City, University of London

MSc Data Science Apprenticeship

Project report

2022

Using Pre-trained Transformer-based Autoregressive Language Models to Generate Data Visualisations from Natural Language Statements

William Huw Pitchford

Superviser: Dr Pranava Madhyastha

21 December 2022

Abstract

Over the past decade there has been an increase in competition between UK universities as they have become more reliant on tuition fee income. This has led to an increase in data driven decision making and, in-turn, adoption of centralised data-visualisation platforms, such as Tableau and PowerBI. However, the creation of charts and dashboards requires data manipulation and visualisation expertise. This inhibits many users' ability to create bespoke content and perform new analyses. Typically products now include natural language interfaces to reduce the reliance on technical expertise. But these systems only support analytical phrases in English and require a user to directly resolve ambiguities. In recent years, transformer-based language models have excelled at tasks requiring semantic parsing of text. The goal of this project was therefore to develop a transformer-based model capable of generating data visualisations from natural language statements. The results presented in this dissertation show successful generation of visualisations, from both English and Spanish queries, using pre-trained autoregressive models for the first time.

Table of contents

Chapter 1: Introduction

Synopsis: This chapter introduces the motivation for this research project before describing the proposed research questions and objectives. An outline of this dissertation's structure is also given.

1.1 Background

Over the past decade there has been an increase in competition between UK universities as they have become more reliant on tuition fee income. This has been caused by the removal of student number controls in 2012 and a reduction in teaching funding through funding councils by 78% in real terms since 2010/11 (Bolton, 2021). At the same time, the approach to assessing quality and standards taken by the new higher education sector regulator, the Office for Students, has been data focused. For instance, to be able to charge the maximum undergraduate home tuition fee, universities must now show evidence of a high quality academic experience and value for money via the Teaching Excellence Framework and access and participation plans (Office for Students, 2022a, Office for Students, 2022b, Office for Students, 2022c). The convergence of these phenomena has led to an increase in data driven decision making and, in-turn, adoption of centralised data-visualisation platforms, such as Tableau and PowerBI, in universities. For instance, City University now uses Tableau Online to display data related to core university functions (e.g. admissions), market intelligence and internal performance indicators.

The creation of charts and dashboards requires data manipulation and visualisation expertise. This inhibits many users' ability to create bespoke content and perform new analyses. Typically products now include natural language (NL) interfaces to reduce the reliance on technical expertise (Tableau Software, 2022, Microsoft, 2021, Narechania, Srinivasan & Stasko, 2021). Where a system parses the required data attributes, aggregation functions and chart type from the NL query. Implementation is difficult due to queries frequently being ambiguous, underspecified and informal (Setlur, Tory & Djalali, 2019). As a good user experience requires high precision and recall, commercial systems tend to therefore constrain potential inputs (Luo et al., 2021, Setlur, Tory & Djalali, 2019). For instance, Power BI and Tableau only support analytical phrases in English and require a user to actively resolve ambiguities using recommendations delivered by widgets (Setlur, Tory & Djalali, 2019). Despite these constraints, as semantic parsing is rule-based, the structure of NL requests must still be carefully considered, which has training implications for an organisation.

The challenges described above mean the task of translating NL queries to data visualisations (NL2VIS) is an active research area. The state-of-the-art models can be roughly split into two categories: end-to-end deep learning-based models and rule-based pipelined models (Narechania, Srinivasan & Stasko, 2021, Liu et al., 2021, Luo et al., 2022). The scope of this project was restricted to the former. Specifically, this dissertation describes the application of transformer-based autoregressive models, pre-trained on large web based corpora, to NL2VIS. To the best of the author's knowledge, this type of model has not yet been applied to NL2VIS. However, various studies have highlighted related benefits such as high quality semantic parsing and functional code generation (Shin, Van Durme, 2021, Chen et al., 2021). The target visualisation grammar was Vega-Lite.

1.2 Research question & objectives

"Can state-of-the-art transformer-based autoregressive language models, pre-trained using large web-based corpora, be used to generate data visualisations from natural language statements?"

The research question for this project is stated above. This was broken down into four research objectives as part of the original project proposal. Before developing the research strategy, an investigative hypothesis was formulated for each objective. This enabled the identification of key success criteria and selection of appropriate experiments. The details are outlined below. Please note the key evaluation metric throughout was the proportion of problems of which a model generated a relevant Vega-Lite specification. Where relevance to a given NL query was determined using a series of unit tests.

Research objective 1: Design and create an end-to-end model which will generate Vega-Lite visualisations from NL queries for a given dataset.

The primary output was a model capable of translating NL queries to Vega-Lite visualisation specifications. The smallest GPT-Neo model (~500 mb) was utilised as this lessened resource requirements thereby simplifying development. Nevertheless, a key design characteristic was the ability to easily switch between (and therefore later compare) different autoregressive models. The below investigative hypothesis was devised:

"Vega-Lite visualisations can be generated from natural language statements by using a pre-trained GPT-Neo-125M model in an end-to-end, modular model."

The key success criterion was comparable performance to NL4DV, which is the most advanced rule-based pipelined model available, on a publicly available NL2VIS corpus (Narechania, Srinivasan & Stasko, 2021).

Research objective 2: Compare model performance when using different sized pre-trained transformer-based autoregressive models.

The primary output was a comparative analysis of performance for different sized pre-trained transformer-based autoregressive language models. Prior research has shown improvements in code and text generation for larger models (Brown et al., 2020). Consequently the below investigative hypothesis was formulated.

"Scaling up the size of pre-trained transformed-based models will improve Vega-Lite visualisation generation performance."

This hypothesis was evaluated by comparing different GPT variants' performance on a publicly available NL2VIS corpus. The models tested were: GPT-Neo-125M (~125×10^6 parameters), GPT-Neo-1.3B (~1.3×10^9 parameters), GPT-Neo-2.7B (~2.7×10^9 parameters) and GPT-J-6B (~6×10^9 parameters).

Research objective 3: Assess the sensitivity of the best performing model to the query description.

An evaluation of performance after syntactic and semantic modifications was planned in the original research proposal. As was an assessment of the ability to capture design concepts (e.g. figure size). Neither avenue was pursued due to time constraints. Instead the primary output was an assessment of performance with Spanish NL commands. This was beneficial as competing rule-based systems are usually restricted to English phrases and a large user base for multilingual systems likely exists. The below investigative hypothesis was conceived.

"Vega-Lite visualisations can be generated from Spanish natural language commands using pre-trained transformer-based autoregressive language models."

This hypothesis was evaluated using the best performing pre-trained autoregressive model (as per research objective 2). The key success criterion was superior performance to NL4DV, which is optimised for English inputs (Narechania, Srinivasan & Stasko, 2021).

Research objective 4: Evaluate the optimal model's ability to generate visualisations from Higher Education (HE) data.

The primary output was an evaluation of NL2VIS performance on HE data using the best performing pre-trained autoregressive model (as per research objective 2). The below investigative hypothesis was devised. Once again, the key success criterion was comparable performance to NL4DV.

"Vega-Lite visualisations related to Higher Education data can be generated from natural language commands using pre-trained transformer-based autoregressive language models."

1.3 Beneficiaries

The beneficiaries of this research project are detailed below.

1) City University

An outcome of this project was proof of concept for NL2VIS using pre-trained autoregressive models. Once feasibility is proved, a basic NL2VIS prototype can foreseeably be created as Vega-Lite visualisations can be embedded into a web page. Given the limited number of Tableau Online licences available, this would increase access to data analyses and therefore support data-driven decision making across City University.

2) Wider scientific community

This is novel research that addresses a gap in the literature and practical problems faced by current commercial products. If published, it would therefore contribute to the body of scientific knowledge in this area. For instance, researchers in the NL2VIS community may use a similar approach when testing future autoregressive models.

1.4 Work plan

Figure 1.1 shows the work plan in the original research proposal. It consists of five phases: (1) project launch, (2) optimum model selection, (3) evaluate sensitivity to query description, (4) evaluation performance with HE data and (5) project completion. This plan was deviated from in two instances. First, no fine-tuning of pre-trained autoregressive model parameters, using NL2VIS examples, was conducted during the second phase. This was not part of the project's critical path and provided roughly four extra weeks of time. Second, as already explained, the third phase was adjusted so that performance on non-English NL queries was assessed rather than the effects of syntactic and syntactic modifications.

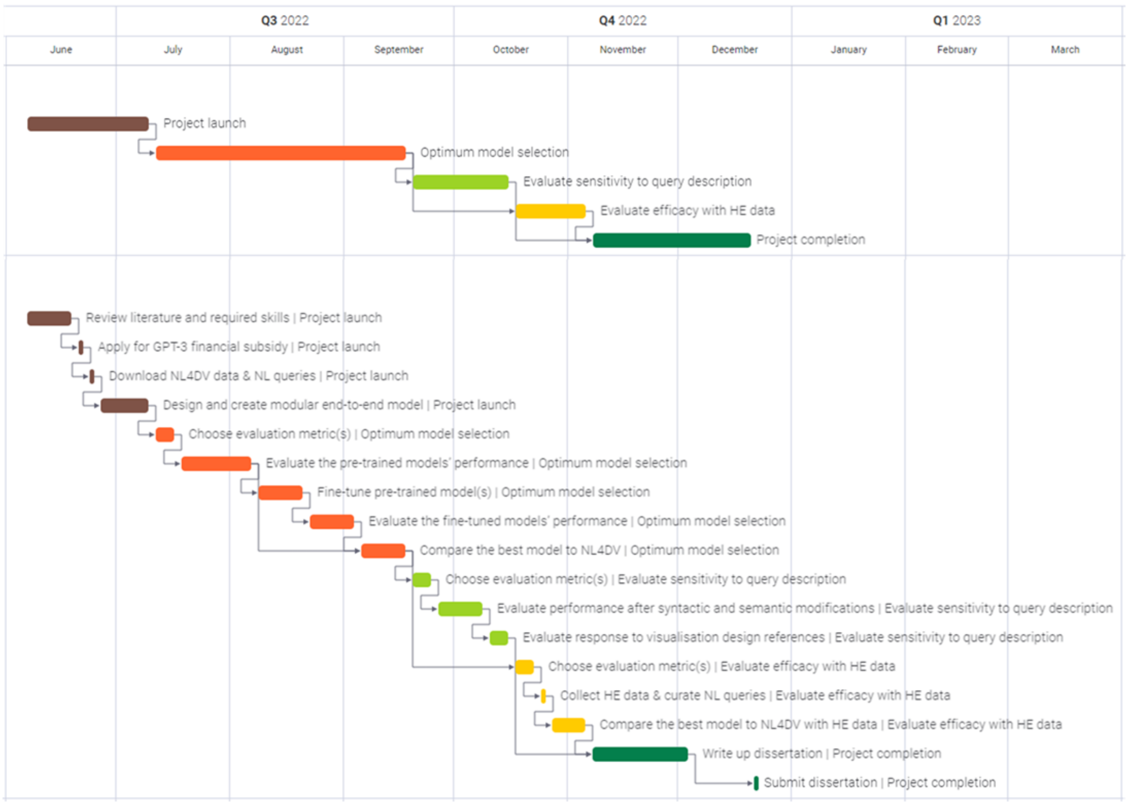


Figure 1.1 Gantt chart showing the proposed work plan. Reprinted from the original research proposal.

1.5 Dissertation outline

Chapter 2 introduces the critical context for this study. Namely prior NL2VIS models and recent advances in transformer-based language models. The methods used to implement pre-trained autoregressive models for NL2VIS are discussed in Chapter 3. The unit tests which underpin model evaluation are also explained. In Chapter 4, experimental results and analysis are provided for each of the research objectives. Chapter 4 and 5 conclude this dissertation by summarising results and discussing potential future work.

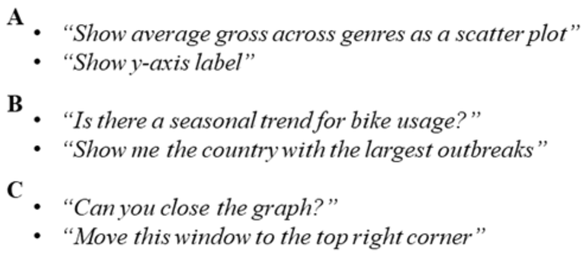
Chapter 2: Critical context

Synopsis: This chapter provides an overview of the two main types of NL2VIS model: rule-based and deep learning-based models.

2.1 Natural language to visualisation translation

The first data visualisation tool with an NL interface was presented by Cox et al in 2001 (Cox et al., 2001). Since this study, a variety of systems have been developed. Interfaces have been designed to work with three types of NL command. Firstly, visualisation focused queries used to specify or interact with a chart (Srinivasan, Stasko, 2017). Secondly, data focused queries where a question about a dataset is posed (Srinivasan, Stasko, 2017). Finally, system control queries which enable a user to manipulate a system's general user interface (Srinivasan, Stasko, 2017). Figure 2.1 shows examples for each category.

The architecture of models can be roughly split into two categories: deep learning-based models and rule-based pipelined models. The following sections will provide an overview of state-of-the-art models in both categories.



A
- *"Show average gross across genres as a scatter plot"*
- *"Show y-axis label"*

B
- *"Is there a seasonal trend for bike usage?"*
- *"Show me the country with the largest outbreaks"*

C
- *"Can you close the graph?"*
- *"Move this window to the top right corner"*

Figure 2.1 Examples of NL queries which would require A) visualisation focused capabilities, B) data focused capabilities and C) system control capabilities (Narechania, Srinivasan & Stasko, 2021).

2.2 Rule-based, pipelined models

Rule-based, pipelined models typically break down NL2VIS into three subtasks: query parsing, inference of user intent and visualisation generation (Gao, T. et al., 2015, Sun et al., 2010, Srinivasan, Stasko, 2017, Narechania, Srinivasan & Stasko, 2021). Query parsing usually involves applying standard text pre-processing techniques (i.e. tokenization, stop word removal and stemming) before using part-of-speech tagging and dependency parsing to identify relationships between different types of word. User intent is then commonly inferred by classifying different sized n-grams ($n \geq 1$) using lexicons of general phrases. This enables identification of dataset attributes (e.g. column names), analytical tasks (e.g. filter conditions and aggregation functions) and visualisation properties (e.g. chart type). Finally, by coupling outputs from the first two stages, an appropriate visualisation specification can be populated and the corresponding visualisation generated (Setlur, Tory & Djalali, 2019, Sun et al., 2010, Gao et al., 2015, Narechania, Srinivasan & Stasko, 2021, Srinivasan, Stasko, 2017).

The state-of-the-art model is NL4DV (Narechania, Srinivasan & Stasko, 2021). Its pipeline is based on the three stages described above. First, the NL command is tokenized and stemmed before using StanFord's CoreNLP (or Spacy) for POS tagging and dependency parsing. Next, references to dataset attributes are identified by computing Wu-Palmer and cosine similarity scores between n-grams ($n \geq 1$) and data source metadata (e.g. field names and values) (Wu, Palmer, 1994). Following this, references to analytical tasks and visualisation properties are searched for in premade lexicons. One or more Vega-Lite specifications are then generated and a user can select the most appropriate visualisation (Narechania, Srinivasan & Stasko, 2021). Overall, this approach has several positives: it doesn't rely on training data, its modular design means the system architecture can be modified to use an alternative visualisation library and support for conversational interaction enables users to tune a generated visualisation (Mitra et al., 2022).

A drawback of rule-based pipelined systems, including NL4DV, is their dependence on pre-made lists of visualisation types and analytical tasks. This hinders semantic parsing and limits performance with underspecified and implicit queries. Common workarounds include constraining the input (e.g. to a specific language), applying predefined rules and enabling a user to directly resolve ambiguities via a graphical user interface (GUI) (Setlur, Tory & Djalali, 2019, Sun et al., 2010, Gao et al., 2015, Narechania, Srinivasan & Stasko, 2021, Srinivasan, Stasko, 2017). For example, DataTone asks users to directly resolve uncertainties in column names using drop-down menus. Furthermore, it can infer a relevant visualisation type using other detected attributes (e.g. a bar chart when comparing quantitative values across nominal categories) (Gao et al., 2015). NL4DV also utilises pre-defined rule when no analytical task or visualisation type is detected. In addition, developers can input common aliases for column names (Narechania, Srinivasan & Stasko, 2021). All of these strategies require human intervention and this hampers generalisation to new domains. There is therefore a demand for alternative approaches to semantic parsing.

2.3 Deep learning-based models

The most advanced deep learning-based NL2VIS models use portions of the transformer architecture. This architecture was introduced by Vaswani et al in 2018 (Vaswani et al., 2017). Its key advantage is its structure facilitates parallelised training. This has enabled substantial increases in model and training corpus size, which has led to superior latent representations of NL structure. As a result, transformer-based language models have demonstrated state-of-the-art performance across a range of natural language processing (NLP) tasks such as machine translation and document summarisation (Vaswani et al., 2017).

Three variants of the transformer architecture are commonly used: masked language models (e.g. BERT (Devlin et al., 2019)) that use just the encoder section, autoregressive language models (e.g. GPT-n series (Brown et al., 2020, Radford et al., 2018)) that use just the decoder section, and encoder-decoder models (e.g. AlphaCode (Li et al., 2022)) that use both components (Min et al., 2021). Examples will be presented after a short overview of how the Transformer works.

2.3.1 The Transformer

The transformer models sequence-to-sequence (seq2seq) problems as the mapping of an input sequence of *n* vectors (equation 2.1) to a target sequence of vectors with variable length *m* (equation 2.2). Where each of the input vectors is a word (or subword) embedding. Whilst each of the output vectors is a word (or subword) sampled from a vocabulary (Min et al., 2021, Vaswani et al., 2017).

$$X_{1:n} = \{x_1, \dots, x_n\} \tag{2.1}$$

$$Y_{1:m} = \{y_1, \dots, y_m\} \tag{2.2}$$

Where, the input and output sequence is given by $X_{1:n}$ and $Y_{1:m}$, respectively. For translation, each vector represents a word (or token) (Rush, Nguyen & Klein, 2018, von Platen, 2020).

