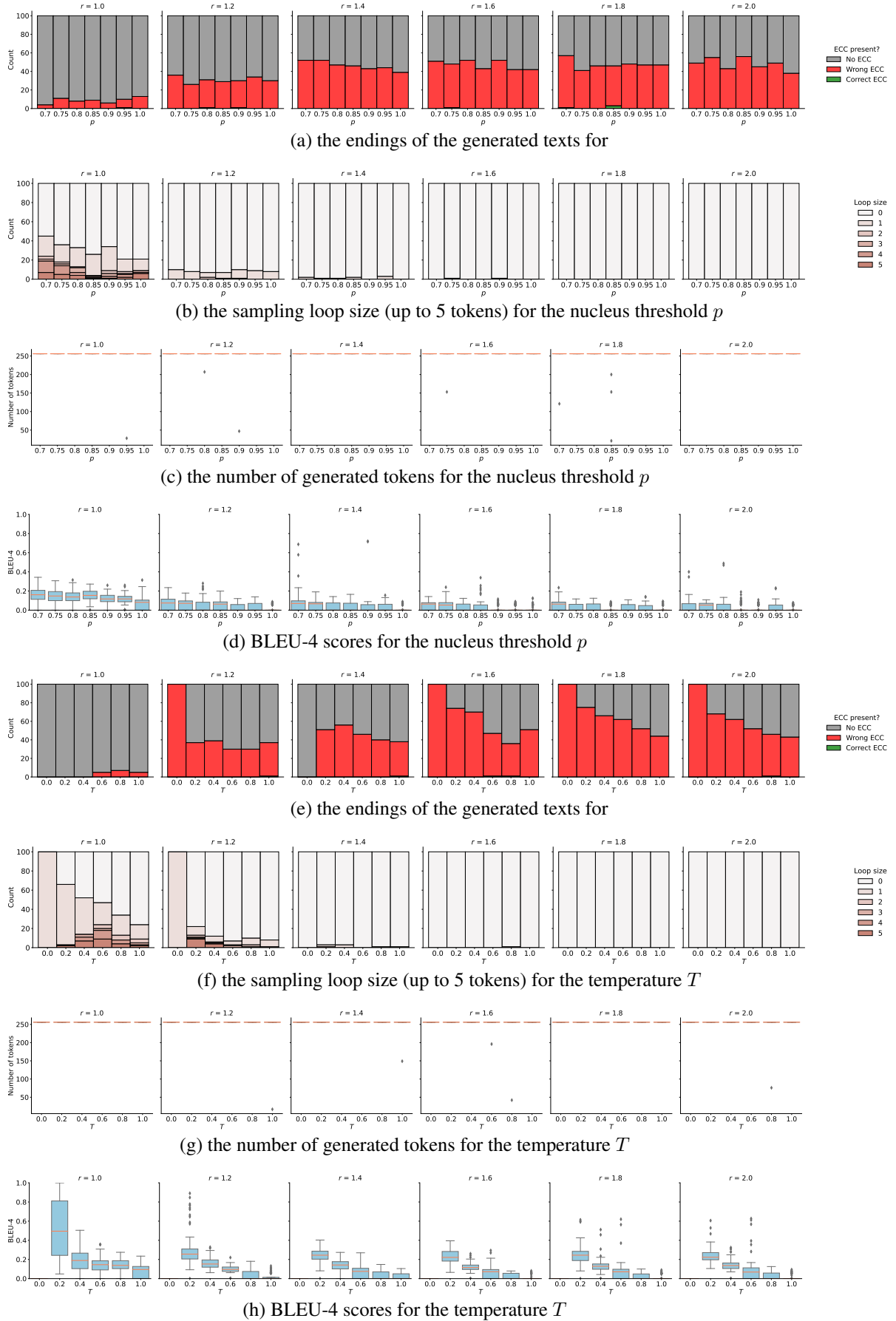
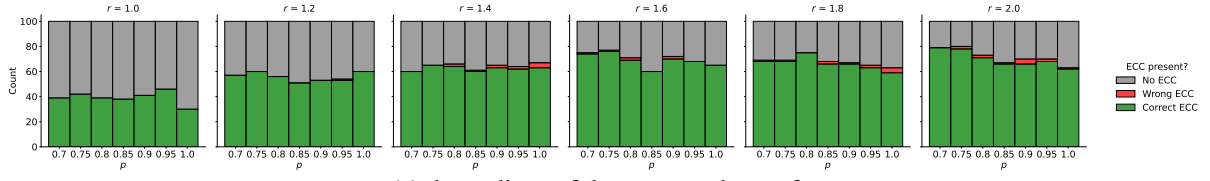
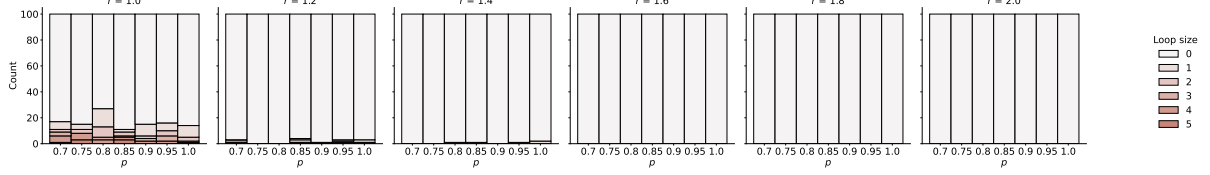


