Prompting for Numerical Sequences: A Case Study on Market Comment Generation

Masayuki Kawarada, Tatsuya Ishigaki and Hiroya Takamura
Artificial Intelligence Research Center, AIST
masayuki.kawarada.m@gmail.com
{ishigaki.tatsuya, takamura.hiroya}@aist.go.jp

Abstract

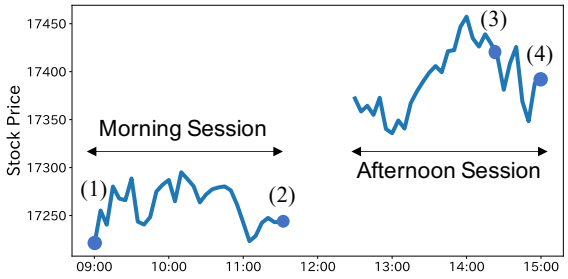
Large language models (LLMs) have been applied to a wide range of data-to-text generation tasks, including tables, graphs, and time-series numerical data-to-text settings. While research on generating prompts for structured data such as tables and graphs is gaining momentum, in-depth investigations into prompting for time-series numerical data are lacking. Therefore, this study explores various input representations, including sequences of tokens and structured formats such as HTML, LaTeX, and Python-style codes. In our experiments, we focus on the task of Market Comment Generation, which involves taking a numerical sequence of stock prices as input and generating a corresponding market comment. Contrary to our expectations, the results show that prompts resembling programming languages yield better outcomes, whereas those similar to natural languages and longer formats, such as HTML and LaTeX, are less effective. Our findings offer insights into creating effective prompts for tasks that generate text from numerical sequences.

Keywords: Generation, Data-to-text, Large language model

1. Introduction

Large language models (LLMs) have demonstrated remarkable performance in various text-to-text natural language generation tasks, such as text summarization (Wang et al., 2023b; Zhang et al., 2023; Pham et al., 2023), machine translation (Wang et al., 2023a; Karpinska and Iyyer, 2023; Vilar et al., 2023; Agrawal et al., 2023), and dialogue systems (Li et al., 2023; Jin et al., 2023). Although LLMs have also been applied to data-to-text generation tasks, their application has been limited to tasks where the input data are structured and their components are represented as words, such as tables (Saha et al., 2023; Zhao et al., 2023) and structured data (Jiang et al., 2023). However, the question of how to effectively use LLMs for text generation from numerical data remains unanswered. This paper addresses this question through a case study on few-shot market comment generation from stock prices (Murakami et al., 2017; Aoki et al., 2018; Hamazono et al., 2021). In this task, the stock price is provided as a numerical sequence, and the goal is to generate a market commentary at a specific point in time. Figure 1 illustrates an example of the market comment generation.

The key challenge in this study is to explore effective ways to represent these numerical sequences as prompts for LLMs, because their characteristics differ significantly from the text used to pretrain the models. We hypothesize that better performance could be achieved by creating prompts with characteristics similar to those used in the pretraining of LLMs. To test this hypothesis, we compare various prompt designs with features that are both similar to and different from the texts used for pretraining.



	Time	Market Comment
(1)	09:00	Nikkei 225 begins to rebound, opening at 17,193 yen, up 263 yen.
(2)	11:30	Nikkei 225 rebounded, closing of the morning session at 17,243 yen, an increase of 312 yen.
(3)	14:18	Tokyo Stock Exchange, 2:00 p.m., up more than 500 yen, buying continues in a wide range of stocks.
(4)	15:00	Nikkei 225 rebounded for the first time in 3 days, closing at 17,388 yen, up 457 yen.

Figure 1: Example of market comment generation. We represent the time-series numerical data using a line graph. The objective of this task is to generate comments on the stock price movement at a specific time.

Specifically, we explore four categories of prompt designs for numerical sequences: 1) directly providing a sequence of numbers; 2) treating the input numerical sequence as a structured table and linearizing it; 3) representing the input sequence as programming code (e.g., Python List, Python Dictionary, HTML, and LaTeX formats), which is a common strategy in code generation research; and

LREC-COLING 2024, pages 13190–13200

4) using templates to transform the input sequence into natural language.

Contrary to our expectations, our experiments revealed that prompts using expressions closer to programming languages performed better, whereas those closer to natural languages performed worse. In addition, longer prompts, such as HTML and LaTeX formats, were found to be less effective. These findings suggest that converting numerical data into prompts that closely resemble the format used during pretraining can lead to improved performance.

The main contributions of this study are threefold: 1) we present the performance of zero-shot and few-shot approaches for generating market commentary; 2) we compare various prompting strategies; and 3) we provide insights into the effectiveness of different prompt designs for numerical data.

2. Related Work

Existing data-to-text generation settings can be categorized based on the input and output types. The input types include tables (Puduppully et al., 2019; Lebret et al., 2016), graphs (Bai et al., 2022; Konstas et al., 2017), RDF triples (Gardent et al., 2017), and time-series numerical sequence (Murakami et al., 2017; Chang et al., 2022; Ishigaki et al., 2021; Kantharaj et al., 2022). From the output types, existing studies generate e.g., bibliographies (Lebret et al., 2016), summaries of sporting events (Puduppully et al., 2019), and commentaries (Ishigaki et al., 2021; Zhang et al., 2022; Chang et al., 2022). Despite the large amount of time-series numerical data in the real world, many existing settings deal with tabular and graphical data owing to public benchmark datasets such as WebNLG, E2E, and RotoWire. Our study aims to accelerate the research on time-series numerical data.