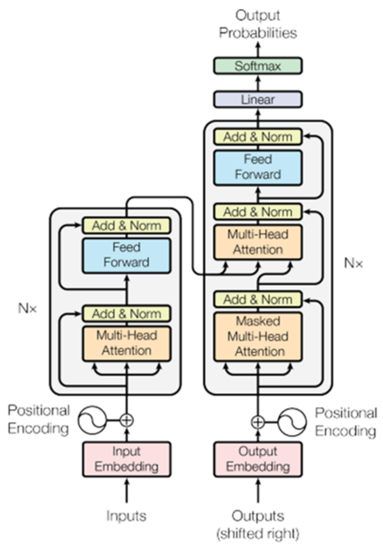


Figure 2.2 The transformer architecture introduced by Vaswani and co-workers (Vaswani et al., 2017). The left and right hand side correspond to the encoder and decoder blocks, respectively.

It achieves this mapping using an encoder-decoder structure (Figure 2.2). The encoder is used to map the input sequence to a sequence of context dependent vector representations for each word (equation 2.3). The decoder is then used to define the conditional probability distribution of the output sequence given this contextualised encoding sequence (equation 2.4). Crucially, Bayes theorem shows this is the product of the conditional probability of each output vector (equation 2.5). The conditioning of the output on the contextualised encoding sequence ($\overline{x}_{1:n}$) and all previous output vectors $Y_{0:i-1}$ should be noted (von Platen, 2020, Vaswani et al., 2017, Rush, Nguyen & Klein, 2018).

$$f_{enc}: X_{1:n} \rightarrow \bar{X}_{1:n} \tag{2.3}$$

$$p_{dec}(Y_{1:m}|\bar{X}_{1:n}) \tag{2.4}$$

$$p_{dec}(Y_{1:m}|\bar{X}_{1:n}) = \prod_{i=1}^{m} p_{dec}(y_i|Y_{0:i-1}, \bar{X}_{1:n}) \tag{2.5}$$

Where, the input and output sequence is given by $X_{1:n}$ and $Y_{1:m}$, respectively. Each y_i vector represents a word in the output sequence. The contextualised encoding sequence is given by $\bar{x}_{1:n}$.

The key steps by which the Transformer architecture enables these processes are described below.

1. The input sequence is tokenized (i.e. split) into words (or sub-words) mapped to token ids found in a tokenizer vocabulary (Vaswani et al., 2017).
2. Each token id is embedded using a matrix learned during pre-training. An additional matrix is used to encode the position of each token. They are summed thereby generating a sequence of d_{model}-dimensional vectors for each token in the input sequence (Vaswani et al., 2017).
3. The sequence is passed to the encoder. Each position corresponds to an input token. Each layer of the encoder contains a multi-head self-attention layer and a feed forward neural network. The first self-attention layer creates the contextualised encoding sequence ($\bar{x}_{1:n}$) and this is refined over successive layers. Each vector in the encoded sequence depends on (i.e. attends to) all embeddings in the input sequence ($X_{1:n}$) (Vaswani et al., 2017).

   Note: The Transformer uses an attention function called "scaled dot-product attention" (described in Appendix A). Multi-head attention involves applying this function multiple times in parallel before combining the outputs. This process enables the capture of different range dependencies (Vaswani et al., 2017).
4. The decoder generates the output sequence one position at a time. Layers are similar to that of the encoder but each position corresponds to an output token and an additional multi-head attention sub layer is present. The self-attention sub layer is masked meaning each position attends to only previous outputs ($Y_{0:i-1}$). The additional sub layer enables attention to the contextualised encoding sequence ($\bar{x}_{1:n}$) (Vaswani et al., 2017).
5. The decoder outputs a vector for each position. This is passed through a feed forward neural network (also known as the language modelling head) to create a logits vector. Crucially, its dimension equals the size of the vocabulary. Application of the softmax function therefore produces the probability distribution across the vocabulary (von Platen, 2020, Vaswani et al., 2017, Rush, Nguyen & Klein, 2018).
6. A decoding mechanism is used to sample the output sequence one word at a time from the conditional probability distribution (equation 2.6).(von Platen, 2020, Vaswani et al., 2017, Rush, Nguyen & Klein, 2018).

$$p_{dec}(y_i | Y_{0:i-1}, \bar{X}_{1:n}), \ \forall_i \epsilon \{1, \ldots, m\} \tag{2.6}$$

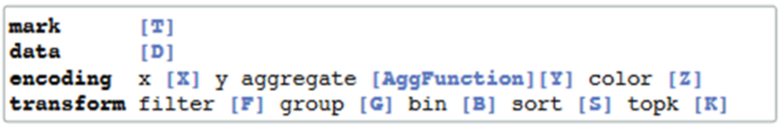
Where the previously generated output sequence is given by $Y_{0:i-1}$. Each y_i vector represents a word in the output sequence. The contextualised encoding sequence is given by $\bar{x}_{1:n}$.

2.3.2 Encoder-decoder models

NL2VIS can be viewed as a sequence-to-sequence (seq2seq) problem where an input sequence of text is mapped to an output visualisation specification. Given the distinct difference in the type of input and output sequence, this mapping is well suited to encoder-decoder models, which use both portions of the transformer architecture (Min et al., 2021).

In early 2022, Luo et al applied such a model to NL2VIS (Luo et al., 2022). It works as follows. Firstly, the input sequence is created by concatenating embeddings generated for the NL command, related dataset and a masked chart template. This is the input to the encoder, which generates a contextual representation of the sequence. Next, the decoder is used to autoregressively generate tokens for a masked specification. The basic template, called Vega-Zero, is shown in Figure 2.3. Masked tokens relate to the visualisation mark type, data source, data transformations (e.g. filtering conditions) and visual encodings (e.g. x/y-axis encodings). Finally, the researchers used pre-defined rules to map the completed template to a Vega-Lite specification (a common Python visualisation library) (Luo et al., 2022).

Compared to NL4DV, the design of ncNET has two key advantages. Firstly, it provides a direct mapping from the NL command to a visualisation specification via a single model (Liu et al., 2021, Luo et al., 2022, Narechania, Srinivasan & Stasko, 2021). This means the visualisation type does not need to be generated using predefined rules. Secondly, the user interface enables the user to constrain specific visualisation properties (e.g. mark type) by selecting a preferred chart type. This results in partial completion of the masked template thereby reducing the solution search space and alleviating problems with underspecified and ambiguous commands. Despite these positives, the model does have its limitations. Firstly, Vega-Zero does not enable multi-view charts, composite charts with different mark types or any interactivity. Secondly, it does not utilise a particularly large training corpus, which tends to limit the generalisation of systems (Brown et al., 2020, McCoy, Pavlick & Linzen, 2019). The nVBench corpus was used, which contains 25,750 (NL command, Vega-Lite specification) pairs across 105 domains. This is relatively small compared to similar models (BART: 3.3 billion tokens from Wikipedia and eBooks) (Lewis et al., 2019). As a result, application to new domains may require additional data to update model weights (Luo et al., 2022).



```
mark      [T]
data      [D]
encoding  x [X] y aggregate [AggFunction][Y] color [Z]
transform filter [F] group [G] bin [B] sort [S] topk [K]
```

Figure 2.3 The masked chart template used by ncNET (Luo et al., 2022). This is a simplified and flattened version of Vega-Lite called Vega-Zero.

2.3.3 Masked language models

Masked language models (MLMs) use the encoder part of the transformer architecture to generate a bidirectional contextualised representation of an input sequence (Min et al., 2021). They are typically used for semantic parsing where encoding acts as a form of context sensitive feature extraction and encoded sequences are used by a downstream model for a specific task (Min et al., 2021). This approach was first demonstrated by Devlin and co-workers in 2018 using BERT (short for Bidirectional Encoder Representations from Transformers) (Devlin et al., 2019).

In 2021, Li et al introduced a model called ADVISor, which formulated the NL2VIS task as a classification problem underpinned by the BERT model (Liu et al., 2021). Figure 2.3 shows the model pipeline. It is designed to generate visualisations for a given tabular dataset from an NL command. First, an input sequence is created by concatenating the NL command with table column names. Next, the BERT model (pre-trained) is used to generate a contextual representation of this sequence. This representation is then used by two feed-forward neural network based classifiers, which output the necessary data area (i.e. fields and filter conditions) and aggregation functions (e.g. sum). Like NL4DV, pre-defined rules are then used to select an appropriate type of visualisation (Liu et al., 2021, Narechania, Srinivasan & Stasko, 2021).

Using ADVISor, Li and co-workers reported superior performance to NL4DV for NL queries containing implicit references to attributes. This indicates the benefits of using BERT for semantic parsing. However, the ADVISor pipeline does have its drawbacks. Firstly, a large quantity of additional data (80,654 NL commands on 24,241 tables) was required to train the classifiers and fine-tune BERT. This may hamper out-of-distribution generalisation by the model (McCoy, Pavlick & Linzen, 2019, Brown et al., 2020). Furthermore, the use of predefined rules to generate visualisations means all possible requests are difficult to capture (Liu et al., 2021).

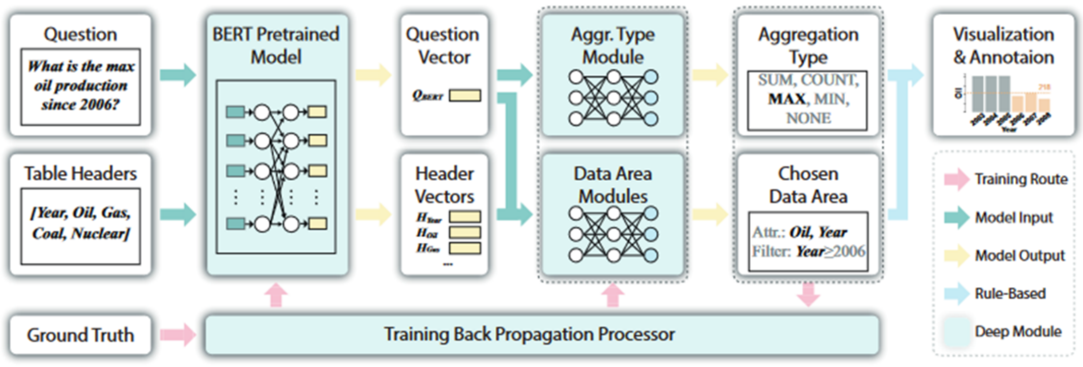
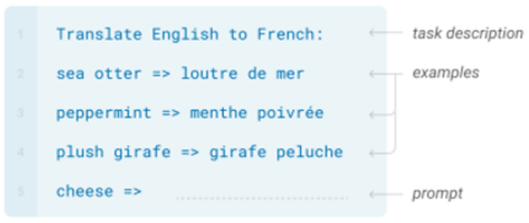


Figure 2.3. The ADVISor pipeline developed by Li et al (Liu et al., 2021).

2.3.4 Autoregressive language models

Autoregressive language models are trained to predict the probability of a word given an input and all previously generated words (Min et al., 2021). They use just the decoder section of the Transformer architecture. They are commonly applied to NL problems using so-called prompt based learning (Brown et al., 2020, Radford et al., 2018). This approach relies on framing a problem as a generative task using short pieces of text called prompts. Prompts may include a task description and completed in-context examples of the task (Figure 2.4). Using three in-context examples would be referred to as 3-shot learning. Whilst including none is called 0-shot learning (Brown et al., 2020).



Translate English to French:
sea otter => loutre de mer
peppermint => menthe poivrée
plush girafe => girafe peluche
cheese =>

Figure 2.4 An example of prompt-based learning demonstrated by Brown et al where a task description and three completed examples are added to the input (Brown et al., 2020).

As models and training datasets have increased in size, prompt-based learning has exhibited strong performance across a range of NL tasks without requiring any additional finetuning of model weights after pre-training (Brown et al., 2020). This has incentivised the collection of increasingly large, unlabelled, continuous training corpora from diverse sources across the internet. For example, the Colossal Clean Crawled Corpus was curated from over 365 million domains and contains roughly 156 billion tokens (Dodge et al., 2021, Raffel et al., 2020). It has now been shown that models pre-trained using corpora containing code from GitHub can generate functional code using prompt-based learning. For instance, Hendrycks et al have demonstrated GPT models (with between ~1 billion and ~175 billion parameters) are capable of solving Python coding challenges (albeit with accuracies ≤ ~5%) when the problem and completed examples are included in the prompt (Hendrycks et al., 2021). Similarly, using models with between 244 million and 137 billion parameters, Austin et al have demonstrated synthesis of Python functions from programming problems (Austin et al., 2021). Lastly, Chen and co-workers have shown a descendant of GPT-3, called Codex, can generate Python functions from prompts containing doc strings (Chen et al., 2021) .

To the best of the author's knowledge, no such model has yet been applied to NL2VIS. However, the studies described above have highlighted relevant benefits. Firstly, various open-source models pre-trained on corpora containing code are available and free to download (e.g. GPT-J and GPT-Neo) (Wang, Komatsuzaki, 2021, Black et al., 2021). Secondly, NL2VIS can foreseeably be framed as a generative problem and these models can infer both text and code given an appropriate prompt. If complete examples are tailored to problems, this could rule out the need for fine-tuning. Next, the large pretraining corpora mean models may capture the syntax of different visualisation libraries (e.g. Vega-Lite and MatPlotLib) (Brown et al., 2020, Min et al., 2021, Austin et al., 2021). It may therefore be possible to generate a broad range of visualisations unconstrained by a visualisation library or domain. Lastly, the large pre-training corpora also means these models capture general linguistic knowledge. This could improve semantic parsing of underspecified and ambiguous NL commands (Brown et al., 2020, Min et al., 2021, Austin et al., 2021).

2.4 Conclusion

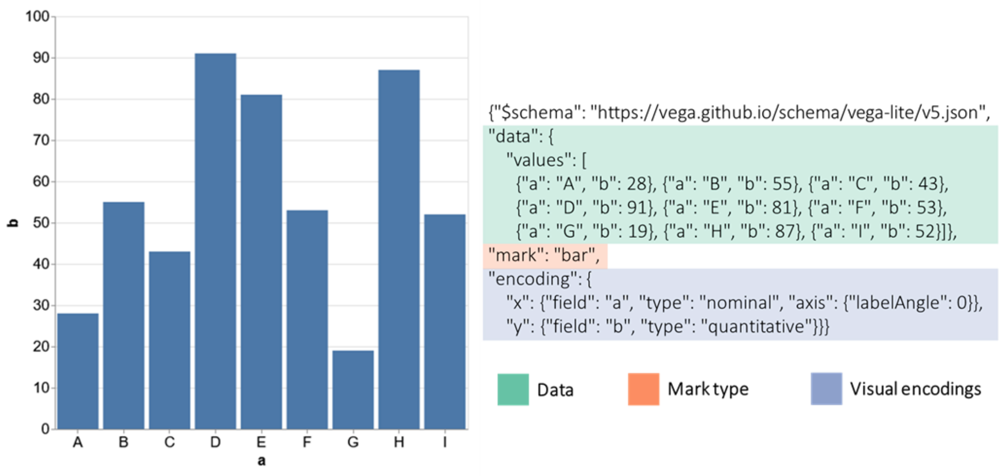
NL2VIS can be viewed as a sequence-to-sequence problem where an input sequence of text is mapped to an output visualisation specification. In recent years, transformer-based language models have excelled at such tasks. However, a review of the literature has shown that pre-trained transformer based autoregressive models have not yet been applied to NL2VIS. This type of model will therefore be investigated.

Chapter 3: Methods

Synopsis: The methods used to implement pre-trained autoregressive models for NL2VIS are discussed in this chapter. This includes a description of the chosen visualisation library, data and model implementation. The unit tests which underpin model evaluation are also explained.

3.1 Visualisation library

This project focused on developing a model capable of generating Vega-Lite visualisation specifications, which were then rendered using the Vega-Altair library (https://vega.github.io/vega-lite/). Vega-Lite is a declarative visualisation grammar where specifications are JSON objects that describe the data source, data transformations, mark types and visual encodings (i.e. mapping of variables to visual properties such as position and colour) required by a chart (Satyanarayan et al., 2017). Figure 3.1 provides an example of a specification describing a simple bar chart. Vega-Lite was chosen for three reasons. Firstly, it is more concise than imperative visualisation libraries like Matplotlib, which require a more detailed description of the steps involved in plotting. It is assumed this will simplify model inference. Secondly, recent studies in this area have utilised Vega-Lite meaning performance can be benchmarked using existing NL2VIS models and corpuses (Narechania, Srinivasan & Stasko, 2021, Luo et al., 2021, Srinivasan et al., 2021). Finally, its JSON based syntax enables interactive visualisations which can be embedded in a webpage. This is a core component of City University's current approach to delivering business intelligence.



```
{"$schema": "https://vega.github.io/schema/vega-lite/v5.json",
"data": {
  "values": [
    {"a": "A", "b": 28}, {"a": "B", "b": 55}, {"a": "C", "b": 43},
    {"a": "D", "b": 91}, {"a": "E", "b": 81}, {"a": "F", "b": 53},
    {"a": "G", "b": 19}, {"a": "H", "b": 87}, {"a": "I", "b": 52}]},
"mark": "bar",
"encoding": {
  "x": {"field": "a", "type": "nominal", "axis": {"labelAngle": 0}},
  "y": {"field": "b", "type": "quantitative"}}}
```

Figure 3.1 A simple bar chart created using a Vega-Lite specification. The core components of the corresponding specification (data, mark type and visual encodings) are highlighted.

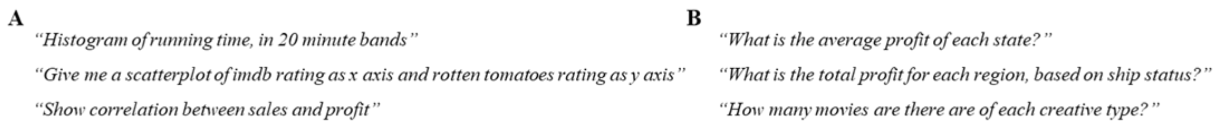
3.2 Data

3.2.1 Cross-domain corpus

Srinivasan and co-workers at Georgia Institute of Technology have previously collected a (NL command, Vega-Lite specification) corpus via an online study. Participants in the online survey were presented with a fixed set of visualisations (shown in Appendix B) and asked to propose NL commands for generating the chart (Srinivasan et al., 2021). This corpus was combined with examples published by Narechania et al when showcasing the capability of NL4DV (Narechania, Srinivasan & Stasko, 2021). In total, this provided 801 NL queries for 81 distinct visualisations across four heterogeneous datasets (described in Appendix C). The combined data set was named the cross-domain corpus (cdCorpus). It has several benefits. Firstly, multiple data sources will enable testing of in-domain and out-of-domain problems. Secondly, four popular chart types (and their variants) are well represented: bar chart, histogram, line chart and scatter chart. Table 3.1 shows the percentage breakdown of NL queries by chart type. Finally, both data focused queries (where a question about a dataset is posed) and visualisation focused queries (where a chart is requested) are present (Figure 3.2).