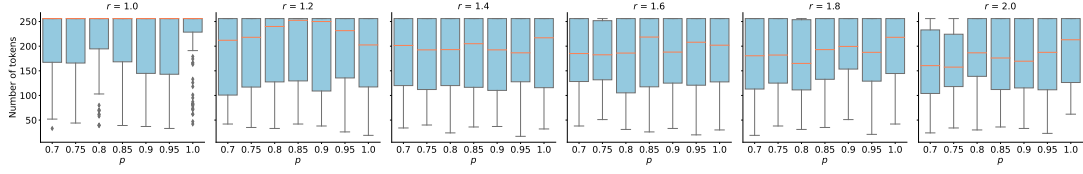
Figure 7: Automatic metrics for the generation HP search for the “literature” category



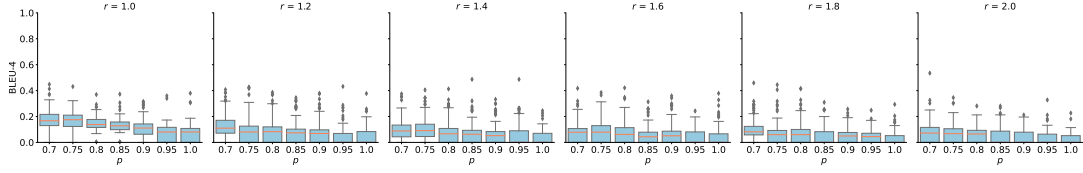
(a) the endings of the generated texts for



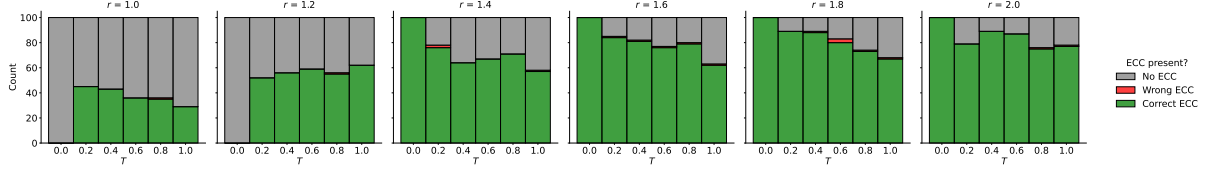
(b) the sampling loop size (up to 5 tokens) for the nucleus threshold  $p$