Various tasks have been proposed as time-series numerical-data-to-text settings, such as commentary generation from user gameplay (Ishigaki et al., 2021), explaining a drone's movement from sensor data (Chang et al., 2022) and explaining price changes in a market (Murakami et al., 2017; Hamazono et al., 2021; Aoki et al., 2018).

Two primary model architectures prevail in data-to-text settings: the encoder-decoder and the decoder-only. A conventional approach is fine-tuning pretrained encoder-decoder models, for example, BART (Lewis et al., 2020) and T5 (Raffel et al., 2020). Recently, interest has shifted to zero-shot or few-shot generation using decoder-only models such as GPTs (Brown et al., 2020). Several prompting methods have been proposed for the table-to-text and graph-to-text settings (Axelsson and Skantze, 2023; Lorandi and Belz, 2023); however, none have been proposed for the time-series numerical-data-to-text setting.

Time-series numerical data can be converted into tables by aligning numerical values and timestamps. In this case, the linearization method proposed in the existing table-to-text research can be applicable. In addition, expressions in programming languages such as Python have been used in research on automatic code generation. In this study, we propose using time-series numerical sequences as such representations.

3. Task

This section describes the task settings. Figure 1 presents an example of an input and its corresponding output market comments. We use two time-series numerical sequences derived from the Nikkei Stock Average (Nikkei225): 1) a long-term series that records daily closing prices over the last seven days, and 2) a short-term series that captures daily price fluctuations at five min intervals from market opening to closing. The system generates a market comment on the target timestamp based on these two inputs.

4. Methods

This section compares the prompting methods. We assume that each number in sequences can be aligned to a timestamp. In the example shown in Table 4, a value 9988.05 yen in an input time-series numerical sequence can be aligned to, for example, the timestamp of 15:00, at which the value was tracked.

Table 3 lists the prompt templates we use for zero- and few-shot generations. In practice, we replace `[INPUT FORMAT(short-term)]` and `[INPUT FORMAT(long-term)]` with the respective input formats in Tables 1 and 2. This prompt contains two placeholders, i.e., `[INPUT FORMAT] (short-term)` and `[INPUT FORMAT] (long-term)`, which are replaced by one of the nine prompts listed in Tables 1 and 2. These prompts are divided into four categories: 1) direct prompts; 2) converting the sequence into a table and then linearizing it; 3) converting the sequence into expressions used for computer programs; and 4) filling a template that produces human-like language.

4.1. Direct Prompt (Baseline)

This method simply combines adjacent numerical values with a space to represent a time-series numerical sequence as a sequence of space-separated numerical values.

Direct	...9988.05 9982.06 9978.11 9972.66 9967.11, 9961.37 ...
Column	Time:15:00 14:55 14:50 14:45 14:40 14:35 ... Nikkei225:9988.05 9982.06 9978.11 9972.66 9967.11, 9961.37 ...
Row	Time Nikkei225\n15:00 9988.05\n14:55 9982.06\n14:50 9978.11\n14:45 9972.66\n14:40 9967.11\n14:35 9961.37 ...
Python List	Time = [..., "15:00", "14:55", "14:50", "14:45", "14:40", "14:35", ...] Nikkei225 = [..., 9988.05, 9982.06, 9978.11, 9972.66, 9967.11, 9961.37, ...]
Python List (nested)	Nikkei225 = [..., [15:00, 9988.05], [14:55, 9982.06], [14:50, 9978.11], 14:45, 9972.66], [14:40,9967.11], [14:35,9961.37], ...]
Python Dictionary	Nikkei225 = {..., "15:00":9988.05, "14:55":9982.06, "14:50":9978.11, "14:45":9972.66, "14:40":9967.11, "14:35":9961.37, ...}
HTML Table	<table><tr><th>Time</th><th>Nikkei225</th></tr>...(omitted)<tr><td>15:00</td><td>9988.05</td></tr><tr><td>14:55</td><td>9982.06</td></tr><tr><td>14:50</td><td>9978.11</td></tr><tr><td>14:45</td><td>9972.66</td></tr><tr><td>14:40</td><td>9967.11</td></tr><tr><td>14:35</td><td>9961.37</td></tr>...</table>
LaTeX Table	\begin{table}[t] \begin{tabular} & \hline Timestamp & Nikkei225 \\\hline \hline 15:00 & 9988.05 \\\hline 14:55 & 9982.06 \\\hline 14:50 & 9978.11 \\\hline 14:45 & 9972.66 \\\hline 14:40 & 9967.11 \\\hline 14:35 & 9961.37 \\\hline ... \end{tabular} \end{table}
Text (English)	Nikkei225 as of 15:00 is 9982.06 yen.\nNikkei225 stock price as of 15:00 is 9988.05 yen.\nNikkei225 stock price as of 14:55 is 9982.06 yen.\nNikkei225 stock price as of 14:50 is 9978.11 yen.\nNikkei225 stock price as of 14:45 is 9972.66 yen.\nNikkei225 stock price as of 14:40 is 9967.11 yen.\nNikkei225 stock price as of 14:35 is 9961.37 yen....(omitted)
Text (Japanese)	Nikkei225 as of 15:00 is 9982.06円.\n15:00時 点 のNikkei225は9988.05円.\n14:55時 点 のNikkei225は9982.06円.\n14:50時 点 のNikkei225は9978.11円.\n14:45時 点 のNikkei225は9972.66円.\n14:40時 点 のNikkei225は9967.11円.\n14:35時 点 のNikkei225は9961.37円....(omitted)

Table 1: Examples of `[INPUT FORMAT (short-term)]`. The actual prompts are written in Japanese except for Text (Japanese).