Chart type	**n**	**%**
Area chart	2	0.2
Bar chart (basic)	183	22.8
Bar chart (multiples)	63	7.9
Box plot	1	0.1
Histogram	88	11.0
Line chart	83	10.4
Line chart (coloured)	68	8.5
Pie chart	2	0.2
Scatter chart	94	11.7
Scatter chart (coloured)	71	8.9
Scatter chart (multiples)	68	8.5
Stacked bar chart	71	8.9
Strip plot	5	0.6
Table	2	0.2
Total	**801**	**100.0**

Table 3.1 A proportional breakdown of chart types in the (NL command, Vega-Lite specifications) corpus.



A

"Histogram of running time, in 20 minute bands"

"Give me a scatterplot of imdb rating as x axis and rotten tomatoes rating as y axis"

"Show correlation between sales and profit"

B

"What is the average profit of each state?"

"What is the total profit for each region, based on ship status?"

"How many movies are there are of each creative type?"

Figure 3.2 Examples of A) visualisation-focused and B) data-focused NL statements.

3.2.2 Higher Education corpus

An online survey was used to collect (NL command, Vega-Lite specification) pairs related to HE data. The survey is presented in Appendix D. The survey was posted on the SampleSize Reddit forum (https://www.reddit.com/r/SampleSize/, membership size: 197k) with the goal of recruiting typical end users for a visualisation system. As per the ethical guidelines, participation was voluntary and anonymous. Furthermore, participants received no compensation. The information provided to all participants can be found at the beginning of the survey. Please note this research project's ethics review form and approval are shown in Appendix E and F, respectively.

Similar to Srinivasan *et al*, participants were presented with 13 visualisations and asked to propose NL commands for generating the chart (Srinivasan et al., 2021). The survey covered five distinct chart types (bar chart, scatter plot, strip plot, line chart and histogram) and their variants (i.e. stacked bar chart, multiple bar chart, multi-series line chart, multiple scatter plots and coloured scatter plot). These were chosen as they visualise different quantitative relationships: nominal comparison (via bar charts), time-series (via line charts), part-to-whole (via stacked bar charts), frequency distribution (via strip plots and histograms) and correlation (via scatter plots). Each visualisation corresponded to one of three heterogeneous datasets related to the Complete University Guide league table rankings (described in Appendix G). These were selected as they provide different data types (i.e. quantitative, categorical and temporal) and are widely used by UK Universities to benchmark performance and develop marketing materials. Please note a preview of the underpinning data was provided to participants as this can result in more natural NL commands (Srinivasan et al., 2021).

Over the course of roughly 10 days, 36 individuals visited the survey URL. However, only 18 of these participants completed the survey and not all respondents answered every question. Some data cleaning was conducted. This involved removal of responses containing irrelevant text (e.g. "no idea. seems complicated") or those where the task was clearly misunderstood (e.g. single word answers like "Member"). This left a total of 136 NL commands, which are presented in Appendix H. Table 3.2 shows the percentage associated with each chart type. Given the small size of the dataset, these were not labelled as fully-specified and under-specified. Furthermore, as no model training was conducted using this data, no train/test split was conducted. The final corpus of (NL command, Vega-Lite specification pairs) was named the Higher Education corpus (heCorpus).

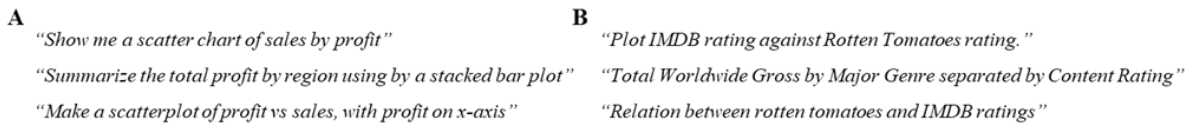
The approach taken reduced the amount time required to recruit participants but the results have clear limitations. Firstly, the small sample size and introduction of self-selection bias mean the results are not representative of the target population (i.e. the SampleSize reddit forum with 197k members). As an example, Oates recommends a sample size of 888 for a target population of 100,000 people (confidence level: 95%, accuracy range: ± 3%) (Oates, 2006). Secondly, unlike the corpus presented in section 3.2.1, almost all responses explicitly or implicitly request a visualisation rather than pose a question about the underpinning dataset. This is a direct consequence of the survey's design and hinders assessment of data focused capabilities. Despite these limitations, it was decided the dataset was sufficient for a proof of concept.

Visualisation	n	%	M
Bar chart (basic)	14	10.3	13
Bar chart (sorted)	10	7.4	19
Bar chart (multiples)	12	8.8	12
Stacked bar chart	11	8.1	23
Stacked bar chart (%)	11	8.1	24
Line chart	10	7.4	13
Line chart (multi-series)	10	7.4	17
Line chart (multiples)	10	7.4	19
Histogram	9	6.6	10
Strip plot	9	6.6	11
Scatter plot	10	7.4	13
Scatter plot (coloured)	10	7.4	17
Scatter plot (multiples)	10	7.4	17
Grand Total	**136**	**100.0**	**16**

Table 3.2 Proportional breakdown of the Higher Education test set by visualisation type. Where, n is the number of NL prompts, % is the percentage of all prompts and M is the median length in number of words.

3.2.3 Training and test sets

The cdCorpus was used for hyperparameter optimisation. Consequently it was randomly split into a training and test set (training set: 80%; test set: 20%). Training set instances were manually labelled as fully-specified or underspecified. Fully-specified problems represent the simplest use case: NL commands containing explicit references to the required variables, data aggregation functions and visualisation type. Under-specified commands are those missing at least one of these attributes. Figure 3.3 shows examples of each category. This classification system could be ambiguous and it was sometimes unclear whether a command should be excluded. For instance, some statements appeared incomplete (i.e. missing attributes which could not reasonably be inferred) and others referenced the wrong chart type (e.g. confusing bar charts with histograms). Ideally, multiple data labellers would be used so classifications could be verified. However, this was not possible in the given timeframe. To avoid introducing bias, the test set wasn't labelled as its smaller size increases the margin for error.


A

"Show me a scatter chart of sales by profit"

"Summarize the total profit by region using by a stacked bar plot"

"Make a scatterplot of profit vs sales, with profit on x-axis"

B

"Plot IMDB rating against Rotten Tomatoes rating."

"Total Worldwide Gross by Major Genre separated by Content Rating"

"Relation between rotten tomatoes and IMDB ratings"

Figure 3.3 Examples of A) fully-specified and B) under-specified NL queries.

Tables 3.3 and 3.4 show the proportional breakdown of problems by chart type for the training and test set, respectively.

Visualisation	Fully specified			Under specified			Grand Total		
	n	%	M	n	%	M	n	%	M
Area chart	1	0.2	13	0	0.0	-	1	0.2	13
Bar chart (basic)	43	6.7	11	104	16.3	7	147	23.0	8
Bar chart (multiples)	15	2.3	15	35	5.5	11	50	7.8	12
Box plot	0	0.0	-	1	0.2	5	1	0.2	5

Histogram	23	3.6	7	44	6.9	7.5	67	10.5	7
Line chart	14	2.2	9	55	8.6	8	69	10.8	8
Line chart (multi-series)	19	3.0	12	38	5.9	9.5	57	8.9	11
Pie chart	2	0.3	9.5	0	0.0	-	2	0.3	9.5
Scatter chart	31	4.8	9	39	6.1	7	70	10.9	8
Scatter chart (coloured)	23	3.6	11	33	5.2	10	56	8.8	10
Scatter chart (multiples)	29	4.5	12	28	4.4	9.5	57	8.9	10
Stacked bar chart	23	3.6	15	35	5.5	11	58	9.1	12
Strip plot	1	0.2	8	3	0.5	15	4	0.6	11.5
Table	0	0.0	-	1	0.2	14	1	0.2	14
Grand Total	**224**	**35.0**	**11**	**416**	**65.0**	**8**	**640**	**100.0**	**9**

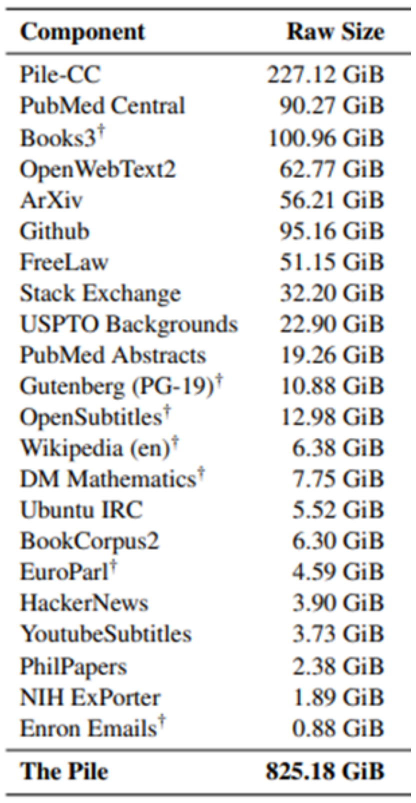
Table 3.3 Proportional breakdown of the cdCorpus train set by visualisation type and prompt type (fully-specified/under-specified). Where, n is the number of NL prompts, % is the percentage of all prompts and M is the median length in number of words.

Visualisation	**n**	**%**	**M**
Area chart	1	0.6	14
Bar chart (basic)	36	22.4	7
Bar chart (multiples)	13	8.1	14
Histogram	21	13.0	7
Line chart	14	8.7	9
Line chart (multi-series)	11	6.8	10
Scatter chart	24	14.9	8
Scatter chart (coloured)	15	9.3	11
Scatter chart (multiples)	11	6.8	11
Stacked bar chart	13	8.1	11
Strip plot	1	0.6	9
Table	1	0.6	17
Grand Total	**161**	**100.0**	**9**

Table 3.4 Proportional breakdown of the cdCorpus test set by visualisation type and prompt type. Where, n is the number of NL prompts, % is the percentage of all prompts and M is the median length in number of words.

3.3 Pre-trained language models

Two pre-trained autoregressive models developed by EleutherAI were used in this research project: GPT-Neo and GPT-J (Wang, Komatsuzaki, 2021, Black et al., 2021). Both were pre-trained using the Pile (Gao, L. et al., 2021). They were chosen as prior studies have shown they can generate functional code when pre-trained using the Pile (Hendrycks et al., 2021). Table 3.5 provides an overview of the constituent datasets. Crucially, code from GitHub and multilingual text from the proceedings of the European Parliament are present and YouTube subtitles are included (Gao et al., 2021).



Component	Raw Size
Pile-CC	227.12 GiB
PubMed Central	90.27 GiB
Books3[†]	100.96 GiB
OpenWebText2	62.77 GiB
ArXiv	56.21 GiB
Github	95.16 GiB
FreeLaw	51.15 GiB
Stack Exchange	32.20 GiB
USPTO Backgrounds	22.90 GiB
PubMed Abstracts	19.26 GiB
Gutenberg (PG-19)[†]	10.88 GiB
OpenSubtitles[†]	12.98 GiB
Wikipedia (en)[†]	6.38 GiB
DM Mathematics[†]	7.75 GiB
Ubuntu IRC	5.52 GiB
BookCorpus2	6.30 GiB
EuroParl[†]	4.59 GiB
HackerNews	3.90 GiB
YoutubeSubtitles	3.73 GiB
PhilPapers	2.38 GiB
NIH ExPorter	1.89 GiB
Enron Emails[†]	0.88 GiB
The Pile	**825.18 GiB**

Table 3.5 Overview of datasets in the Pile (Gao et al., 2021).

In terms of structure, both GPT-J and GPT-Neo consist of a stack of transformer-decoder blocks. Where, each transformer-block consists of a masked multi-head self-attention layer and a feedforward network. GPT-Neo uses local attention rather than global attention in every other layer (Wang, Komatsuzaki, 2021, Black et al., 2021). This means it attends to a subset of tokens determined by a sliding attention window, which is 256 tokens in size. In contrast, GPT-J uses global attention in all layers. In order to study the effect of model size, three variants of GPT-Neo were tested: GPT-Neo-125M, GPT-Neo-1.3B and GPT-Neo-2.7B. The original research proposal also included plans to use GPT-3 (~175 billion parameters). Unfortunately, an Open AI Researcher Access Program application was unsuccessful so this wasn't possible due to the high cost. Table 3.6 shows key hyperparameters for all models.

Model name	n_{params}	n_{layers}	n_{heads}	d_{model}	d_{ff}	$l_{sequence}$
GPT-Neo-125M	125M	12	12	768	3,072	2,048
GPT-Neo-1.3B	1.3B	16	24	2,048	8,192	2,048
GPT-Neo-2.7B	2.7B	20	32	2,560	10,240	2,048
GPT-J-6B	6B	28	16	4,096	16,384	2,048

Table 3.6 Key hyperparameters for all models used in this research project. Where n_{params} is the total number of parameters, n_{layers} is the number of decoder blocks, n_{heads} is the number of self-attention heads in each decoder block, d_{model} is the model dimension, d_{ff} is the feedforward network dimension and $l_{sequence}$ is the max size of the input sequence .

The HuggingFace library (https://huggingface.co/) was used to implement all models. This library was chosen for two reasons. Firstly, pre-trained models with a causal language modelling head can be downloaded locally free-of-charge. Secondly, it is simple to instantiate and switch between different models and tokenizers using the generic AutoModel and AutoTokenizer classes (Hugging Face, 2022). When building the model, only GPT-Neo-125M was used. Its low memory consumption (~500 MB) and short inference time simplified development and enabled use of the free tier of Google Colab (typical resources allocated with a NVIDIA T4 GPU: 12GB RAM, 16GB GPU RAM). Once a working prototype was developed, all further experiments were conducted using City University's Hyperion higher performance cluster (HPC). In terms of hardware, Hyperion includes 72 compute nodes and four GPU nodes. All compute nodes have 2 Intel® Xeon® Gold 6248R processors with 24 cores, 384 GB RAM and a 1TB SSD, respectively. Two GPU nodes with a total of 12 NVIDA A100 Tensor Core GPUs were available for use.

3.4 Model implementation

3.4.1 Model design

The high-level overview of the required system architecture, shown in Figure 3.7, was devised. It was decided a prototype, rather than a complete system, would be created with the aim of demonstrating proof of concept. In practice, this meant no user interface was designed and distinct Python scripts were used for prompt engineering, model inference (i.e. specification generation) and processing model outputs (i.e. parsing, validation and rendering).

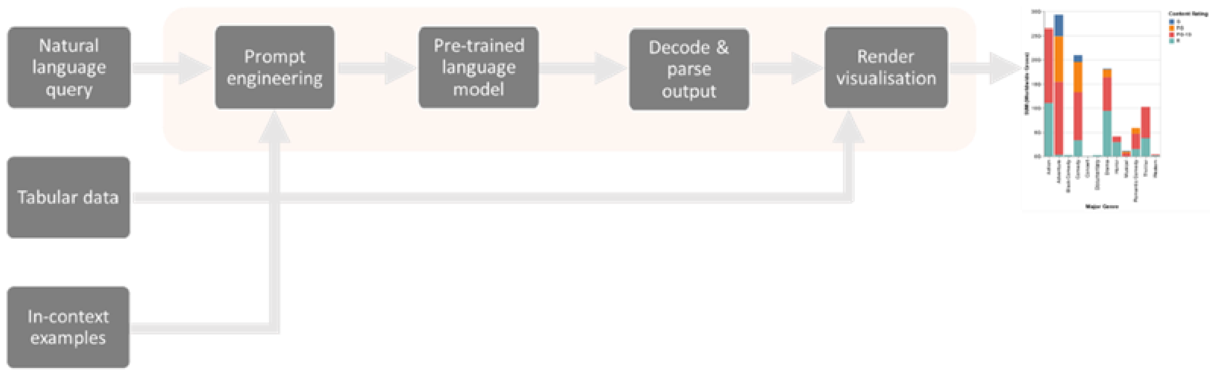
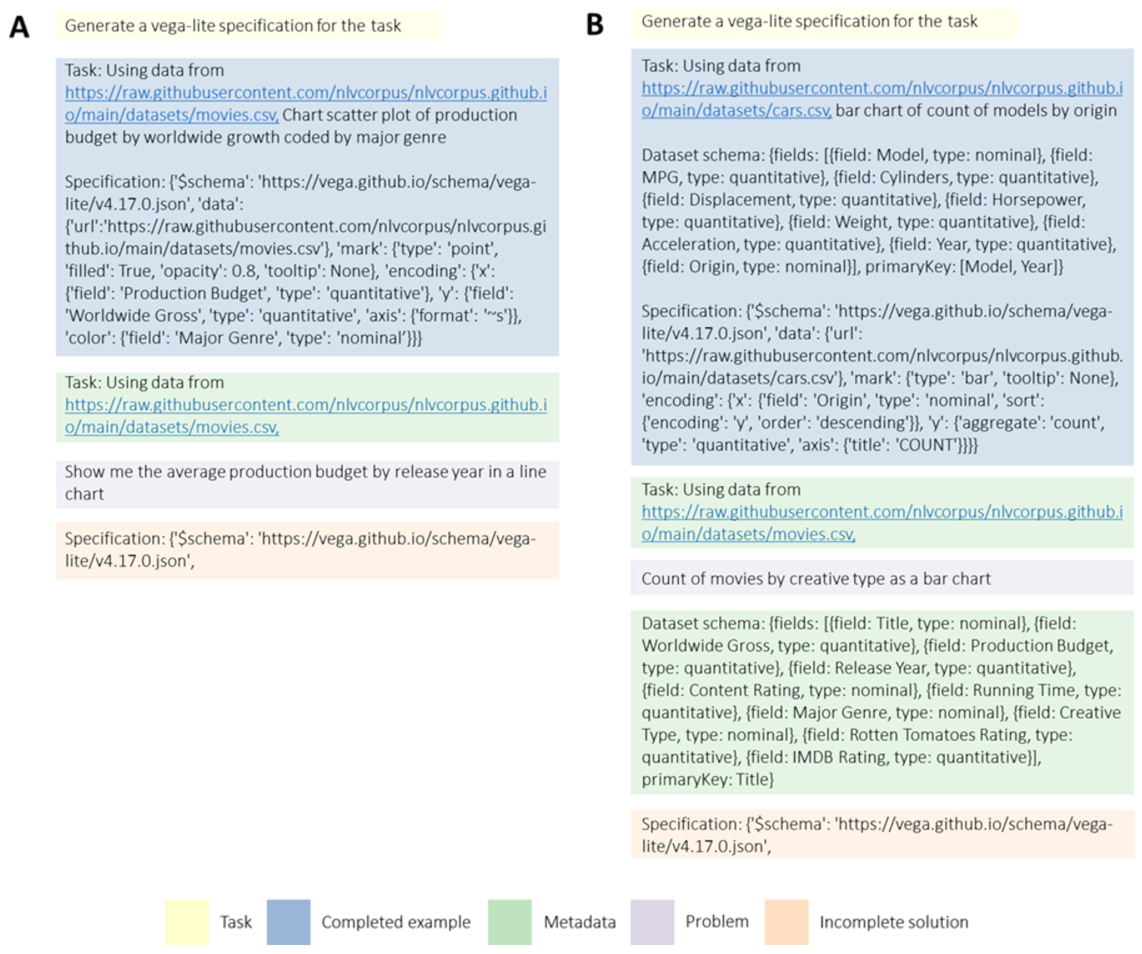


Figure 3.7 High-level overview of the model design.