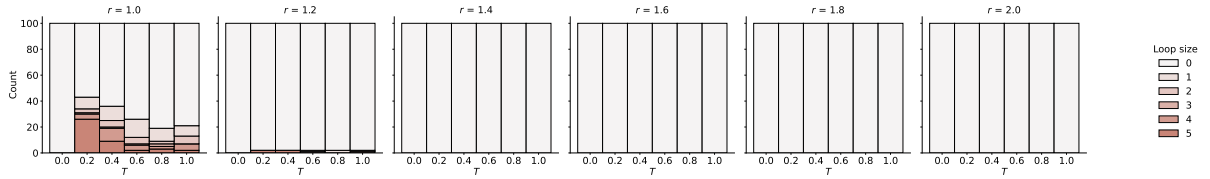
(c) the number of generated tokens for the nucleus threshold  $p$



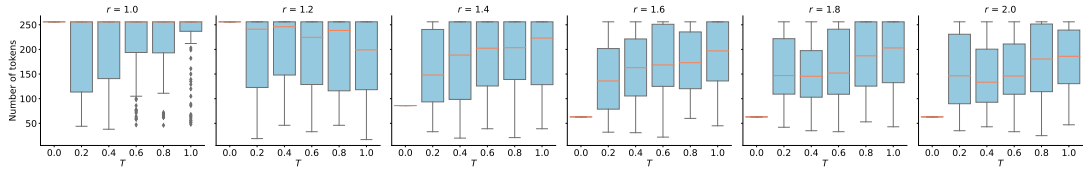
(d) BLEU-4 scores for the nucleus threshold  $p$



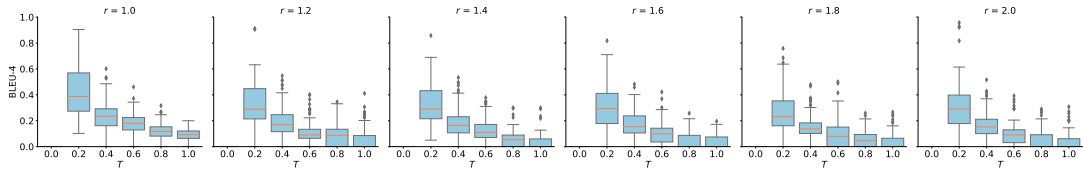
(e) the endings of the generated texts for