4.2. Linearized Table

A direct prompt does not use the information about the alignment between a value and its timestamp. Thus, we promptly included richer information by adding more information about the alignments. We convert a time-series numerical sequence into a table as shown in Table 4. Each value in the input sequence is aligned with the timestamp at which the market price is recorded. The converted table has two rows: the first row contains a sequence of timestamps, and the second row represents a sequence of market prices. Each row has the headings "Time" and the name of the market index, i.e., "Nikkei225", respectively.

Next, we apply linearization inspired by existing table-to-text studies. In this category, we compare two linearization methods: `Column` and `Row` as explained below:

Column This method extracts the heading and values for each column and concatenates them by adding spaces. This process is performed for two columns, and the two token sequences are combined by placing a space to obtain the final column.

Row As with the `Column` method, a sequence of space-delimited tokens is obtained for each column, and the two series are joined by a breakline symbol ("\n"). This includes the information that the two token sequences are generated from different columns.

4.3. Programming Language-like Prompts

We compare the idea of adopting the representation methods used in programming languages, such as Python List and Dictionary. LLMs are also pretrained on source code extracted from repositories such as GitHub. The inclusion of source-code in a prompt is common, especially in source code generation tasks, in which high performance in both understanding and generation has been reported. Therefore, it is beneficial to convert an input time-series numerical sequence into a form similar to that of a programming language.

Python List This method first creates two python-like lists named "*Time*" and "*Nikkei225*". The former list contains the timestamps as string values, e.g., *["15:00", "14:55", "14:40", ... "14:35"]*. The latter list contains the numerical stock prices as floating values *[9988.05, 9982.06, ..., 9961.37]*. Finally, these two lists are concatenated by a space.

Direct	9988.05 9982.06 9978.11 9972.66 9967.11, 9961.37 9960.20
Column	Date: 7DaysAgo 6DaysAgo 5DaysAgo 4DaysAgo 3DaysAgo 2DaysAgo 1DayAgo Nikkei225: 9988.05 9982.06 9978.11 9972.66 9967.11, 9961.37 9960.20
Row	Date Nikkei225\n7DaysAgo 9988.05\n6DaysAgo 9982.06\n5DaysAgo 9978.11\n4DaysAgo 9972.66\n3DaysAgo 9967.11\n2DaysAgo 9961.37 \n1DayAgo
Python List	Time = ["7DaysAgo", "6DaysAgo", "5DaysAgo", "4DaysAgo", "3DaysAgo", "2DaysAgo", "1DayAgo"] Nikkei225 = [9988.05, 9982.06, 9978.11, 9972.66, 9967.11, 9961.37, 9960.20]
Python List (nested)	Nikkei225 = [["7DaysAgo", 9988.05], ["6DaysAgo" 9982.06], ["5DaysAgo", 9978.11], ["4DaysAgo", 9972.66], ["3DaysAgo",9967.11], ["2DaysAgo",9961.37], ...]
Python Dictionary	Nikkei225 = {"7DaysAgo":9988.05, "6DaysAgo":9982.06, "5DaysAgo":9978.11, "4DaysAgo":9972.66, "3DaysAgo":9967.11, "2DaysAgo":9961.37, ... }
HTML	<table><tr><th>Date</th><th>Nikkei225</th></tr><tr><td>7DaysAgo</td><td>9988.05</td></tr><tr><td>6DaysAgo</td><td>9982.06</td></tr><tr><td>5DaysAgo</td><td>9978.11</td></tr><tr><td>4DaysAgo</td><td>9972.66</td></tr><tr><td>3DaysAgo</td><td>9967.11</td></tr><tr><td>2DaysAgo</td><td>9961.37</td>...</tr></table>
LaTeX	\begin{table}[t] \begin{tabular} & \hline Timestamp & Nikkei225 \\\hline \hline 7DaysAgo & 9988.05 \\\hline 6DaysAgo & 9982.06 \\\hline 5DaysAgo & 9978.11 \\\hline 4DaysAgo & 9972.66 \\\hline 3DaysAgo & 9967.11 \\\hline 2DaysAgo & 9961.37 \\\hline ... \end{tabular} \end{table}
Text (English)	Nikkei225 as of 7 days ago was 9982.06 yen.\nNikkei225 closing stock price as of 6 days ago was 9988.05 yen.\nNikkei225 closing stock price as of 5 days ago was 9982.06 yen.\nNikkei225 closing stock price as of 4 days ago was 9978.11 yen.\nNikkei225 closing stock price as of 3 days ago was 9972.66 yen.\nNikkei225 closing stock price as of 2 days ago was 9967.11 yen.\nNikkei225 closing stock price as of yesterday was 9961.37 yen. ...
Text (Japanese)	7日 前 のNikkei225終 値 は9982.06円.\n6日 前 のNikkei225終 値 は9988.05円.\n5日 前 のNikkei225終値は9982.06円.\n4日前のNikkei225終値は9978.11.\n3日前のNikkei225終値 は9972.66円.\n2日 前 のNikkei225終 値 は9967.11円.\n2日 前 のNikkei225終 値 は9961.37円... ...