3.4.2 Prompt engineering

The basic template used to construct prompts was similar to that used by Brown et al (Brown et al., 2020). However, it included an additional placeholder for metadata describing the content of the relevant dataset. This was crucial as the model must be able to infer the data source URL, column names and data types when generating a Vega-Lite specification. In total there were five key components: a description of the task, completed in-context examples, metadata, the problem of interest and an incomplete solution which acted as a prefix to the generated tokens.

Two types of problems were assessed in this project: in-domain and out-of-domain problems. In-domain problems could be matched with in-context examples associated with the same dataset. Whilst out-of-domain problems could only be matched with in-context examples associated with different datasets. Figure 3.8 shows an example of the final model input for both. For out-of-domain problems, the metadata includes the data source URL and a structured table schema. The latter provides a description of each variable (field name and datatype) and the variable(s) which could be used to uniquely identify each record (called a 'primarykey'). Its design is similar to the json table schema defined by dataprotocols.org. In contrast, only the data source URL is present for in-domain problems. As in-context examples can be associated with the same dataset, it was hoped the model would be able to infer column names and datatypes. This was advantageous as a shorter model input reduces the computation time.



A

Generate a vega-lite specification for the task

Task: Using data from https://raw.githubusercontent.com/nlvcorpus/nlvcorpus.github.io/main/datasets/movies.csv, Chart scatter plot of production budget by worldwide growth coded by major genre

Specification: {'$schema': 'https://vega.github.io/schema/vega-lite/v4.17.0.json', 'data': {'url':'https://raw.githubusercontent.com/nlvcorpus/nlvcorpus.github.io/main/datasets/movies.csv'}, 'mark': {'type': 'point', 'filled': True, 'opacity': 0.8, 'tooltip': None}, 'encoding': {'x': {'field': 'Production Budget', 'type': 'quantitative'}, 'y': {'field': 'Worldwide Gross', 'type': 'quantitative', 'axis': {'format': '~s'}}, 'color': {'field': 'Major Genre', 'type': 'nominal'}}}

Task: Using data from https://raw.githubusercontent.com/nlvcorpus/nlvcorpus.github.io/main/datasets/movies.csv,

Show me the average production budget by release year in a line chart

Specification: {'$schema': 'https://vega.github.io/schema/vega-lite/v4.17.0.json',

B

Generate a vega-lite specification for the task

Task: Using data from https://raw.githubusercontent.com/nlvcorpus/nlvcorpus.github.io/main/datasets/cars.csv, bar chart of count of models by origin

Dataset schema: {fields: [{field: Model, type: nominal}, {field: MPG, type: quantitative}, {field: Cylinders, type: quantitative}, {field: Displacement, type: quantitative}, {field: Horsepower, type: quantitative}, {field: Weight, type: quantitative}, {field: Acceleration, type: quantitative}, {field: Year, type: quantitative}, {field: Origin, type: nominal}], primaryKey: [Model, Year]}

Specification: {'$schema': 'https://vega.github.io/schema/vega-lite/v4.17.0.json', 'data': {'url': 'https://raw.githubusercontent.com/nlvcorpus/nlvcorpus.github.io/main/datasets/cars.csv'}, 'mark': {'type': 'bar', 'tooltip': None}, 'encoding': {'x': {'field': 'Origin', 'type': 'nominal', 'sort': {'encoding': 'y', 'order': 'descending'}}, 'y': {'aggregate': 'count', 'type': 'quantitative', 'axis': {'title': 'COUNT'}}}}

Task: Using data from https://raw.githubusercontent.com/nlvcorpus/nlvcorpus.github.io/main/datasets/movies.csv,

Count of movies by creative type as a bar chart

Dataset schema: {fields: [{field: Title, type: nominal}, {field: Worldwide Gross, type: quantitative}, {field: Production Budget, type: quantitative}, {field: Release Year, type: quantitative}, {field: Content Rating, type: nominal}, {field: Running Time, type: quantitative}, {field: Major Genre, type: nominal}, {field: Creative Type, type: nominal}, {field: Rotten Tomatoes Rating, type: quantitative}, {field: IMDB Rating, type: quantitative}], primaryKey: Title}

Specification: {'$schema': 'https://vega.github.io/schema/vega-lite/v4.17.0.json',

🟨 Task 🟦 Completed example 🟩 Metadata 🟪 Problem 🟧 Incomplete solution

Figure 3.8. An example of the final model input for an A) in-domain and B) out-of-domain problem.

In-context examples were matched with problems based on semantic similarity. For a given problem the semantic search was implemented using the below steps.

1. The NL query and pool of potential examples were vectorized using one of two pre-trained sentence encoders: Sentence-BERT (paraphrase-MiniLM-L6-v2, embedding size: 384) or a

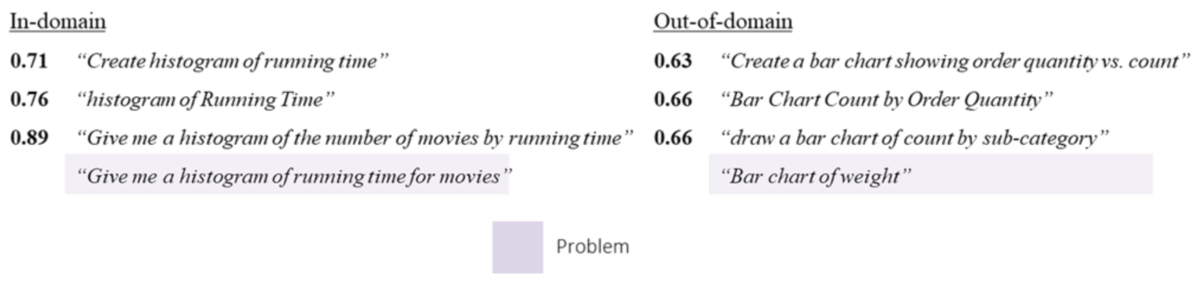
multilingual Universal Sentence Encoder (distiluse-base-multilingual-cased-v1, embedding size: 512) (Reimers, Gurevych, 2019, Yang et al., 2019). The former was used during research objectives one, two and four. The latter was used during research objective three when matching English examples with Spanish problems.

Note: The ‘SentenceTransformers’ Python library (https://www.sbert.net) was used.

2. Cosine similarity between the given NL query and pool of example embeddings was computed.
3. The pool of potential examples was sorted in descending order of cosine similarity score. The highest scoring example was assumed to be the most semantically similar. The top one to three examples were used.

   Note: Lu et al showed the order of examples can have a substantial impact on task performance (Lu et al., 2021). Consequently, the most similar examples immediately preceded a problem.

As an example, Figure 3.9 shows an in-domain and out-of-domain problem from the cdCorpus and the three in-context examples matched to each. For the in-domain problem, all examples relate to the same visualisation: a histogram of running time using the movies dataset. This is a reflection of how the cdCorpus was collected. Specifically, Srinivasan et al asked participants to propose NL commands for the same set of charts. For the out-of-domain problem, all examples relate to a bar chart but a different dataset.



In-domain		Out-of-domain	
0.71	*“Create histogram of running time”*	**0.63**	*“Create a bar chart showing order quantity vs. count”*
0.76	*“histogram of Running Time”*	**0.66**	*“Bar Chart Count by Order Quantity”*
0.89	*“Give me a histogram of the number of movies by running time”*	**0.66**	*“draw a bar chart of count by sub-category”*
	“Give me a histogram of running time for movies”		*“Bar chart of weight”*

▨ Problem

Figure 3.9. An example of the top three examples matched with an in-domain and out-of-domain problem. The scores shown are cosine similarity scores calculated after vectorization using Sentence-Bert (paraphrase-MiniLM-L6-v2).

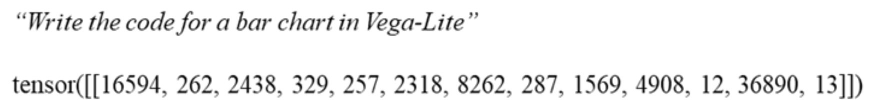
3.4.3 Tokenization

Model inputs need to be tokenized and embedded before they can be passed to the first decoding block of the autoregressive language model. The basic process is as follows:

1. The Hugging Face library provides a pre-trained byte-level byte-pair-encoding (BBPE) tokenizer for both GPT-Neo and GPT-J. Both are pre-trained using the Pile dataset and have a vocabulary size of 50,257 tokens, where tokens represent variable-length byte n-grams (Black et al., 2021, Wang, Komatsuzaki, 2021). The relevant tokenizer was used to split the input

sequence into words (or sub-words). These are automatically mapped to token ids found in the tokenizer's vocabulary.

Note: The key advantage of BBPE tokenizers is any sentence can be broken down into a byte sequence (Wang, C., Cho & Gu, 2019). This means that, after training on a large corpus, they generate fewer unknown tokens when tokenizing an input sequence (Wang, Cho & Gu, 2019). As an example, Figure 3.10 shows the list of token ids generated when tokenizing an input sequence using the GPT-Neo tokenizer.



"Write the code for a bar chart in Vega-Lite"

```
tensor([[16594, 262, 2438, 329, 257, 2318, 8262, 287, 1569, 4908, 12, 36890, 13]])
```

Figure 3.10. The list of token ids generated by the GPT-Neo-125M tokenizer for the phrase "Write the code for a bar chart in Vega-Lite".

2. Embedding is conducted automatically by the HuggingFace AutoModel class used to instantiate pre-trained autoregressive language models.

   Note: Essentially each token id is embedded using matrices learned during pre-training of the language model. An additional matrix is also used to encode the position of each token. The positional encodings and input embeddings are both d_{model}-dimensional vectors (for GPT-Neo-125M, d_{model}= 768). These are summed (and dropout applied) thereby generating a list of d_{model}-dimensional vectors for each token in the input sequence (Rush, Nguyen & Klein, 2018, Vaswani et al., 2017).

3.4.4 Decoding strategy

As already discussed, generative models like GPT-Neo and GPT-J produce output scores (logits) across a vocabulary of tokens and a decoding strategy is then used to select a token. In this way, sequences are generated, one token at a time. A recent study has shown the most probable sequence is not necessarily judged by humans to have the highest quality (Zhang et al., 2020). Regardless, as both these model's vocabulary is ~50,000 tokens long, it's not computationally feasible to identify the sequence with the highest conditional probability or review the relevance of all possible sequences (Wiher, Meister & Cotterell, 2022). Instead, the following sampling strategies were used to approximate the best sequence: Top-K sampling with softmax temperature scaling, and Top-P sampling. The performance of both methods was compared during model development.

Top-K sampling was introduced by Fan et al in 2018. It involves sampling a token from the K most probable tokens at each time step. In practice, this means sampling from the top-K vocabulary which maximises the right hand side of equation 3.1 (Fan, Lewis & Dauphin, 2018). Holtzman et al have demonstrated the potential problems of K being fixed during inference (Holtzman et al., 2019). If the probability distribution is flat, creativity is limited due to appropriate tokens being excluded from the top-K vocabulary. Conversely, the tail of sharp probability distributions may be included leading to the sampling of inappropriate tokens. Softmax temperature scaling via equation 3.3 can mitigate issues with sharp distributions as lowering the temperature (T) skews the probability distribution towards higher probability tokens (Holtzman et al., 2019). In all experiments, softmax temperature scaling was therefore used to redistribute the probability mass prior to sampling. Grid searches were used to optimise K and T.

$$p' = \sum_{x \in V^{(K)}} P(x|x_{1:m-1}) \tag{3.1}$$

$$p(x = V_i|x_{1:m-1}) = \frac{exp(u_i/T)}{\sum_{j=1}^{K} exp\,(u_j/T)} \tag{3.2}$$

Where, p' is the sum of the conditional probabilities of the set of size k, V is the original vocabulary, $V^{(K)}$ is the top-K vocabulary, T is the softmax temperature and $u_{1:|K|}$ are the logits across the top-K vocabulary.

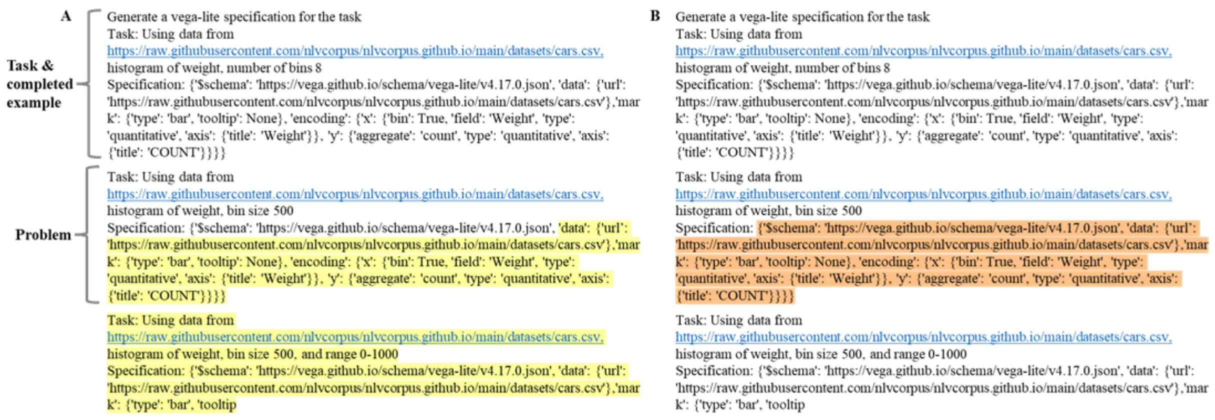
Top-P sampling was introduced by Holtzman et al in 2020 (Holtzman et al., 2019). They demonstrated superior performance to Top-K for long-form text generation. The technique involves sampling from the smallest set of tokens whose cumulative probability exceeds a threshold p. In practice this means finding the smallest set which meets the condition defined below. The primary advantage of this approach is that the subset of tokens will adjust depending on the shape of the probability distribution. The value of P was also optimised via grid search during model development.

$$\sum_{x \in V^{(p)}} P(x|x_{1:i-1}) \geq p \tag{3.4}$$

Where, p is the probability threshold, V is the original vocabulary and $V^{(p)}$ is the top-P vocabulary.

3.4.4 Parsing strategy

The generation of tokens was terminated once the output sequence was 10% longer than the average length of a Vega-Lite specification. Following completion, output sequences were decoded using the appropriate tokenizer. Figure 3.11A shows an example of the output for an in-domain problem (using one shot-learning). The generated output sequence is highlighted in yellow. The completed Vega-Lite specification, which must be extracted, is highlighted in Figure 3.1B.