(f) the sampling loop size (up to 5 tokens) for the temperature  $T$

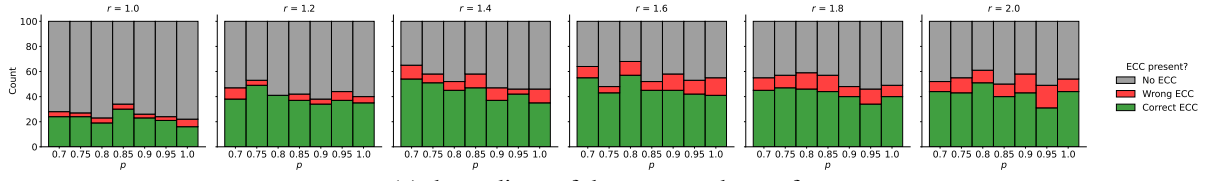


(g) the number of generated tokens for the temperature  $T$

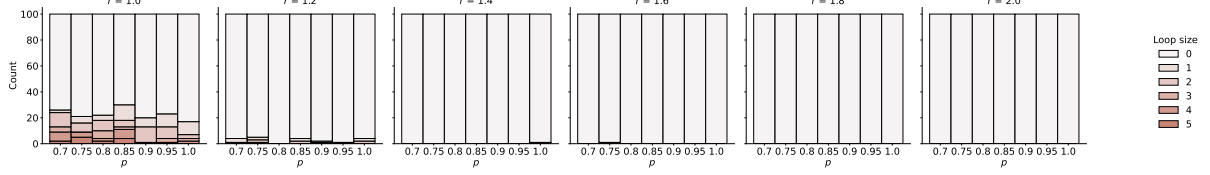


(h) BLEU-4 scores for the temperature  $T$

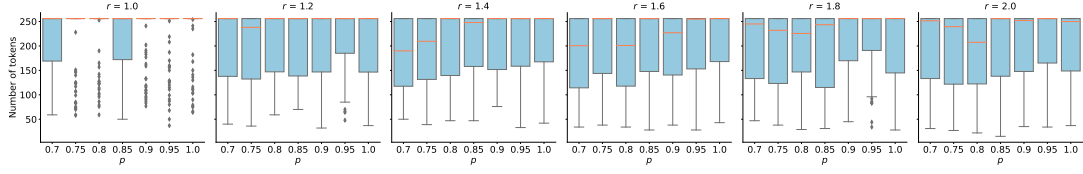
Figure 8: Automatic metrics for the generation HP search for the “news” category



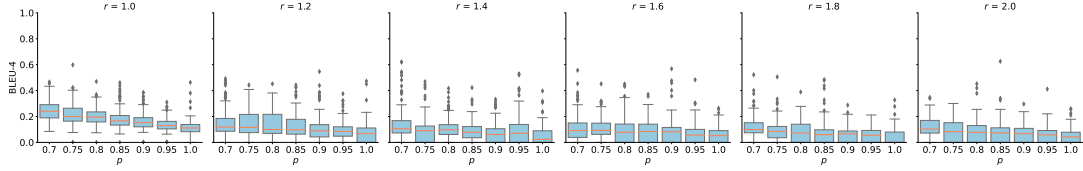
(a) the endings of the generated texts for



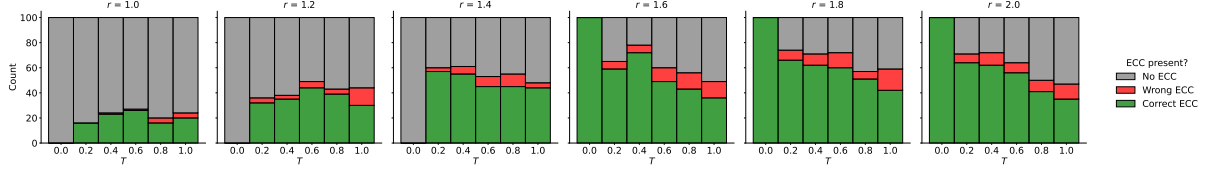
(b) the sampling loop size (up to 5 tokens) for the nucleus threshold  $p$



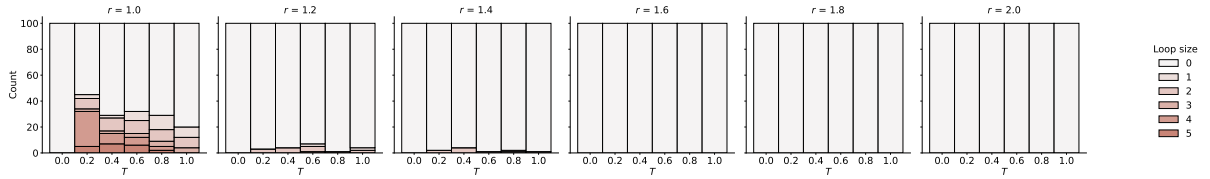
(c) the number of generated tokens for the nucleus threshold  $p$



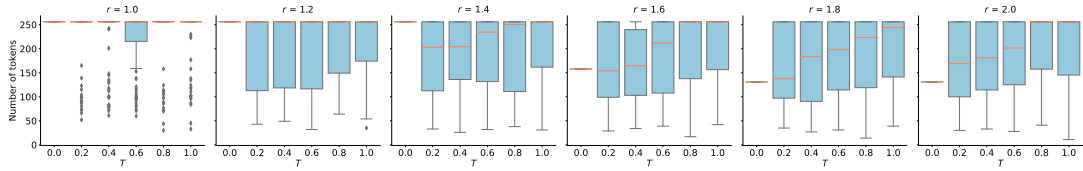
(d) BLEU-4 scores for the nucleus threshold  $p$