Table 2: Examples of `[INPUT FORMAT (long-term)]`. The actual prompts are written in Japanese except for Text (English).

Output the market comment at the current time in the form of a <comment>market comment</comment>.
###
Input:
`[INPUT FORMAT (short-term)]`
`[INPUT FORMAT (long-term)]`
Output:
Nikkei225 closes at large, rebounding yen strength pushes mainstay stocks higher
###
Input:
`[INPUT FORMAT (short-term)]`
`[INPUT FORMAT (long-term)]`
Output:

Table 3: The template we use for the few-shot setting.

Python List (nested) This prompting method first creates a python-like list with two elements for each timestamp and price pair, e.g., *[15:00, 9988.05]* means that the price is 9988.05 at 15:00. This method iteratively adds the created list into another python list *Nikkei225*, as listed in Table 1.

Timestamp	Nikkei225
15:00	9988.05
14:55	9982.06
14:50	9978.11
14:45	9972.66
14:40	9967.11
14:35	9961.37

Table 4: Example of a table converted from a numerical sequence of market prices.

Python Dictionary This method converts a numerical sequence into a Python Dictionary format with key-value pairs, where each key represents the timestamp and each value corresponds to a stock price e.g., {"15:00": 9988.05, "14:55": 9982.06, ...}.

HTML As with the `Column` method, `HTML` method assumes that the time-series numerical sequence is represented by a two-row table.

	0-shot			5-shot			10-shot		
	BLEU	METEOR	BERTScore	BLEU	METEOR	BERTScore	BLEU	METEOR	BERTScore
Direct	0.01	0.48	60.30	8.26	25.22	73.50	9.39	26.55	73.96
Column	0.38	14.06	65.33	8.30	24.99	73.35	9.49	26.00	73.65
Row	0.42	8.86	64.83	9.16	26.33	73.76	10.49	27.88	74.31
Python List	0.36	16.16	65.01	8.32	25.32	73.54	9.59	26.51	73.87
Python List (nested)	0.40	8.94	65.77	9.15	26.77	74.01	9.86	27.42	74.15
Python Dictionary	0.44	9.60	65.40	9.17	26.42	73.96	10.41	28.25	74.56
HTML Table	0.35	12.26	63.93	8.30	26.10	73.92	8.45	26.44	74.08
LaTeX Table	0.44	15.11	67.59	8.36	26.10	73.76	9.53	27.56	74.02
Text (English)	0.30	15.66	60.13	8.49	25.92	73.97	9.10	26.95	74.52
Text (Japanese)	0.03	0.95	55.35	8.51	26.55	74.00	9.26	27.60	74.21

Table 5: Comparison of methods in terms of BLEU, METEOR, and BERTScore.

	BLEU	METEOR	BERTScore
EncDec	11.41	30.90	75.94
Direct	9.39	26.55	73.96
Column	9.49	26.00	73.65
Row	10.49	27.88	74.31
Python List	9.59	26.51	73.87
Python List (nested)	9.86	27.42	74.15
Python Dictionary	10.41	28.25	74.56
HTML Table	8.45	26.44	74.08
LaTeX Table	9.53	27.56	74.02
Text (English)	9.10	26.95	74.52
Text (Japanese)	9.26	27.60	74.21

Table 6: Comparison of prompts using 10-shot and EncDec in terms of BLEU, METEOR, and BERTScore.

This method represents a table as an HTML code, as shown in Table 1.

LaTeX As another format for representing a table, we also compare the Latex format.

4.4. Language Template-based Prompt

LLMs are trained mainly on natural language text on the Web. We hypothesize that prompts resembling natural language work well. Thus, we propose a template-based method for converting a time-series numerical sequence into a natural language-like-prompt. Specifically, we use two templates: `Text (English)` and `Text (Japanese)`, as explained below:

Text (Japanese) This setup uses the template *"AAA時点での日経平均はBBB。"*, which means *"Nikkei225 as of AAA is BBB yen"*. In this template, AAA refers to the timestamp and BBB is the corresponding market price. For each time-price pair, we obtain a sentence by filling the slots. All obtained sentences are finally combined by a breakline symbol ("\n").

Text (English) For many released LLMs, a large portion of the training data for pretraining is in English. Thus, we also compare the English version of the prompt, i.e., *"Nikkei225 as of AAA is BBB yen"*.

5. Experiments

Here, we describe our experiments; dataset, comparison methods, and evaluations.

5.1. Dataset

We use the dataset in Murakami et al. (2017); Aoki et al. (2018); Hamazono et al. (2021). The dataset contains 18,489 pairs of time-series numerical sequences and market comments. The time-series numerical sequences are obtained from IBISquare[1]. Stock price data are collected from December 2010 to September 2016. The comments are provided by Nikkei Quick News[2]. We split this dataset into training, validation, and testing sets, comprising 15,035, 1,759, and 1,695 data points, respectively. For this study, we define "short-term sequence" as the sequence of stockvalues recorded every five minutes and "long-term sequence" as the closing prices for the stock over the last seven days. The Nikkei market operates in two sessions: 1) a morning session from 9:00 to 11:30, and 2) an afternoon session from 12:30 to 15:00. Consequently, the short-term sequence comprised a maximum of 62 prices per day, whereas the long-term sequence comprises seven days.

5.2. Compared Methods

We compare our zero- and few-shot models with an existing fine-tuned encoder-decoder model (Murakami et al., 2017).