A

Task & completed example

```
Generate a vega-lite specification for the task
Task: Using data from
https://raw.githubusercontent.com/nlvcorpus/nlvcorpus.github.io/main/datasets/cars.csv,
histogram of weight, number of bins 8
Specification: {'$schema': 'https://vega.github.io/schema/vega-lite/v4.17.0.json', 'data': {'url':
'https://raw.githubusercontent.com/nlvcorpus/nlvcorpus.github.io/main/datasets/cars.csv'},'mar
k': {'type': 'bar', 'tooltip': None}, 'encoding': {'x': {'bin': True, 'field': 'Weight', 'type':
'quantitative', 'axis': {'title': 'Weight'}}, 'y': {'aggregate': 'count', 'type': 'quantitative', 'axis':
{'title': 'COUNT'}}}}
```

Problem

```
Task: Using data from
https://raw.githubusercontent.com/nlvcorpus/nlvcorpus.github.io/main/datasets/cars.csv,
histogram of weight, bin size 500
Specification: {'$schema': 'https://vega.github.io/schema/vega-lite/v4.17.0.json', 'data': {'url':
'https://raw.githubusercontent.com/nlvcorpus/nlvcorpus.github.io/main/datasets/cars.csv'},'mar
k': {'type': 'bar', 'tooltip': None}, 'encoding': {'x': {'bin': True, 'field': 'Weight', 'type':
'quantitative', 'axis': {'title': 'Weight'}}, 'y': {'aggregate': 'count', 'type': 'quantitative', 'axis':
{'title': 'COUNT'}}}}

Task: Using data from
https://raw.githubusercontent.com/nlvcorpus/nlvcorpus.github.io/main/datasets/cars.csv,
histogram of weight, bin size 500, and range 0-1000
Specification: {'$schema': 'https://vega.github.io/schema/vega-lite/v4.17.0.json', 'data': {'url':
'https://raw.githubusercontent.com/nlvcorpus/nlvcorpus.github.io/main/datasets/cars.csv'},'mar
k': {'type': 'bar', 'tooltip
```

B

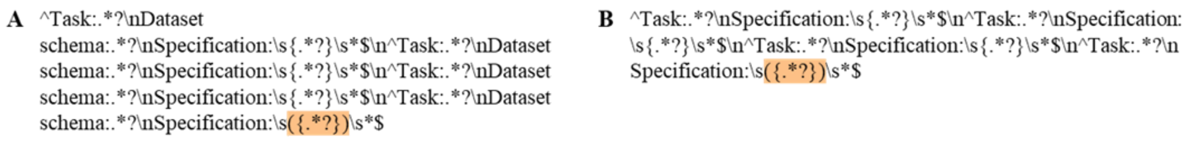
```
Generate a vega-lite specification for the task
Task: Using data from
https://raw.githubusercontent.com/nlvcorpus/nlvcorpus.github.io/main/datasets/cars.csv,
histogram of weight, number of bins 8
Specification: {'$schema': 'https://vega.github.io/schema/vega-lite/v4.17.0.json', 'data': {'url':
'https://raw.githubusercontent.com/nlvcorpus/nlvcorpus.github.io/main/datasets/cars.csv'},'mar
k': {'type': 'bar', 'tooltip': None}, 'encoding': {'x': {'bin': True, 'field': 'Weight', 'type':
'quantitative', 'axis': {'title': 'Weight'}}, 'y': {'aggregate': 'count', 'type': 'quantitative', 'axis':
{'title': 'COUNT'}}}}

Task: Using data from
https://raw.githubusercontent.com/nlvcorpus/nlvcorpus.github.io/main/datasets/cars.csv,
histogram of weight, bin size 500
Specification: {'$schema': 'https://vega.github.io/schema/vega-lite/v4.17.0.json', 'data': {'url':
'https://raw.githubusercontent.com/nlvcorpus/nlvcorpus.github.io/main/datasets/cars.csv'},'mar
k': {'type': 'bar', 'tooltip': None}, 'encoding': {'x': {'bin': True, 'field': 'Weight', 'type':
'quantitative', 'axis': {'title': 'Weight'}}, 'y': {'aggregate': 'count', 'type': 'quantitative', 'axis':
{'title': 'COUNT'}}}}

Task: Using data from
https://raw.githubusercontent.com/nlvcorpus/nlvcorpus.github.io/main/datasets/cars.csv,
histogram of weight, bin size 500, and range 0-1000
Specification: {'$schema': 'https://vega.github.io/schema/vega-lite/v4.17.0.json', 'data': {'url':
'https://raw.githubusercontent.com/nlvcorpus/nlvcorpus.github.io/main/datasets/cars.csv'},'mar
k': {'type': 'bar', 'tooltip
```

Figure 3.11 Example of the output generated by GPT-Neo-125M for an in-domain problem when using one completed example (one-shot learning). In panel A, the output sequence is highlighted in yellow. In panel B, the completed Vega-Lite specification is highlighted in orange.

The standard Python re module (https://docs.python.org/3/library/re.html) was used to capture completed Vega-Lite specifications. The regular expression patterns shown in Figure 3.12 were utilised. If no match was found, an attribute error was raised.



A

```
^Task:.*?\nDataset
schema:.*?\nSpecification:\s{.*?}\s*$\n^Task:.*?\nDataset
schema:.*?\nSpecification:\s{.*?}\s*$\n^Task:.*?\nDataset
schema:.*?\nSpecification:\s{.*?}\s*$\n^Task:.*?\nDataset
schema:.*?\nSpecification:\s({.*?})\s*$
```

B

```
^Task:.*?\nSpecification:\s{.*?}\s*$\n^Task:.*?\nSpecification:
\s{.*?}\s*$\n^Task:.*?\nSpecification:\s{.*?}\s*$\n^Task:.*?\n
Specification:\s({.*?})\s*$
```

Figure 3.12 The regular expression patterns used to capture completed Vega-Lite specifications for A) out-of-domain problems and B) in-domain problems. The capturing group is highlighted in orange.

3.5 Model evaluation

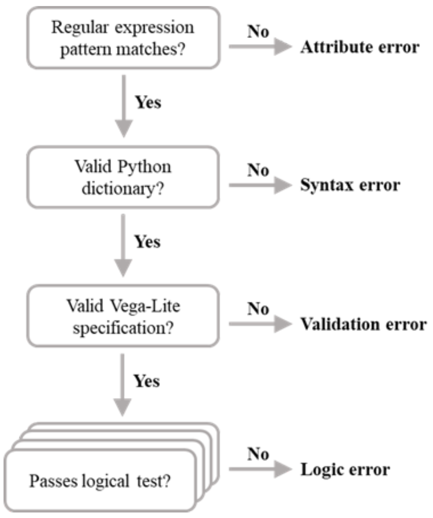
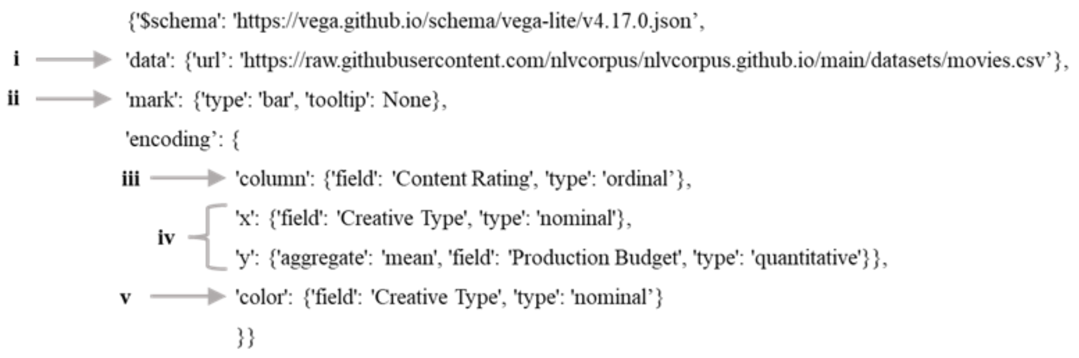


Figure 3.13 The workflow of unit testing used to assess model outputs.

A series of unit tests were used to assess model outputs. When a unit test failed, a unique error was raised and no further tests were conducted. Figure 3.13 depicts the workflow. The first test involved capturing the completed Vega-Lite specification using the relevant regular expression pattern. As described in section 3.4.4, an attribute error was raised when no match was found. Next, the ast module (https://docs.python.org/3/library/ast.html) was used to check whether parsed outputs were valid Python dictionaries. An invalid dictionary raised a syntax error. Afterwards, the jsonschema module (https://python-jsonschema.readthedocs.io/) was used to check compliance with the Vega-Lite schema. A validation error was raised when an invalid instance was encountered. Finally, a set of match based logical tests were used to assess the relevance of valid Vega-Lite specifications. These involved comparing a generated specification to the original (NL command, Vega-Lite specification) pair found in the corpus. A logic error was raised when any one of these tests failed.

A well known limitation of match based metrics is that they can classify functionally equivalent code as incorrect (Chen et al., 2021, Ren et al., 2020). Consequently, logical tests were limited to distinct chart properties. These were the data source, mark type, positional encodings (x/y axis field encodings and associated aggregation functions), facet channel (row/column field encodings), mark channel (mark colour field encodings and associated aggregation functions) and the detail channel (i.e. fields used to aggregate underlying data). Figure 3.14 shows the relevant sections of a Vega-Lite specification. It should be noted the detail channel was only relevant for specifications associated with tables, which made up less than 1% of all problems (Table 3.4). Additionally, x/y axis encodings could be used interchangeably.



```
          {'$schema': 'https://vega.github.io/schema/vega-lite/v4.17.0.json',
i   ───▶  'data': {'url': 'https://raw.githubusercontent.com/nlvcorpus/nlvcorpus.github.io/main/datasets/movies.csv'},
ii  ───▶  'mark': {'type': 'bar', 'tooltip': None},
          'encoding': {
          iii ───▶ 'column': {'field': 'Content Rating', 'type': 'ordinal'},
                   ┌ 'x': {'field': 'Creative Type', 'type': 'nominal'},
               iv ─┤
                   └ 'y': {'aggregate': 'mean', 'field': 'Production Budget', 'type': 'quantitative'}},
          v   ───▶ 'color': {'field': 'Creative Type', 'type': 'nominal'}
                   }}
```

Figure 3.14 The Vega-Lite specification properties tested during logical unit tests: (i) the URL from which to load the underpinning data, (ii) the type of mark used to visually encode the data, (iii) the row/column encodings used when plotting faceted charts, (iv) the positional encodings which determine mark position and (v) the colour encodings which determine mark colour.

Two evaluation metrics were crucial to the assessment of model performance: valid@k and pass@k. Valid@k represents the validation rate with k samples. For example, valid@1 would represent the fraction of problems of which the model generated a valid Vega-Lite specification in 1 sample. Similarly, pass@k represents the pass rate with k samples. Where pass@1 would represent the fraction of problems of which the model generated a Vega-Lite specification that passed all logical tests. Similar to recent studies, model performance was also assessed when k was greater than one (Li et al., 2022, Chen et al., 2021, Austin et al., 2021). For instance, pass@10 represents the proportion of problems in which the model generated at least one solution that passed all unit tests (with 10 independent samples per problem).

3.6 Research design

The overall research design was based on the four research objectives as they defined the expected outcomes of this project.

3.6.1 Research objective 1

The primary output was a model capable of translating NL queries to Vega-Lite visualisation specifications. Experiments were designed to test the below investigative hypothesis.

"Vega-Lite visualisations can be generated from natural language statements by using a pre-trained GPT-Neo-125M model in an end-to-end, modular model."

The smallest GPT-Neo model (~500 mb) was utilised as this lessened resource requirements thereby simplifying development. The key success criterion was comparable performance to NL4DV, which was applied using its default settings and the Spacy English language dependency parser (en_core_web_sm) (Narechania, Srinivasan & Stasko, 2021). All experiments were conducted using (NL query, Vega-Lite specification) pairs from the cdCorpus as NL4DV has been successfully applied to this data in the past (Srinivasan et al., 2021, Narechania, Srinivasan & Stasko, 2021). For GPT-J-6B, the scope was restricted to in-domain problems as this represents the simplest use case. GPT-Neo-125M hyperparameter optimisation was conducted using ten-fold cross validation hyperparameter grid searches on the training set (see section 3.2.3). When testing problems in a validation set, in-context examples were drawn from the remaining nine sets. This approach enabled optimisation of the prompt design (number of in-context examples) and decoding strategy (Top-P: threshold p, Top-K: threshold k & softmax temperature). The test set was used for final evaluation where problems were matched with in-context examples from the training set (see section 3.2.3). Table 3.7 shows the median cosine similarity scores. The below null hypothesis was then evaluated at a 95% confidence level using McNemar's test (Dietterich, 1998). Please note NL4DV does not receive in-context examples as an input. It is designed for use with NL queries alone and therefore does not distinguish between in-domain/out-of-domain versions of a test set.

"There is no significant difference in pass@1 between GPT-Neo-125M and NL4DV when applied to in-domain problems from the cdCorpus"

Cosine similarity	**1**	**2**	**3**
Median	0.85	0.81	0.79
IQR	0.11	0.11	0.12

Table 3.7 The median cosine similarity for the top three examples associated with each problem in the in-domain test set. The IQR (interquartile range) is also given.

3.6.2 Research objective 2

The primary output was a comparative analysis of performance for different sized pre-trained transformer-based autoregressive language models. Experiments were designed to test the below investigative hypothesis.

"Scaling up the size of pre-trained transformed-based models will improve Vega-Lite visualisation generation performance."

The performance of the following models was compared: GPT-Neo-125M, GPT-Neo-1.3B, GPT-Neo-2.7B and GPT-J-6B. The scope was restricted to the cdCorpus. Hyperparameter optimisation was conducted using ten-fold cross validation hyperparameter grid searches on an in-domain and out-of-domain version of the training set (see section 3.2.3). When testing problems in a validation set, in-context examples were drawn from the remaining nine sets. Both prompt design (number of in-context examples) and decoding strategy (Top-P: threshold p) were optimised in this way. The smallest (GPT-Neo-125M) and largest model (GPT-J-6B) were then evaluated using an in-domain and out-of-domain version of the test set (see section 3.2.3). Test set problems were matched with in-context examples from the training set. Table 3.8 shows the median cosine similarity scores. The below null hypothesis was then evaluated at a 95% confidence level using McNemar's test (Dietterich, 1998).

"There is no significant difference in pass@1 between GPT-Neo-125M and GPT-J-6B when applied to in-domain/out-of-domain problems from the cdCorpus"

Cosine similarity	**In-domain**			**Out-of-domain**		
	1	**2**	**3**	**1**	**2**	**3**
Median	0.85	0.81	0.79	0.52	0.49	0.47
IQR	0.11	0.11	0.12	0.15	0.15	0.15

Table 3.8 The median cosine similarity for the top three examples associated with each problem in the out-of-domain and in-domain test set. The IQR (interquartile range) is also given.

3.6.3 Research objective 3

The primary output was an assessment of the most powerful pre-trained autoregressive model on Spanish NL queries. The choice of Spanish was predicated on the assumption that as it is widely spoken (~480 million native speakers (Ardila, 2020)) all models would have been exposed to the language during pre-training via the European parliament and YouTube subtitles datasets (see section 3.3). Experiments were designed to test the below investigative hypothesis.

"Vega-Lite visualisations can be generated from Spanish natural language commands using pre-trained transformer-based autoregressive language models."

The most powerful model and hyperparameter configuration (as per research objective 2) was tested. The key success criterion was superior performance to NL4DV. The scope was restricted to the cdCorpus. An in-domain version of the test set was tested (see section 3.2.3). Google Translate was used to translate problems to Spanish prior to being matched with English in-context examples from the training set (see section 3.4.2).Table 3.9 shows the median cosine similarity scores.

Cosine similarity	**1**	**2**	**3**
Median	0.76	0.71	0.67
IQR	0.15	0.15	0.15

Table 3.9 The median cosine similarity for the top three examples associated with each problem in the test set. The IQR (interquartile range) is also given.

NL4DV was applied using its default settings and the Spacy English dependency parser (en_core_web_sm). All attempts to instantiate it using one of Spacy's Spanish language dependency parsers resulted in an error. It is assumed this is not currently possible as no reference is made to non-English languages in the original journal article or supporting documentation (Narechania, Srinivasan & Stasko, 2021). To evaluate this assertion, a small random sample of problems were tested using NL4DV. This enabled a preliminary assessment of the below null hypothesis. Please note NL4DV only receives NL queries from the test set (i.e. not any in-context examples). In contrast to a pre-trained autoregressive model, it therefore doesn't distinguish between in-domain/out-of-domain versions of a test set.

"There is no significant difference in pass@1 between GPT-J-6B and NL4DV when applied to the Spanish in-domain problems from the cdCorpus"

3.6.4 Research objective 4

The primary output was an assessment of the most powerful pre-trained autoregressive model on NL queries related to HE data. Experiments were designed to test the below investigative hypothesis.

"Vega-Lite visualisations related to Higher Education data can be generated from natural language commands using pre-trained transformer-based autoregressive language models."

The most powerful model and hyperparameter configuration was chosen based on results for research objective 2. The key success criterion was comparable performance to NL4DV, which was applied using its default settings and the Spacy English dependency parser (en_core_web_sm). Evaluation was conducted using the entire heCorpus (see section 3.2.2). All problems were matched with in-context examples from the cdCorpus training set thereby creating an out-of-domain test set. Table 3.10 shows the median cosine similarity scores. The below null hypothesis was then evaluated at a 95% confidence level using McNemar's test (Dietterich, 1998). As already explained, NL4DV only receives NL queries from the test set so it doesn't distinguish between in-domain/out-of-domain versions of a test set.

"There is no significant difference in pass@1 between GPT-J-6B and NL4DV when applied to out-of-domain problems from the heCorpus"

Cosine similarity	1	2	3
Median	0.48	0.46	0.45
IQR	0.13	0.13	0.13

Table 3.10 The median cosine similarity for the top three examples associated with each problem in the test set. The IQR (interquartile range) is given.

3.7 Conclusion

In this chapter, the design and implementation of a novel NL2VIS model that utilises a pre-trained autoregressive model has been described. This will be tested using two corpuses: cdCorpus and heCorpus. The former has been published previously so will facilitate benchmarking. The latter was collected as part of this research project and relates to HE data. The approach to matching in-context examples to problems based on semantic similarity was also explained. Additionally, the research design for each objective was outlined.

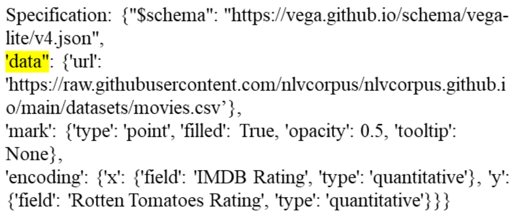
Chapter 4: Results

Synopsis: This chapter provides an overview of key experimental results collected when investigating each research objective.