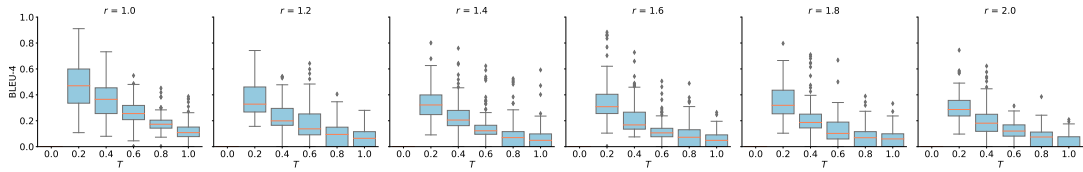
(e) the endings of the generated texts for



(f) the sampling loop size (up to 5 tokens) for the temperature  $T$



(g) the number of generated tokens for the temperature  $T$



(h) BLEU-4 scores for the temperature  $T$

Figure 9: Automatic metrics for the generation HP search for the “review” category

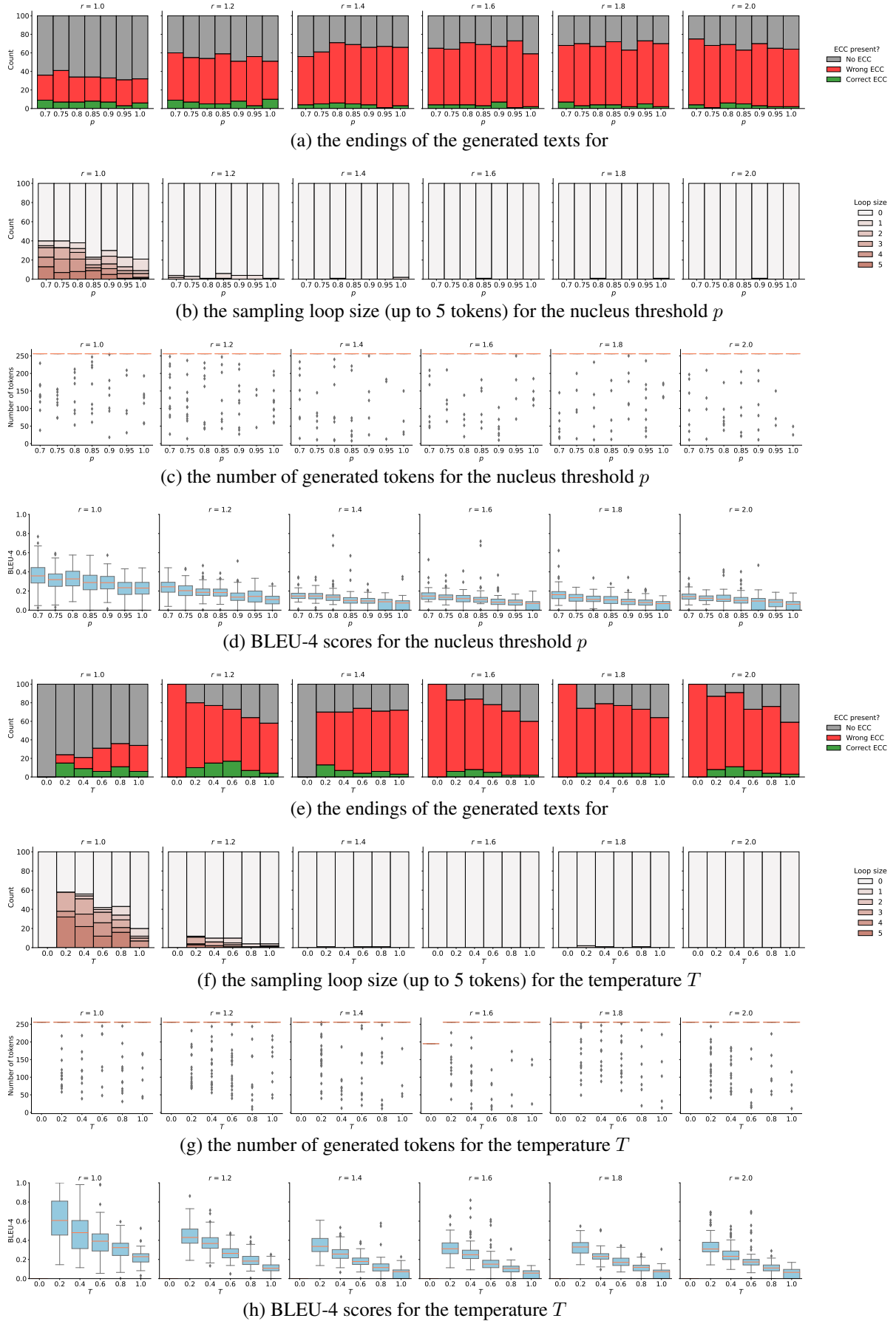


Figure 10: Automatic metrics for the generation HP search for the “simple” category





Figure 11: Automatic metrics for the generation HP search for the “wiki” category

Tack att du hjälper oss! Du kommer att få se ett antal texter (en i taget). Författaren till texten har blivit instruerad att utföra en specifik uppgift, till exempel:

*Skriv ett blogginlägg som börjar med orden "Stockholm är bäst på"*

Alla uppgifter var av samma typ: skribenten skulle skriva en text i en specifik genre (ett blogginlägg i exemplet ovan) som börjar med en specifik startfras ("Stockholm är bäst på" i exemplet ovan).

**DIN UPPGIFT** är att granska texterna och hitta följande feltyper (om de finns):

1. **Stilfel**, när skribenten har skrivit i en annan genre än den som efterfrågades, till exempel, en Wikipedia-artikel i stället för ett blogginlägg
2. **Ämnesglidning**, när merparten av texten är orelaterat till startfrasen.
3. **Grammatiska fel**, när det finns stavfel ("Sockholm är Sveriges huvudstad"), fel ordföljd ("Har gjort du dina läxor?"), eller fel böjning, till exempel genus ("en röd hus"), tempus ("Jag kommer att gå på bio för två dagar sedan"), numerus ("två stol"), osv.