Existing Encoder-decoder Model (EncDec)

We use an extended implementation of an existing model by Murakami et al. (2017). The original implementation directly encodes long- and short-term time-series numerical sequences using multilayer perceptrons (MLPs), and then, the encoded vectors

[1] http://www.ibi-square.jp/index.html

[2] "We have released the source code for preprocessing at https://github.com/aistairc/market-reporter.

are fed into an LSTM-based encoder-decoder. Our extended implementation uses a pretrained BART [3] for the encoder-decoder, which has been used more frequently in recent data-to-text settings. Our implementation uses multilayer perceptrons (MLPs) to convert both short-term and long-term input vectors into fixed-size vectors of size 768. This size is selected because the embedding layer of the BART is 768. Finally, BART then receives the two vectors and outputs the comments. This model is fine-tuned on the abovementioned dataset to minimize the cross-entropy loss.

Proposed Zero- and Few-shot Models

We use an instruct-tuned GPT (Brown et al., 2020)[4]. In the zero-shot setting, the prompt shown in Table 3 is used as is. In the few-shot setting, we append a reference comment to the prompt after *Output:* in the prompt. We compare up to 10 shots in our experiments. We set the maximum number to 10 owing to the length limits of the GPT. In preliminary experiments, we found that the performance varied significantly depending on the instances used for the shots. Therefore, we randomly select the instances for shots, repeat the experiments 10 times, and report the averaged scores.

5.3. Automatic and Human Evaluations

In this subsection, we describe the evaluation.

5.3.1. Metrics for Automatic Evaluation

For the scores, we adopt the commonly used BLEU and F1-score from the BERTScore. BLEU has been employed in many existing studies; however, further evaluation is required because it considers only surface words. Therefore, BERTScore, which uses neural network embedding to capture semantic similarities, is employed.

Such automatic evaluation metrics only capture the similarity between automatically generated and reference comments but do not capture the correctness of the generated text.

5.3.2. Evaluation by Human Judges

Therefore, we also evaluate the generated comments by human judges. For each comparison method, we present 30 comments of time-series numerical data to a human evaluator, who is a native Japanese speaker, and ask the evaluator to evaluate whether the output text is consistent or inconsistent with the reference. The evaluator can also refer to the short-term and long-term sequences, if needed. We report the number of consistent and inconsistent results for each method.

[3] https://huggingface.co/stockmark/bart-base-japanese-news

[4] In particular, we use *gpt-3.5-turbo* in OpenAI's API.

6. Results and Discussions

The results are presented in the following section.

6.1. Automatic Evaluation

Figure 2 shows the changes in BLEU scores with the number of shots. The detailed scores of BLEU, METEOR, and BERTScore are shown in Table 6. Overall, the scores generally increase with the number of shots, except for the `HTML` prompt, as shown in the bottom-left graph in Figure 2. All methods on the zero-shot setting achieve very low scores in terms of BLEU, METEOR, and BERTScore, implying that the multitasking capability of the LLM is not sufficient for the market comment generation. `Direct` prompt does not work well for this task, and all other prompts outperform it.

A further look at the scores for prompts other than the `Direct` prompt yielded three findings: 1) `Python Dictionary`, `Python List (nested)`, and `Row` work better than the other prompts, 2) `Text` unexpectedly works worse; and 3) among programming language-based prompts, `HTML` performs the worst. Each finding is explained in detail below.

Finding 1: `Python Dictionary`, `Python List (nested)`, and `Row` perform better

Table 6 lists the BLEU, METEOR, and BERTScores for each prompt in all categories. Owing to the space limitation, we show the values obtained in the setting of 0, 5, and 10 shots. Overall, we observe that there are three best-performing models among all models: `Python Dictionary`, `Python List (nested)`, and `Row`. These best-performing prompts share common characteristics: 1) they have similar characteristics to the data used for pretraining, and 2) the timestamps and market prices are written closely to each other in a prompt. Python codes are also used for the pretraining of the LLMs, thus, better results may be reasonable. `Row` also achieves better results because this approach is used for dataset creation for pretraining to linearize HTML tables on the web.

Furthermore, the timestamps and market prices are written closely to each other, for example, *[15:00, 9988.05]* in the `Python List (nested)` and *15:00":9988.05* in the `Python Dictionary`. Other programming language-like prompts, such as `Python List`, `HTML`, and `LaTeX`, with lower performance write the timestamps and the market prices far away from each other. One possible reason for this is that GPT