4.1 Research objective 1: Design and create an end-to-end model which will generate Vega-Lite visualisations from NL queries for a given dataset

4.1.1 Prompt design

The efficacy of the basic prompt design was assessed by observing error rates during hyperparameter optimisation. It was noticed syntax errors were the predominant source of error when using in-context examples. For example, Figures 4.2A to 4.2C show the syntax error rates generated using one to three in-context examples (threshold k: 10-100 with a step size of 10, softmax temperature: 0.1-1 with a step size of 0.1). The average syntax error rate across the entire hyperparameter space was 73.6 ± 8.7 %, 65.1 ± 5.3 % and 52.9 ± 1.7 % for 1-shot, 2-shot and 3-shot learning, respectively. The average syntax error rate clearly decreases as the number of in-context examples increases. However, it was important to identify the source of error as the number of examples cannot be increased indefinitely. As a syntax error was raised when a generated specification wasn’t a valid Python dictionary, model outputs were visually inspected. This revealed inconsistent types of opening and closing quotation marks (Figure 4.1). By subsequently standardising the type of quotation mark across model inputs, the syntax error rate was substantially lowered across the entire hyperparameter space (Figures 4.2D-4.2F). For instance, the average syntax error rate when using a softmax temperature below 0.5 was close to zero (1-shot: 1.2 ± 0.3 %, 2-shot: 1.0 ± 0.6 %, 3-shot: 1.3 ± 0.3). Interestingly, syntax error rates have a uniform distribution with respect to threshold k, and a positive correlation above ~0.5 with respect to the softmax temperature. This suggests syntax errors increase in frequency as the likelihood of sampling a high probability token decreases (Holtzman et al., 2019).



```
Specification: {"$schema": "https://vega.github.io/schema/vega-
lite/v4.json",
'data": {'url':
'https://raw.githubusercontent.com/nlvcorpus/nlvcorpus.github.i
o/main/datasets/movies.csv’},
'mark': {'type': 'point', 'filled': True, 'opacity': 0.5, 'tooltip':
None},
'encoding': {'x': {'field': 'IMDB Rating', 'type': 'quantitative'}, 'y':
{'field': 'Rotten Tomatoes Rating', 'type': 'quantitative'}}}
```

Figure 4.1 This specification raises a syntax error due to the opening and closing quotation marks differing in type (highlighted in yellow).

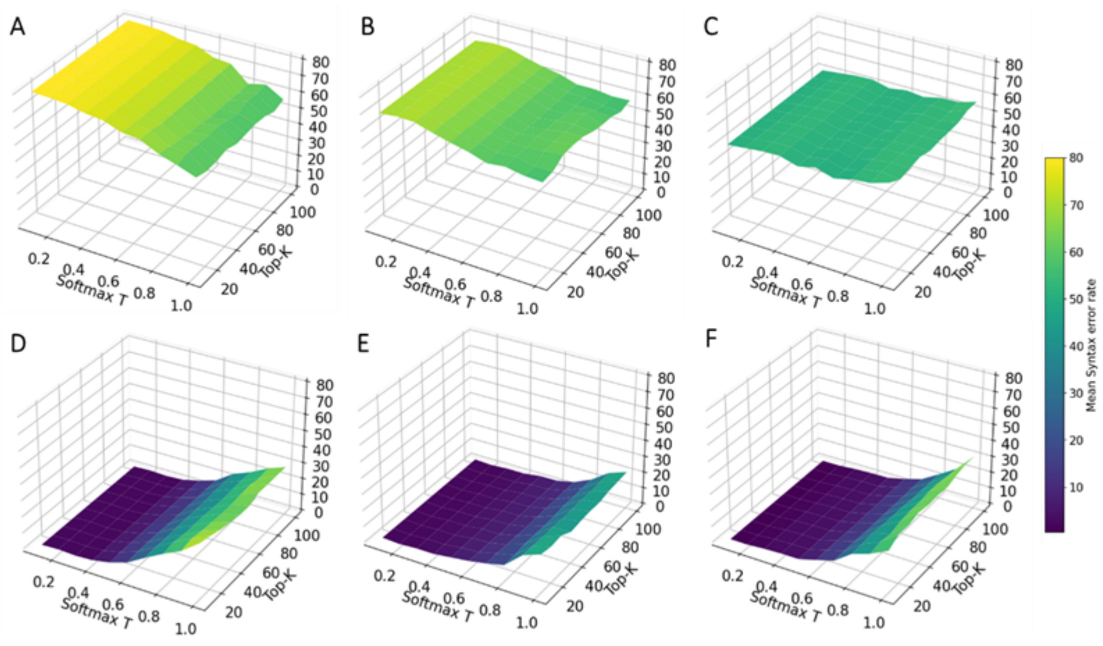


Figure 4.2 Average syntax error rate (%) generated by 10-fold cross validation hyperparameter grid searches before (A-C) and after (D-F) standardisation of quotation marks. All results were collected with GPT-Neo-125M using the Top-K decoding strategy and fully-specified in-domain problems. Panels A and D, B and E, and C and F correspond to 1-shot, 2-shot and 3-shot learning respectively.

4.1.2 Decoding strategy

No meaningful difference in performance was observed between Top-K and Top-P decoding over a broad hyperparameter space. Top-K decoding was tested with k thresholds from 10 to 100 (step size: 10) and softmax temperatures from 0.1 to 1 (step size: 0.1). Whilst, Top-P sampling was tested using p thresholds from 0.3 to 1 (step size: 0.05). Figure 4.3 shows valid@1 as a function of these hyperparameters when applied to fully-specified problems using 3-shot learning. Where valid@1 is the average proportion of problems of which the model generated a valid Vega-Lite specification in one sample (see section 3.5). The behaviour shown indicates performance is superior when tokens with the highest conditional probability are favoured (Holtzman et al., 2019). For instance, for Top-K sampling, valid@1 is highest at low softmax temperatures. Similarly, for Top-P sampling, valid@1 is highest when the threshold p is lowered. The relationship between hyperparameters and performance will be described in more detail in the next section. Crucially, the maximum valid@1 exhibited by both techniques overlap (Top-K: 95.3 ± 3.2 % when k = 50 and softmax temperature = 0.1, Top-P: 95.8 ± 2.9 % where p = 0.3). It was decided the Top-P strategy would be used in all further experiments as only one hyperparameter requires tuning (i.e. threshold p), which lowers the time taken for optimisation. This was beneficial as City’s HPC queue system prioritises shorter jobs.

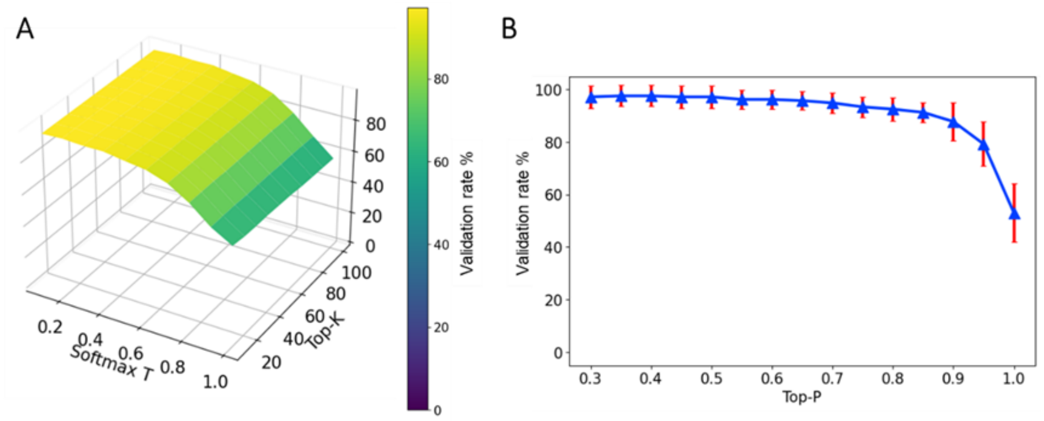


Figure 4.3 Average fraction of problems where the model generates a specification with a valid Vega-Lite schema using A) Top-K and B) Top-P decoding. All grid search results were collected with GPT-Neo-125M for fully-specified, in-domain problems using 3-shot learning.

4.1.3 Hyperparameter optimisation

The number of in-context examples n and threshold p were optimised using hyperparameter grid searches on both fully- and under-specified in-domain problems. The key evaluation metric was pass@1 (see section 3.5). Figures 4.4A and 4.4B present results for fully-specified problems (p: 0.3-1 with a step size of 0.05, n: 0-3 with a step size of 1) and under-specified problems (p: 0.5-1 with a step size of 0.05, n: 0-3 with a step size of 1), respectively. Similar trends are observed for both types of problem. Firstly, no solutions pass when n is 0. Furthermore, only minor increases in pass@1 occur as n increases above one. Secondly, the relationship between threshold p and pass@1 is non-linear with an approximately constant and maximum pass rate observed when p is below 0.6. Finally, performance for fully- and under-specified problems is similar at low values of p. For instance, pass@1 is 72.5 ± 12.0% for fully-specified problems and 70.1 ± 8.2% for under-specified problems when p is 0.5. There is also a noticeable decrease in the error for under-specified problems. However, this is due to a difference in training set size. The original corpus contains more under-specified problems, as a result the under-specified training data is larger in size (see section 3.2.3) and standard deviation decreases as sample size increases (Freedman, Pisani & Purves,).

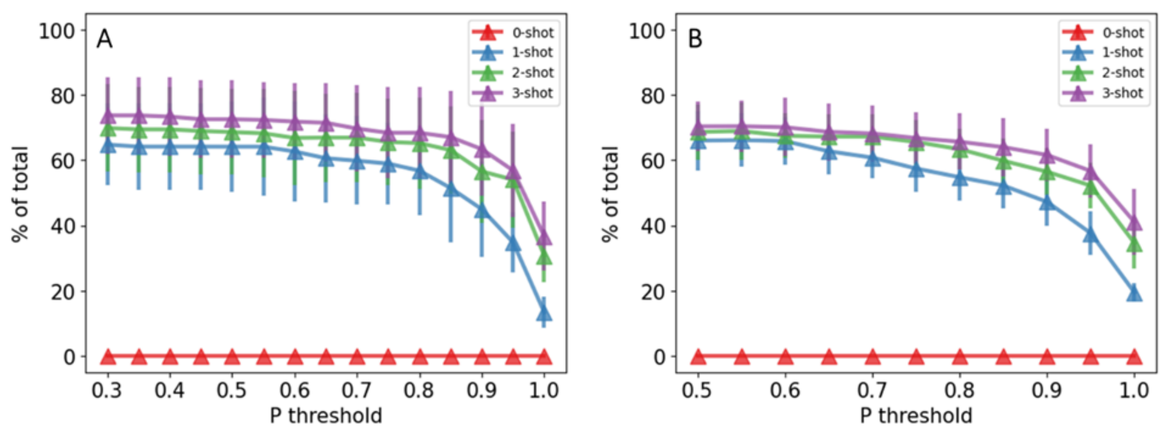


Figure 4.4 Pass@1 generated by 10-fold cross validation hyperparameter grid searches for A) fully-specified and B) under-specified problems. All results were collected using GPT-Neo-125M for in-domain problems.

A paired student t-test was used to test the assertion that performance is not significantly approved when p is below 0.6. More specifically, the comparative performance when p was set to 0.6 and 0.5 was assessed. The resulting p-value was 0.43 for fully-specified problems and 0.97 for under-specified problems when n was 3 (Table 4.1). At a 95% confidence level, the null hypothesis cannot therefore be rejected (i.e. that there is no difference in performance). It should be noted this test is reported to bias rejecting the null hypothesis when it is applied to k-fold cross-validation results, which further suggests the absence of significant performance improvements (Dietterich, 1998).

Type	**Pass@1 (P=0.6) / %**	**Pass@1 (P=0.5) / %**	**p-value**
Fully-specified	71.8 ± 11.6	72.5 ± 12.0	0.43
Under-specified	70.1 ± 8.9	70.1 ± 8.2	0.97

Table 4.1 Probability (p-value) that differences in performance between p thresholds of 0.5 and 0.6 are statistically significant. All results were collected using GPT-Neo-125M and in-domain problems with 3-shot learning. The p-values were calculated using a paired student t-test.

Before choosing the optimal value of n, its effect on performance and corresponding error rates at low values of p was studied in greater detail (Figure 4.5). As already discussed, pass@1 was zero, irrespective of p, when n was zero. In this scenario, all model outputs led to an attribute, syntax or validation error. For example, when the threshold p is set to 0.3, the proportion of attribute, syntax and validation errors for fully-specified problems was 59.9 ± 7.7%, 7.9 ± 6.4% and 32.2 ± 7.7%, respectively (Figure 4.5B). This reliance on in-context examples is unsurprising as GPT-Neo-125M's pre-training objective was not focused on Vega-Lite specification generation and task specific fine tuning hasn't been conducted. Figures 4.5A and 4.5C show pass@1 increases substantially when n is 1 (fully-specified: 64.8 ± 12.5% when P=0.3, under-specified: 70.4 ± 7.7% when P=0.5). Figures 4.5B and 4.5D show the rate of attribute, syntax and validation errors all decrease simultaneously to less than 5%. The concurrent increase in the logic error rate is because not all generated specifications pass the unit tests.

Prior studies have shown a positive correlation between the number of in-context examples and performance at text generation tasks (Brown et al., 2020). However, when increasing n from 1 to 3, there was only a weak positive and negative correlation with pass@1 and the logic error rate, respectively (Figure 4.5). Given the overlap of the error bars, it is unlikely any differences in performance are statistically significant. This may be a consequence of GPT-Neo's sliding attention window meaning it can only attend to the previous 256 tokens (see section 3.3). Despite only marginal improvements, it was decided setting n to 3 and threshold p to 0.3 would be sufficiently close to the optimum. Crucially, Figure 4.5 shows the model is stable under these conditions.

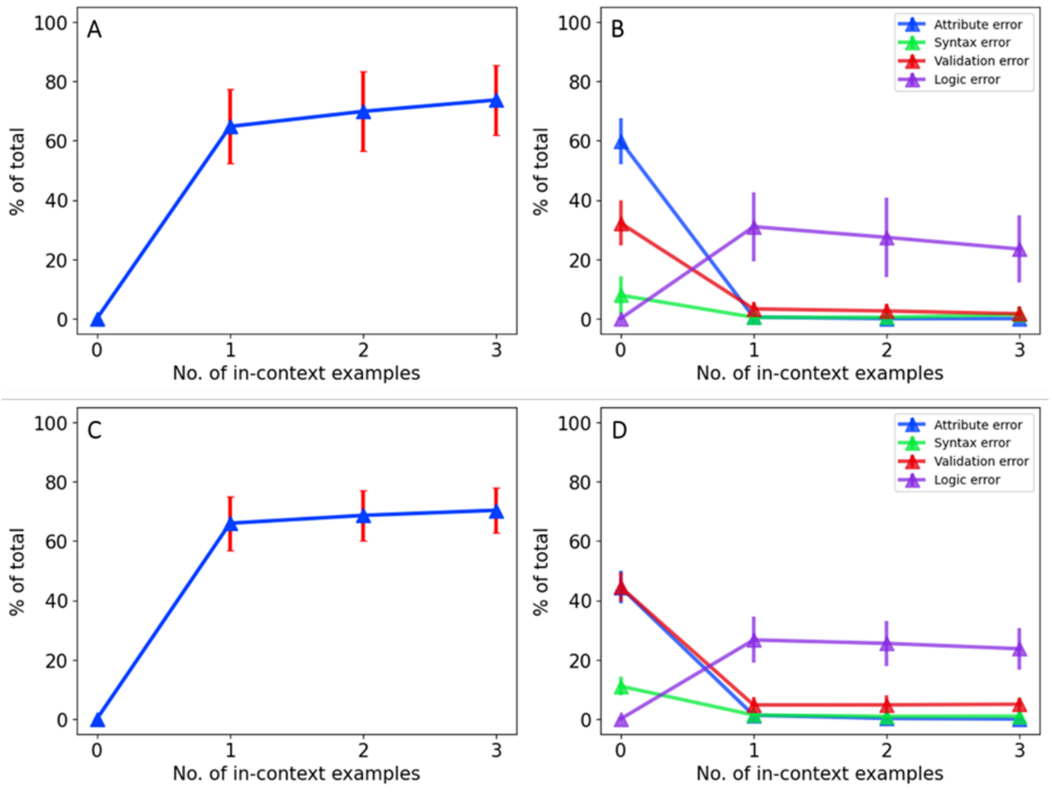


Figure 4.5 Average pass rates (pass@1) and error rates as a function of number of in-context examples for fully-specified (threshold P: 0.3) and under-specified (threshold P: 0.5) problems. Panels A and B, and C and D correspond to fully- and under-specified problems, respectively. All results were collected with GPT-Neo-125M using 10-fold cross validation and in-domain problems.

4.1.4 Model evaluation

GPT-Neo-125M (threshold p: 0.3, n: 3) and NL4DV both generated relevant visualisations, which passed all unit tests, when applied to the in-domain test set. Figure 4.6 shows an example of a visualisation generated by GPT-Neo-125M. A further ten examples, which passed all unit tests, can be found in Appendix I.

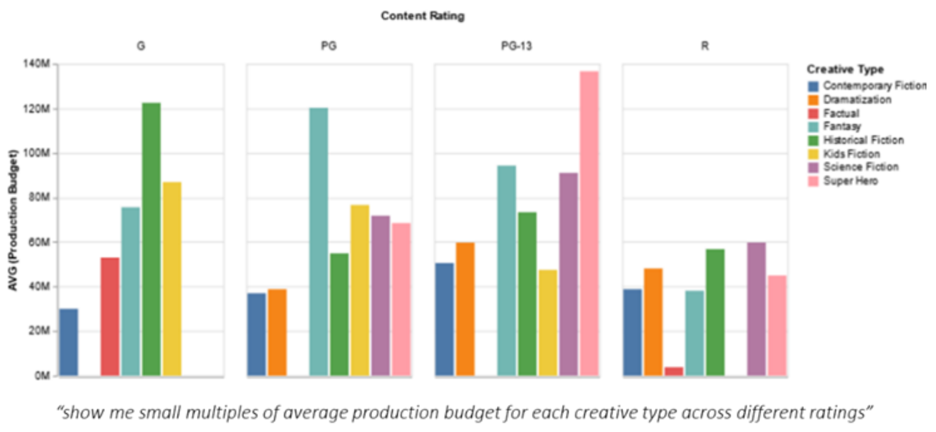


Figure 4.6 A visualisation created by GPT-Neo-125M using a p threshold of 0.3 and three in-context examples (i.e. n =3).

Table 4.2 summarises test results. In terms of the pass@1 metric, GPT-Neo-125M's performance was superior to NL4DV. Out of the 160 problems in the test set, it generated 117 specifications which passed all unit tests (pass@1: 73.1%). Whilst NL4DV generated 70 solutions which passed all unit tests (pass@1: 43.8%). McNemar's test was used to assess the significance of this difference in performance. This test was chosen as Type I errors (i.e. rejecting the null hypothesis when it's actually true) are reported to be infrequent when comparing two models with one test set (Dietterich, 1998). The null hypothesis was as follows: "*There is no significant difference in pass@1 between GPT-Neo-125M and NL4DV when applied to in-domain problems from the cdCorpus*" (see section 3.6.1). The test returned a p-value of 6.1×10^{-8} (McNemar's statistic: 15.0), which means this hypothesis can be rejected at the 95% confidence level.