4. **Fel ordval**, när skribenten har använt ett ord/en fras på ett felaktigt sätt (till exempel, "Jorden hoppar runt solen", eller "Att klappa med armarna är så kul!").
5. **Faktafel**, när skribenten har gjort ett påstående som inte stämmer, till exempel, "1 kilogram är lika med 30 gram" eller "Solen roterar runt jorden", eller "Hundar flyger vanligtvis väldigt snabbt".

Om du hittar **fel av typen 1 eller 2** (alltså "Stilfel" eller "Ämnesglidning"), kryssa i lämplig ruta på panelen "Markörer" till höger.

Om du hittar **fel av typen 3, 4 eller 5**, använd musen för att markera det stället i texten där du har hittat felet, och sedan trycka på en motsvarande markör på panelen "Markörer" till höger. Du kan även använda tangentbordsgenvägar ("g" för grammatiska fel, "o" för ordförrädsfel, och "f" för faktafel).

Texterna kan ibland sluta abrupt, detta ska **INTE** ses som ett fel.

Om du hittar andra feltyper, beskriv dem kortfattat i textrutan "Andra kommentarer" på panelen "Markörer" till höger.

Du behöver **INTE** hitta alla fel av alla typer! Om du hittar mer än 1 fel av någon typ, till exempel grammatiska fel, så räcker det att markera det första felet. Däremot om du har hittat 1 ordförrädsfel och 1 faktafel, ska du markera båda två eftersom de tillhör olika typer.

Om du inte kan hitta några fel, ska du trycka på rutan "Helt rätt" på panelen "Markörer" till höger.

När du känner dig klar med texten, tryck på knappen "Få en ny text" i den högre nedre hörnet.

Figure 12: The original human evaluation guidelines in Swedish