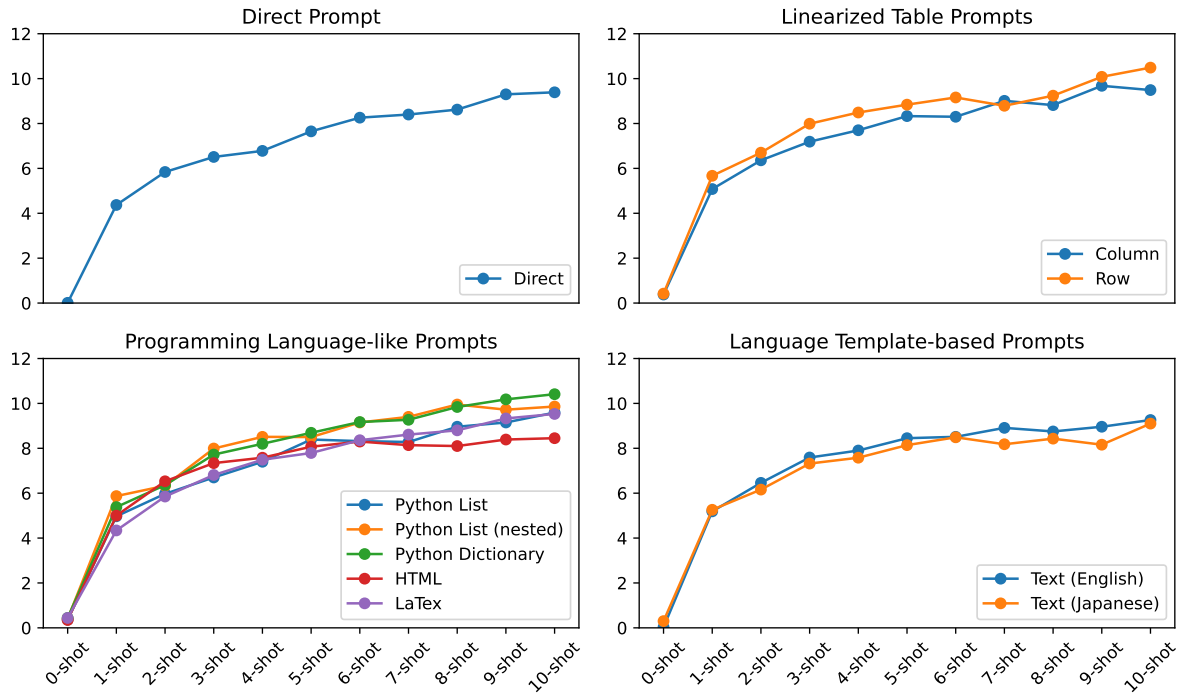


Figure 2: BLEU scores of different prompts on different numbers of shots.

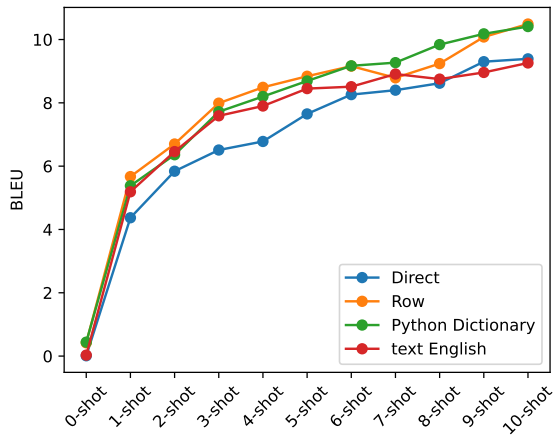


Figure 3: BLEU scores of different prompts on different numbers of shots.

was able to accurately capture the correspondence between the prices and a time.

Finding 2: `Text` unexpectedly performs worse

We hypothesized that `Text (English)` and `Text (Japanese)` would perform better; however, Table 3 reveals that these methods perform worse than the best-performing programming language-like prompts, that is, `Python Dictionary` and `Python List (nested)`. Specifically, the performances of `Text (English)`, represented by a red line, are lower than those of other prompts, that is, `Row` and `Python Dictionary`, which are defined by orange and green lines, respectively, except for 7-shot setting for the `Row` prompt.

One possible reason for this lower performance is that the template generates repetition of the same expression. For example, the English template generates "Nikkei225 as of 15:00 is 9982.06 yen" and repeatedly generates similar expressions. Such text is unnatural and may not be included in the web texts used for pretraining.

Interestingly, the bottom left graph in Table 2 shows that `Text (English)` performs better than `Text (Japanese)` although the task is to generate a comment in Japanese. A possible reason could be that English texts are used more frequently during pretraining.

Finding 3: Among programming language-based prompts, `HTML` and `LaTeX` perform worse

The bottom left graph in Figure 2 shows each model's BLEU scores in the program language-like prompts category. Among the prompts in this category, `HTML` and `LaTeX` perform worse than other prompts. In particular, the performance of `HTML` has not improved after 5-shot and has already converged, whereas the performances of the other prompts are still improving.

Reference	日経平均、続伸 前引けは421円高の１万6349円 *(The Nikkei 225 extended its gains, closing up 421 yen at 16,349 at the midday break)*
EncDec	東証前引け、反落 米株安や円安一服で利益確定売りも *(Tokyo stocks fall at midday close, retreating on Wall Street declines and a pause in yen weakening, with profit-taking also in play.)*
Direct	日経平均、反落 前引けは80円安の１万6349円 *(The Nikkei 225 reversed course, falling 80 yen to close at 16,349 at the midday break.)*
Column	日経平均、反落 前引けは421円安の１万6349円 *(The Nikkei 225 reversed course, falling 421 yen to close at 16,349 at the midday break.)*
Row	日経平均、反発 前引けは421円高の１万6349円 *(The Nikkei 225 rebounded, closing up 421 yen at 16,349 at the midday break.)*
Python List	日経平均、反落 前引けは421円安の１万6349円 *(The Nikkei 225 reversed course, falling 421 yen to close at 16,349 at the midday break.)*
Python List (nested)	日経平均、大幅続落 前引けは421円安の１万6349円 *(The Nikkei 225 continued to drop sharply, falling 421 yen to close at 16,349 at the midday break.)*
Python Dictionary	日経平均、反落 前引けは421円安の１万6349円 *(The Nikkei 225 reversed course, falling 421 yen to close at 16,349 at the midday break.)*
HTML	日経平均、続伸 前引けは421円高の１万6349円 *(The Nikkei 225 extended its gains, closing up 421 yen at 16,349 at the midday break.)*
LaTeX	日経平均、続伸 前引けは200円超の１万6349円 *(The Nikkei 225 extended its gains, closing up 200 yen at 16,349 at the midday break.)*
Text (English)	日経平均、反落 前引けは121円安の１万6349円 *(The Nikkei 225 reversed course, falling 121 yen to close at 16,349 at the midday break.)*
Text (Japanese)	日経平均、反落 前引けは421円安の１万6349円 *(The Nikkei 225 reversed course, falling 421 yen to close at 16,349 at the midday break.)*

Table 7: Comparison of text output by each method, with 'movement terms' highlighted in red in the table.