Model	**Attribute error / %**	**Syntax error / %**	**Validation error / %**	**Logic error / %**	**Pass@1 / %**
GPT-Neo	0.0	1.3	3.1	22.5	73.1
NL4DV	1.3	0.0	0.0	55.0	43.8

Table 4.2 A percentage breakdown of model outputs (when one output is generated per problem) for GPT-Neo-125M and NL4DV. All GPT-Neo-125M results were collected with a threshold P of 0.3 and three in-context examples (i.e. n=3).

Figure 4.7 provides a percentage breakdown of GPT-Neo-125M and NL4DV outputs. This shows the reason for GPT-Neo-125M's superior performance was a lower logic error rate (GPT-Neo-125M: 22.5%, NL4DV: 55.0%). It should be noted that NL4DV raised no syntax or validation errors. This is because its rule based architecture ensures outputs comply with the Python dictionary syntax and Vega-Lite schema. Nevertheless, syntax and validation errors only accounted for 4.4% of GPT-Neo-125M's failed outputs.

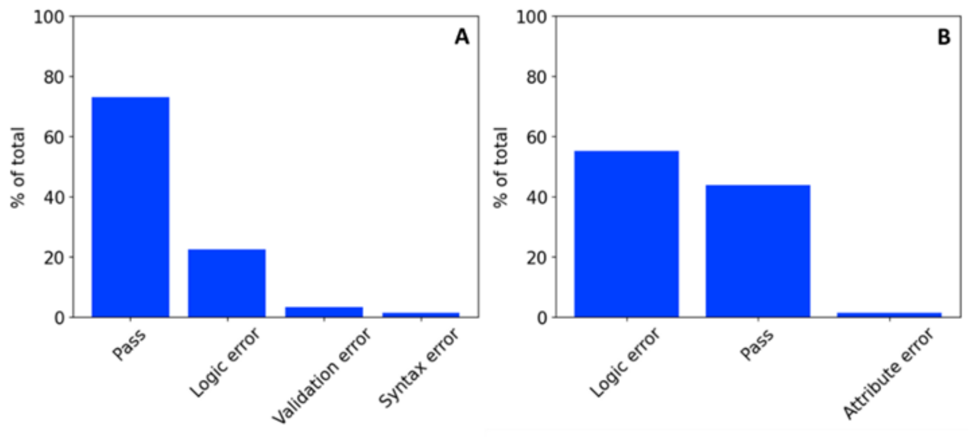


Figure 4.7 A percentage breakdown of model outputs (when one output is generated per problem) for A) GPT-Neo-125M and B) NL4DV. All GPT-Neo-125M results were collected with a threshold P of 0.3 and three in-context examples (i.e. n=3).

Figure 4.8 shows the proportion of problems which failed each logical test. The major source of error (in descending order for GPT-Neo) were incorrect encodings in the position (GPT-Neo-125M: 14.4% , NL4DV: 40.6%), mark (GPT-Neo-125M: 11.3% , NL4DV: 15.0%) and facet (GPT-Neo-125M: 7.5% , NL4DV: 26.9%) channels. The latter two suggest both models struggle with requests related to more complex requests where multiple charts need to be arranged in a grid or mark colour needs to be encoded using a variable.

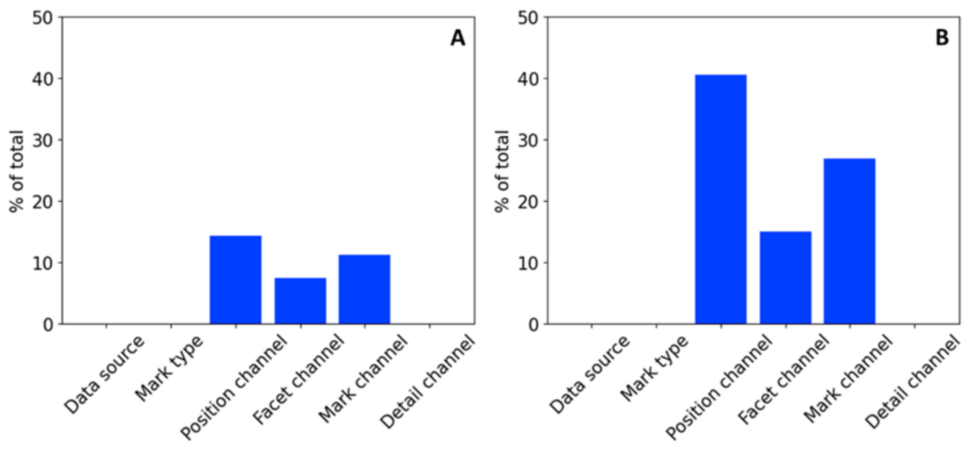


Figure 4.8 The proportion of problems which failed each logical test when one output is generated per problem for A) GPT-Neo-125M and B) NL4DV. All GPT-Neo-125M results were collected with a threshold P of 0.3 and three in-context examples (i.e. n=3).

Performance was broken down by target mark type (Table 4.3). Interestingly, GPT-Neo-125M did not generate a valid area chart (mark type: area) or strip plot (mark type: tick), which may be due to the low number in the training data from which in-context examples were drawn (see section 3.2.3). NL4DV successfully generated both types of charts. The respective subpopulations are too small to draw conclusions. However, this may indicate NL4DV performs better when limited training data is available.

Mark type	Count	GPT-Neo-125M	NL4DV
Area	1	0.0 %	100.0 %
Bar	83	69.9 %	34.9 %
Line	25	80.0 %	24.0 %
Point	50	78.0 %	66.0 %
Tick	1	0.0 %	100.0 %

Table 4.3 Pass@1 by target chart mark type for GPT-Neo-125M and NL4DV. All GPT-Neo-125M results were collected with a threshold P of 0.3 and three in-context examples (i.e. n=3).

Generating multiple solutions per problem did not improve the performance of either model. Table 4.4 shows the final pass rate exhibited by GPT-Neo-125M was the same when sampling one answer (pass@1: 73.1%) and ten answers (pass@10: 73.1%). This might be due to the low value of threshold p reducing the variance in generated solutions (Holtzman et al., 2019). The number of outputs generated by NL4DV cannot be specified and is inconsistent across problems. However, when sampling all available outputs (a maximum of 11 per problem), no improvement was also observed (Table 4.4).

Model	pass@1	pass@k
GPT-Neo-125M	73.1	73.1
NL4DV	43.8	45.0

Table 4.4 Pass@1 and pass@k for GPT-Neo-125M and NL4DV. Where k was 10 for GPT-Neo-125M. For NL4DV, no fixed value of k can be set. However, a maximum of 11 outputs was observed. GPT-Neo-125M results were collected with a threshold P of 0.3 and three in-context examples (i.e. n=3).

4.2 Research objective 2: Compare model performance when using different pre-trained transformer-based autoregressive models

4.2.1 Hyperparameter optimisation: In-domain problems

Similar to the first research objective, exhaustive grid searches were used to assess performance as a function of the probability threshold p and number of in-context examples n. Figure 4.9 shows results for each model with fully- and under-specified problems. All models display similar behaviour to GPT-Neo-125M across the hyperparameter space. Specifically, when using one or more in-context examples (n≥ 1), an inverse non-linear relationship between pass@1 and p is exhibited with performance approaching a maximum at low values of p. No substantial differences in performance for fully- and under-specified problems are observed for larger models. For instance, GPT-J-6B produced a pass@1 of 79.4 ± 9.2% and 75.8 ± 4.7% at a p of 0.5 for fully- and under-specified problems, respectively.

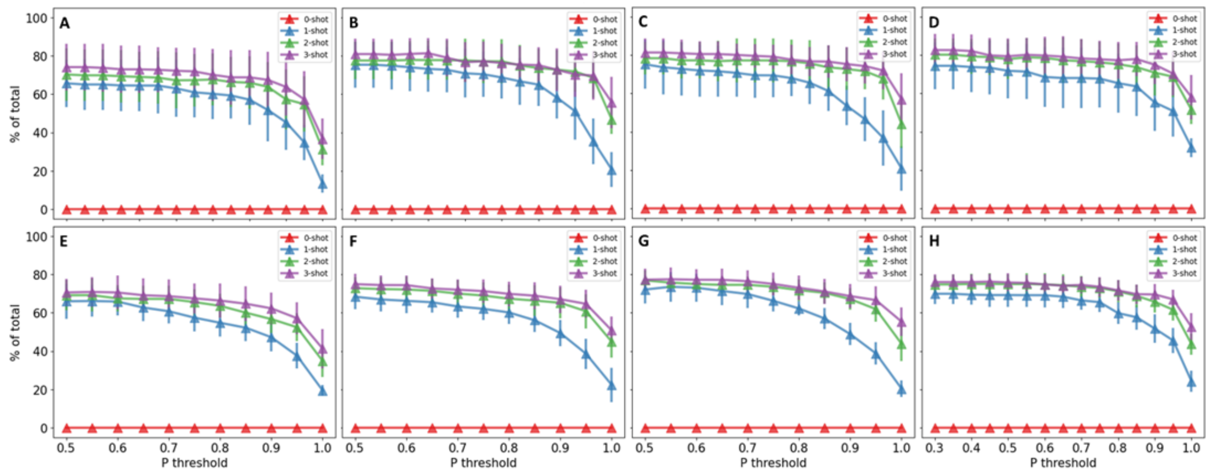


Figure 4.9 Pass@1 as a function of threshold p and number of in-context examples (n). All results relate to in-domain problems (fully-specified: panels A to D, under-specified: panels E to H). Panels A and E correspond to GPT-Neo-125M. Whilst, panels B and G, panels C and G, and panels D and H relate to GPT-Neo-1.3B, GPT-Neo-2.7B and GPT-J-6B, respectively.

In section 4.1.3, it was shown that under optimal conditions (i.e. low values of p), GPT-Neo-125M only exhibits minor improvements in performance as n is increased above one. Similar behaviour was seen for the larger GPT-Neo models, which was expected as the size of the sliding attention window remains fixed (256 tokens). Surprisingly, Figure 4.11 this was also observed for GPT-J-6B despite its far larger attention window (2,048 tokens). This suggests the most semantically similar example is sufficient for the majority of problems. This makes sense as in-domain examples often correspond to the same visualisation (see section 3.4.2).

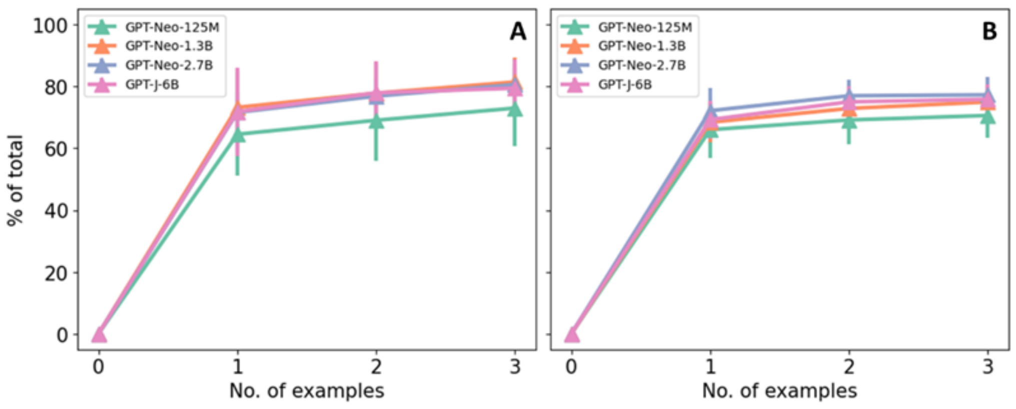


Figure 4.11 Pass@1 as a function of the number of in-context examples (n) and model type. All results were collected using a threshold P of 0.5 and relate to in-domain A) fully-specified and B) under-specified problems.

Figure 4.12 shows pass@1 as a function of model size for fully- and under-specified problems. On average, the worst performing model for both was GPT-Neo-125M. However, performance appears to plateau above 1.3 billion and 2.7 billion parameters for fully- and under-specified problems, respectively. Calculating Spearman's rank correlation coefficients confirmed a weak positive monotonic association (fully-specified problems: 0.15, under-specified problems: 0.30) between number of parameters and pass@1. However, the corresponding p-values (fully-specified problems: 0.35, under-specified problems: 0.06) did not enable rejection of the null hypothesis at a 95% confidence level. Where the null hypothesis was as follows: "*There is no significant difference in pass@1 between GPT-Neo-125M and GPT-J-6B when applied to in-domain problems from the cdCorpus"* (see section 3.6.2).

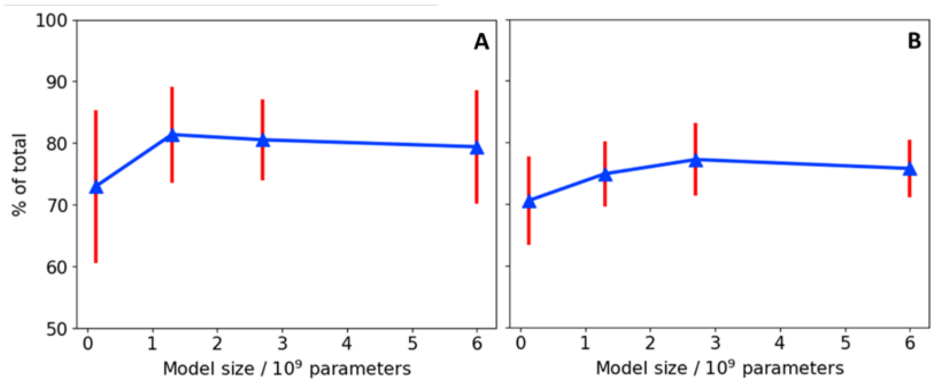


Figure 4.12 Pass@1 as a function of model size. All results were collected using a threshold p of 0.5 and three in-context examples (i.e. n=3), and relate to in-domain A) fully-specified and B) under-specified problems.

4.2.2 Hyperparameter optimisation: Out-of-domain problems

For out-of-domain problems, grid searches were also used to assess performance as a function of hyperparameters (i.e. threshold p and number of in-context examples n). Figure 4.13 shows grid search results for fully-specified problems. Distinct differences in model behaviour are observed. The pass@1 distribution for GPT-Neo-125M and GPT-Neo-1.3B is relatively uniform with neither surpassing a 4% pass rate irrespective of n. Whilst GPT-Neo-2.7B and GPT-J-6B both display the familiar inverse non-linear relationship between pass@1 and threshold P when n is greater than or equal to one.

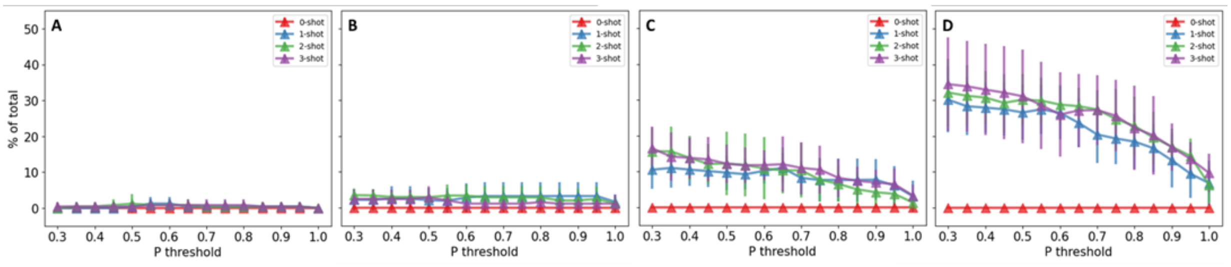


Figure 4.13 Average proportion of model outputs which pass all unit tests (pass@1) as a function of threshold P and number of in-context examples. All results relate to fully-specified, out-of-domain problems. Panels A, B, C and D correspond to GPT-Neo-125M, GPT-Neo-1.3B, GPT-Neo-2.7B and GPT-J-6B, respectively.

Once more, the benefits of increasing n above one appear marginal for all models (Figure 4.14A). Where increases are observed, the corresponding standard deviation error bars overlap which suggests differences are insignificant. This behaviour indicates model inference is most dependent on the most semantically similar in-context example and additional in-context examples do not improve performance for the majority of out-of-domain problems. In contrast, a clear improvement in performance is observed as model size increases. Figure 4.14B shows pass@1 as a function of model size for fully- and under-specified problems when p is 0.3 and n is 3. The corresponding Spearman's rank correlation coefficient was 0.89 which confirms a strong positive monotonic relationship. Crucially, its low p-value (1.3×10^{-14}) means the null hypothesis can be rejected at a 5% significance level. Where the null hypothesis is as follows: "*There is no significant difference in pass@1 between GPT-Neo-125M and GPT-J-6B when applied to out-of-domain fully-specified problems from the cdCorpus"* (see section 3.6.2).

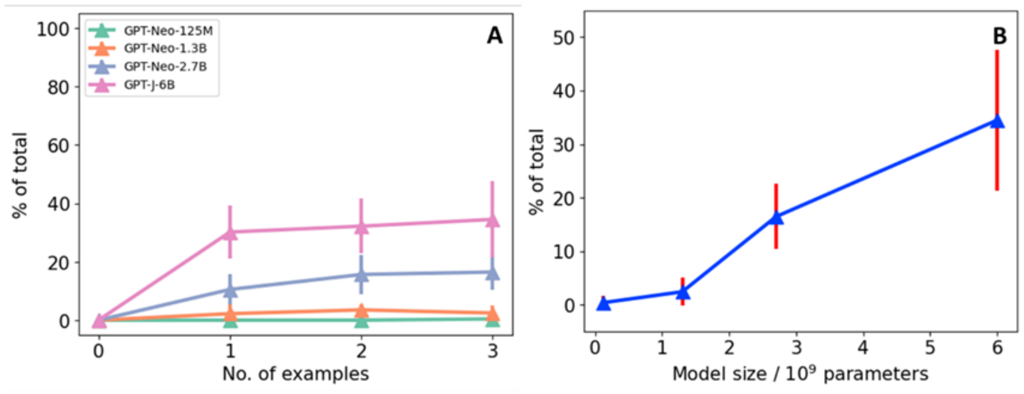


Figure 4.14 Pass@1 as a function of A) number of in-context examples (n) and model type, and B) model size. All results were collected using a threshold p of 0.3 and relate to out-of-domain, fully-specified problems.