	# Consistent	# Inconsistent
EncDec	7	23
Direct	8	22
Row	**14**	16
Python Dictionary	12	18
Text (English)	12	18

Table 8: Results of human evaluation on EncDec, Direct, Row, Python Dictionary, and Text English.

According to the statistics provided by GitHub[5], one of the pretraining sources, the proportion of HTML formatted files is less than 0.1%. One possible reason may be the small percentage of data used for the pretraining. Furthermore, the `HTML` prompt is inherently longer than the others, which makes reasoning using by the GPT difficult.

6.2. Evaluation by Human Judges

We conducted human evaluations of the encoder-decoder method and the method with the highest BLEU score for each category: `Direct`, `Row`, `Python Dictionary`, and `Text (English)`.

Table 8 presents the results of the human evaluation. Across all methods, we discovered that the evaluator judges the generated comments more frequently than comments consistent with the reference comments. This includes many cases in which the reference mentions that the stock price is rising, but the generated comment states that it is falling, or vice versa. Table 7 shows the output of each method. In this table, the 'movement term', referring to the stock price movement such as 'rising' or 'falling', is highlighted in red.

Interestingly, although the fine-tuned encoder-decoder method achieved high scores in the automatic evaluation, it received lower ratings in the human evaluation. By contrast, the prompt-based method yielded comments that were more consistent with the reference comments. We hypothesize that this discrepancy arises because the fine-tuned comments are stylistically similar to the references but do not accurately capture numerical movements. In contrast, the prompt-based method excels in understanding numerical trends but is limited in covering all text styles, given its reliance on at most ten examples as shots. To verify this hypothesis, a separate evaluation focusing on the accuracy of references to text style and numerical movements is necessary, which indicates a potential direction for future research.

[5] https://madnight.github.io/githut

7. Conclusion

In this study, we address the use of LLMs for text generation from numerical sequences through the generation of market comments. We approached the problem by converting numerical sequences into ten different formats across four categories, and conducted experiments to compare these methods with the existing encoder-decoder approach based on BART. Our results indicated that prompts based on programming languages yielded strong performance, whereas textual prompts and those based on HTML and LaTeX tables, were less effective. Furthermore, a human evaluation assessing the consistency with reference revealed that the prompt-based methods outperformed the encoder-decoder method. This highlights the potential of prompt-based approaches to generate market comment and similar data-to-text tasks.

Although our research focused on market comments as a representative task with numerical sequence inputs, many other tasks share comparable characteristics. In future work, it will be valuable to conduct experiments on datasets with other types of numerical sequence inputs to further validate and extend our findings.

Acknowledgements

This study is based on the results obtained from a project JPNP20006, commissioned by the New Energy and Industrial Technology Development Organization (NEDO). Ideas related to time-aware prompts are obtained from discussions in the BRIDGE (Program for Bridging the Gap Between R&D and Ideal Society (Society5.0) and Generating Economic and Social Value) supported by Cabinet Office. For experiments, the computational resource of the AI Bridging Cloud Infrastructure (ABCI) provided by the National Institute of Advanced Industrial Science and Technology (AIST) were used.

8. Bibliographical References

Sweta Agrawal, Chunting Zhou, Mike Lewis, Luke Zettlemoyer, and Marjan Ghazvininejad. 2023. In-context examples selection for machine translation. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 8857–8873, Toronto, Canada. Association for Computational Linguistics.

Tatsuya Aoki, Akira Miyazawa, Tatsuya Ishigaki, Keiichi Goshima, Kasumi Aoki, Ichiro Kobayashi, Hiroya Takamura, and Yusuke Miyao. 2018. Generating market comments referring to external resources. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 135–139, Tilburg University, The Netherlands. Association for Computational Linguistics.

Agnes Axelsson and Gabriel Skantze. 2023. Using large language models for zero-shot natural language generation from knowledge graphs. In *Proceedings of the Workshop on Multimodal, Multilingual Natural Language Generation and Multilingual WebNLG Challenge (MM-NLG 2023)*, pages 39–54, Prague, Czech Republic. Association for Computational Linguistics.

Xuefeng Bai, Yulong Chen, and Yue Zhang. 2022. Graph pre-training for AMR parsing and generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6001–6015, Dublin, Ireland. Association for Computational Linguistics.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Ernie Chang, Alisa Kovtunova, Stefan Borgwardt, Vera Demberg, Kathryn Chapman, and Hui-Syuan Yeh. 2022. Logic-guided message generation from raw real-time sensor data. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 6899–6908, Marseille, France. European Language Resources Association.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. The WebNLG challenge: Generating text from RDF data. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 124–133, Santiago de Compostela, Spain. Association for Computational Linguistics.

Yumi Hamazono, Tatsuya Ishigaki, Yusuke Miyao, Hiroya Takamura, and Ichiro Kobayashi. 2021.

Unpredictable attributes in market comment generation. In *Proceedings of the 35th Pacific Asia Conference on Language, Information and Computation*, pages 217–226, Shanghai, China. Association for Computational Lingustics.

Tatsuya Ishigaki, Goran Topic, Yumi Hamazono, Hiroshi Noji, Ichiro Kobayashi, Yusuke Miyao, and Hiroya Takamura. 2021. Generating racing game commentary from vision, language, and structured data. In *Proceedings of the 14th International Conference on Natural Language Generation*, pages 103–113, Aberdeen, Scotland, UK. Association for Computational Linguistics.

Jinhao Jiang, Kun Zhou, Zican Dong, Keming Ye, Xin Zhao, and Ji-Rong Wen. 2023. StructGPT: A general framework for large language model to reason over structured data. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9237–9251, Singapore. Association for Computational Linguistics.

Zhuoran Jin, Pengfei Cao, Yubo Chen, Kang Liu, and Jun Zhao. 2023. InstructoR: Instructing unsupervised conversational dense retrieval with large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 6649–6675, Singapore. Association for Computational Linguistics.

Shankar Kantharaj, Rixie Tiffany Leong, Xiang Lin, Ahmed Masry, Megh Thakkar, Enamul Hoque, and Shafiq Joty. 2022. Chart-to-text: A large-scale benchmark for chart summarization. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4005–4023, Dublin, Ireland. Association for Computational Linguistics.

Marzena Karpinska and Mohit Iyyer. 2023. Large language models effectively leverage document-level context for literary translation, but critical errors persist. In *Proceedings of the Eighth Conference on Machine Translation*, pages 419–451, Singapore. Association for Computational Linguistics.

Ioannis Konstas, Srinivasan Iyer, Mark Yatskar, Yejin Choi, and Luke Zettlemoyer. 2017. Neural AMR: Sequence-to-sequence models for parsing and generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 146–157, Vancouver, Canada. Association for Computational Linguistics.

Rémi Lebret, David Grangier, and Michael Auli. 2016. Neural text generation from structured data with application to the biography domain. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1203–1213, Austin, Texas. Association for Computational Linguistics.

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

Siheng Li, Cheng Yang, Yichun Yin, Xinyu Zhu, Zesen Cheng, Lifeng Shang, Xin Jiang, Qun Liu, and Yujiu Yang. 2023. AutoConv: Automatically generating information-seeking conversations with large language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1751–1762, Toronto, Canada. Association for Computational Linguistics.

Michela Lorandi and Anya Belz. 2023. Data-to-text generation for severely under-resourced languages with GPT-3.5: A bit of help needed from Google Translate (WebNLG 2023). In *Proceedings of the Workshop on Multimodal, Multilingual Natural Language Generation and Multilingual WebNLG Challenge (MM-NLG 2023)*, pages 80–86, Prague, Czech Republic. Association for Computational Linguistics.

Soichiro Murakami, Akihiko Watanabe, Akira Miyazawa, Keiichi Goshima, Toshihiko Yanase, Hiroya Takamura, and Yusuke Miyao. 2017. Learning to generate market comments from stock prices. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1374–1384, Vancouver, Canada. Association for Computational Linguistics.

Minh-Quang Pham, Sathish Indurthi, Shamil Chollampatt, and Marco Turchi. 2023. Select, prompt, filter: Distilling large language models for summarizing conversations. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12257–12265, Singapore. Association for Computational Linguistics.

Ratish Puduppully, Li Dong, and Mirella Lapata. 2019. Data-to-text generation with content selection and planning. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence*

and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, AAAI'19/IAAI'19/EAAI'19. AAAI Press.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Swarnadeep Saha, Xinyan Yu, Mohit Bansal, Ramakanth Pasunuru, and Asli Celikyilmaz. 2023. MURMUR: Modular multi-step reasoning for semi-structured data-to-text generation. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 11069–11090, Toronto, Canada. Association for Computational Linguistics.

David Vilar, Markus Freitag, Colin Cherry, Jiaming Luo, Viresh Ratnakar, and George Foster. 2023. Prompting palm for translation: Assessing strategies and performance.

Longyue Wang, Chenyang Lyu, Tianbo Ji, Zhirui Zhang, Dian Yu, Shuming Shi, and Zhaopeng Tu. 2023a. Document-level machine translation with large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 16646–16661, Singapore. Association for Computational Linguistics.

Yiming Wang, Zhuosheng Zhang, and Rui Wang. 2023b. Element-aware summarization with large language models: Expert-aligned evaluation and chain-of-thought method. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8640–8665, Toronto, Canada. Association for Computational Linguistics.

Dawei Zhang, Sixing Wu, Yao Guo, and Xiangqun Chen. 2022. MOBA-E2C: Generating MOBA game commentaries via capturing highlight events from the meta-data. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 4545–4556, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Haopeng Zhang, Xiao Liu, and Jiawei Zhang. 2023. SummIt: Iterative text summarization via ChatGPT. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 10644–10657, Singapore. Association for Computational Linguistics.

Yilun Zhao, Haowei Zhang, Shengyun Si, Linyong Nan, Xiangru Tang, and Arman Cohan. 2023. Investigating table-to-text generation capabilities of large language models in real-world information seeking scenarios. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 160–175, Singapore. Association for Computational Linguistics.