Unfortunately, time constraints meant only GPT-J-6B could be tested with under-specified problems. Figure 4.15 shows corresponding grid search results. Comparable behaviour is displayed. Noticeably pass@1 is lower for under-specified problems when using optimum conditions. For instance, when threshold p is set to 0.3, pass@1 was 34.5 ± 13.1 and 21.7 ± 6.7% for fully- and under-specified problems, respectively.

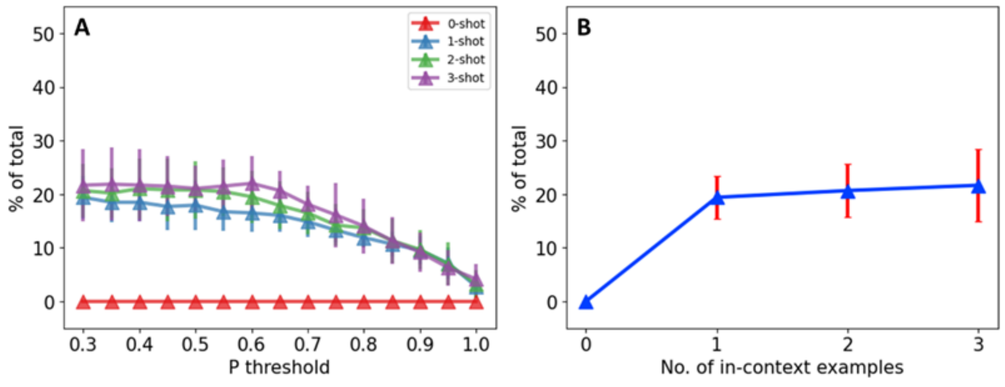


Figure 4.15 A) Pass@1 as a function of threshold p and number of in-context examples (n) B) Pass@1 as a function of number of in-context examples (n) when threshold p is set to 0.3. All results were collected with GPT-J-6B and relate to under-specified, out-of-domain problems.

4.2.3 Model evaluation

GPT-Neo-125M and GPT-J-6B were both tested with an in-domain and out-of-domain version of the test set (see section 3.2.3). Prior grid searches on training data showed performance was reasonably stable as p tends towards 0.3 (e.g. Figure 4.15A). For the purposes of testing, threshold p was set to 0.3 and n to 3 as these conditions are assumed to be sufficiently close to the optimum.

Table 4.5 summarises in-domain test results. Pass@1 was 73.1% and 78.1% with GPT-Neo-125M and GPT-J-6B, respectively. Figure 4.16 shows a visualisation generated by each model. Appendix J displays a further ten visualisations, which passed all unit tests, generated by GPT-J-6B.

Model	Attribute error / %	Syntax error / %	Validation error / %	Logic error / %	Pass@1 / %
GPT-J-6B	0.0	0.6	7.5	13.8	78.1
GPT-Neo-125M	0.0	1.3	3.1	22.5	73.1

Table 4.5 A percentage breakdown of model outputs (when one output is generated per problem) for GPT-Neo-125M and GPT-J-6B. All results were collected with a threshold p of 0.3 and three in-context examples (i.e. n=3).

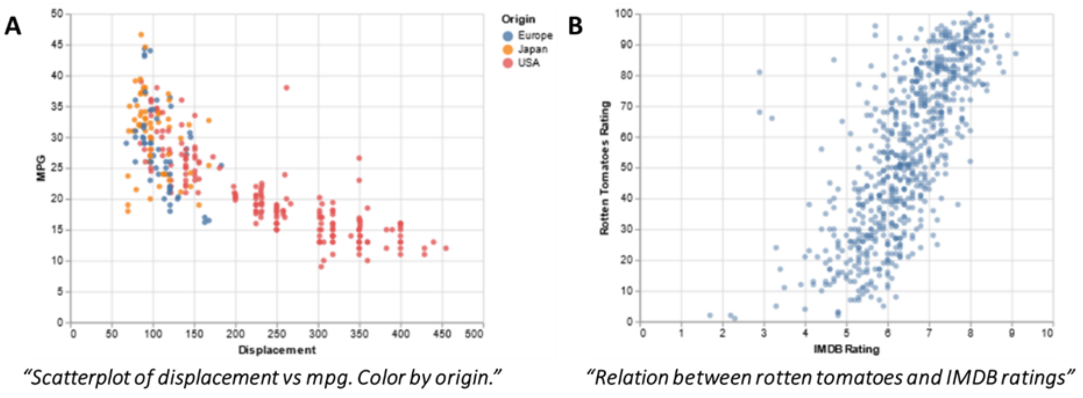


Figure 4.16 A visualisation created by A) GPT-Neo-125M and B) GPT-J-6B for an in-domain problem. Both were generated with a threshold p of 0.3 and three in-context examples (i.e. n=3).

As both models were tested on the same data, a contingency table was used to count and compare problem outcomes (Table 4.6). This analysis showed GPT-J-6B correctly answered 19 problems that GPT-Neo-125M got wrong. Whilst GPT-Neo-125M correctly answered 12 problems which GPT-J-6B got wrong. McNemar's test was used to assess the significance of this difference in performance. The null hypothesis was as follows: "*There is no significant difference in pass@1 between GPT-Neo-125M and GPT-J-6B when applied to in-domain problems from the cdCorpus*" (see section 3.6.2). The test returned a p-value of 0.28 (McNemar's statistic: 12.0), which means this hypothesis cannot be rejected at the 95% confidence level. This aligns with observations during hyperparameter optimisation (Figure 4.12).

	Incorrect $_{\text{GPT-J}}$	**Correct** $_{\text{GPT-J}}$

Incorrect $_{\text{GPT-Neo}}$	24	19
Correct $_{\text{GPT-Neo}}$	12	105

Table 4.6 A contingency table displaying the frequency counts of correct and incorrect solutions generated by GPT-Neo-125M and GPT-J-6B for the in-domain test set. All results were generated with a threshold p of 0.3 and three in-context examples (i.e. n=3).

Table 4.7 summarises out-of-domain test results. No problems were successfully answered using GPT-Neo-125M (pass@1: 0%). However, pass@1 was 30.0% with GPT-J-6B. Appendix K presents ten visualisations, which passed all unit tests, generated by GPT-J. Once again, McNemar's test was used to assess the null hypothesis: "*There is no significant difference in pass@1 between GPT-Neo-125M and GPT-J-6B when applied to out-of-domain problems from the cdCorpus"* (see section 3.6.2) The resulting test statistic (0.0) and corresponding p-value (1.4×10^{-14}) indicate the null hypothesis can be rejected at a 95% confidence level. This again aligns with observations during hyperparameter optimisation (Figure 4.13).

Model	**Attribute error / %**	**Syntax error / %**	**Validation error / %**	**Logic error / %**	**Pass@1 / %**
GPT-J-6B	0.0	0.0	11.9	56.9	30.0
GPT-Neo-125M	3.1	10.0	10.0	76.8	0.0

Table 4.7 A percentage breakdown of model outputs (when one output is generated per problem) for GPT-Neo-125M and GPT-J-6B. All results were collected for out-of-domain problems with a threshold P of 0.3 and three in-context examples.

	Incorrect $_{\text{GPT-J}}$	**Incorrect** $_{\text{GPT-J}}$
Incorrect $_{\text{GPT-Neo}}$	113	47
Correct $_{\text{GPT-Neo}}$	0	0

Table 4.8 A contingency table displaying the frequency counts of correct and incorrect solutions generated by GPT-Neo-125M and GPT-J-6B for the out-of-domain test set. All results were generated with a threshold p of 0.3 and three in-context examples (i.e. n=3).

The decline in performance for out-of-domain problems can be attributed to a high logic error rate (GPT-Neo-125M: 76.8%, GPT-J-6B: 56.9%). Figure 4.17 provides a breakdown of the proportion of problems which failed each unit test. This shows the most frequent reason for a logic error was an incorrect encoding in the position (GPT-Neo-125M: 76.9%, GPT-J-6B: 46.9%), facet (GPT-Neo-125M: 11.9%, GPT-J-6B: 11.9%) or mark colour (GPT-Neo-125M: 36.3%, GPT-J-6B: 26.9%) channel. It should be noted that GPT-Neo-125M does correctly infer the data source and mark type for the majority of problems with only 1.3% of problems failing the respective logical tests. This suggests GPT-J's superior performance is because it is able to map the required column names (and data types) from the metadata (included in the prompt) to the relevant channels in the output specification.

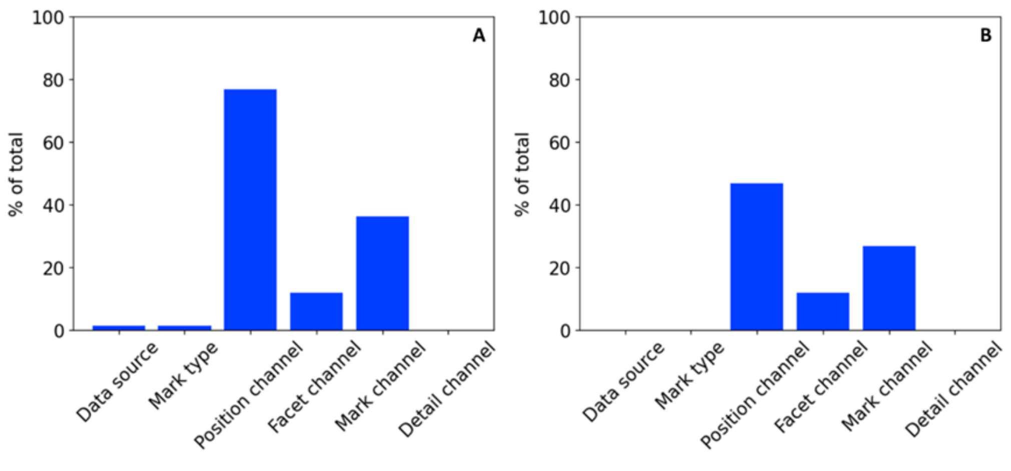


Figure 4.17 The proportion of problems which failed each logical test when one output is generated per problem for A) GPT-Neo-125M and B) GPT-J-6B All results were collected with a threshold P of 0.3 and three in-context examples (i.e. n=3).

As explained earlier, NL4DV does not receive in-context examples as an input. It is designed for use with NL queries alone and therefore does not distinguish between in-domain/out-of-domain versions of a test set. This means its performance on the in-domain test set (pass@1: 43.8%), described in section 4.1.4, can be compared to GPT-J-6B's performance on the out-of-domain test set (pass@1: 30.0%). These results show GPT-J-6B performs worse than NL4DV when prompts do not contain in-context examples associated with the same data as the problem.

4.3 Research objective 3: Assess the sensitivity of the best performing model to the query description

Performance on the Spanish in-domain test set was assessed using the most powerful model and hyperparameter configuration discovered as part of the second research objective. This was GPT-J-6B with threshold p set to 0.3 and the number of in-context examples n set to 3. Table 4.8 and Figure 4.18 summarise the results (see section 3.6.3). Appendix L presents a sample of visualisations that passed all unit tests. Pass@1 was 72.5%. This is only slightly lower than that achieved by GPT-J-6B, using the same conditions, on the English version of the in-domain test set (pass@1: 73.1%).

Model	**Validation error / %**	**Logic error / %**	**Pass@1 / %**
GPT-J-6B	6.3	21.3	72.5

Table 4.8 A percentage breakdown of model outputs (when one output is generated per problem) for in-domain problems (translated to Spanish). All GPT-J-6B results were collected with a threshold p of 0.3 and three in-context examples (i.e. n=3).

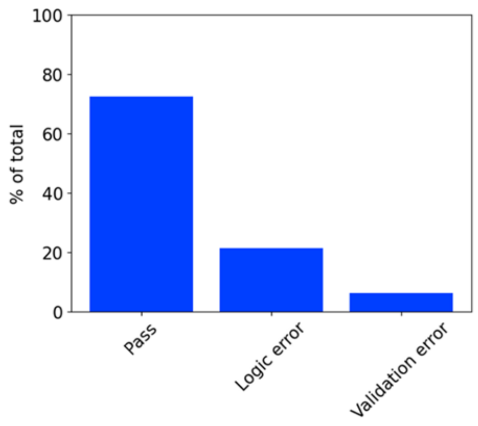


Figure 4.18 A percentage breakdown of model outputs (when one output is generated per problem) for in-domain problems (translated to Spanish). All results were collected using GPT-J-6B with a threshold p of 0.3 and three in-context examples (i.e. n=3).

Figure 4.19 provides a breakdown of the proportion of problems which failed each logical test. Consistent with earlier experiments, incorrect encodings for the x and/or y channel were the most frequent source of error (position channel error rate: 17.5%). A common cause was incorrect capitalization of the relevant field in the specification. Incorrect encodings were also found in the row and column channels (facet channel error rate: 8.8%) and mark colour channel (mark channel error rate: 8.8%). These errors occur for more complex problems where a grid of multiple charts and/or coloured marks are requested.

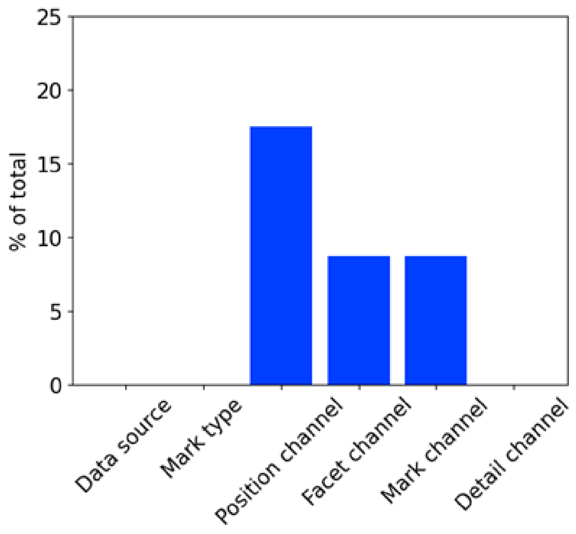


Figure 4.19 The proportion of problems which failed each unit test when one output is generated per problem. All GPT-J-6B results were collected with a threshold p of 0.3 and three in-context examples (i.e. n=3). No solutions failed the data source, mark type and detail channel unit tests.

Performance was broken down by target mark type (Table 4.9). GPT-J did not generate a valid area chart (mark type: area) or strip plot (mark type: tick). The respective subpopulations are too small to draw conclusions. However this may be due to the low number of these charts in the training data from which in-context examples were drawn (see section 3.2.3).

Mark type	Count	Pass@1
area	1	0.0%
bar	83	69.9%
line	25	80.0%
point	50	76.0%
tick	1	0.0%

Table 4.9 Pass@1 by target chart mark type. All GPT-J-6B results were collected with a threshold p of 0.3 and three in-context examples (i.e. n=3).

The benefit of independently sampling multiple solutions per problem was again assessed (Table 4.10) No difference in performance was observed (pass@1: 72.5%, pass@10: 72.5%). This may be due to the low value of threshold p reducing the variance in generated solutions (Holtzman et al., 2019).

Model	pass@1	pass@10
GPT-J-6B	72.5	72.5

Table 4.10 Pass@1 and pass@10 for GPT-J-6B. All results were collected with a threshold p of 0.3 and three in-context examples (i.e. n=3).

As explained in section 3.6.3, no reference is made to non-English languages in the original NL4DV journal article or supporting documentation (Narechania, Srinivasan & Stasko, 2021). A random sample of ten problems was tested. The results are presented in Appendix M. No problems were answered correctly. Eight resulted in attribute errors due to no specification being returned. Visualisations were generated for the remaining two problems. However, both are inappropriate for the given query. Given the sample size, no rigorous statistical test of the proposed null hypothesis (shown below) is possible. Nevertheless, these preliminary results suggest it could be rejected for the default NL4DV configuration.

"There is no significant difference in pass@1 between GPT-J-6B and NL4DV when applied to the Spanish in-domain problems from the cdCorpus"

4.4 Research objective 4: Evaluate the optimal model's ability to generate visualisations from Higher Education (HE) data

Performance on the HE test set was assessed using NL4DV and the most powerful autoregressive model discovered as part of the second research objective. This was GPT-J-6B with threshold p set to 0.3 and the number of in-context examples n set to 3. Table 4.12 summarises the results. Unfortunately, both models exhibited a high logic error rate (GPT-J-6B: 80.3%, NL4DV: 78.8%). In total, GPT-J-6B generated only 10 solutions which passed all unit tests (pass@1: 7.6%) compared to 18 for NL4DV (pass@1: 13.6%).

Model	Attribute error / %	Syntax error / %	Validation error / %	Logic error / %	Pass@1 / %
GPT-J-6B	2.3	0.8	9.1	80.3	7.6
NL4DV	7.6	0.0	0.0	78.8	13.6

Table 4.12 A percentage breakdown of model outputs (when one output is generated per problem) for the HE test set. All GPT-J-6B results were collected with a threshold p of 0.3 and three in-context examples (i.e. n=3).

McNemar's test was used to assess the statistical significance of differences in performance and test the null hypothesis: "*There is no significant difference in pass@1 between GPT-J-6B and NL4DV when applied to out-of-domain problems from the heCorpus*". The test returned a p-value of 0.12 (McNemar's statistic: 6.0). At a 95% confidence level, the null hypothesis could therefore not be rejected.

Although not statistically significant, in this instance NL4DV's improved performance can be attributed to lower syntax and validation error rates. Figure 4.20 provides a percentage breakdown of GPT-J-6B and NL4DV outputs. The models exhibit similar logic error rates (GPT-J-6B: 80.3%, NL4DV: 78.8%). However, NL4DV's rule based architecture means generated specifications comply with the Python dictionary syntax and adhere to the Vega-Lite schema. Consequently, no syntax or validation errors were observed. In contrast, syntax and validation errors accounted for roughly 10% of failures with GPT-J-6B.

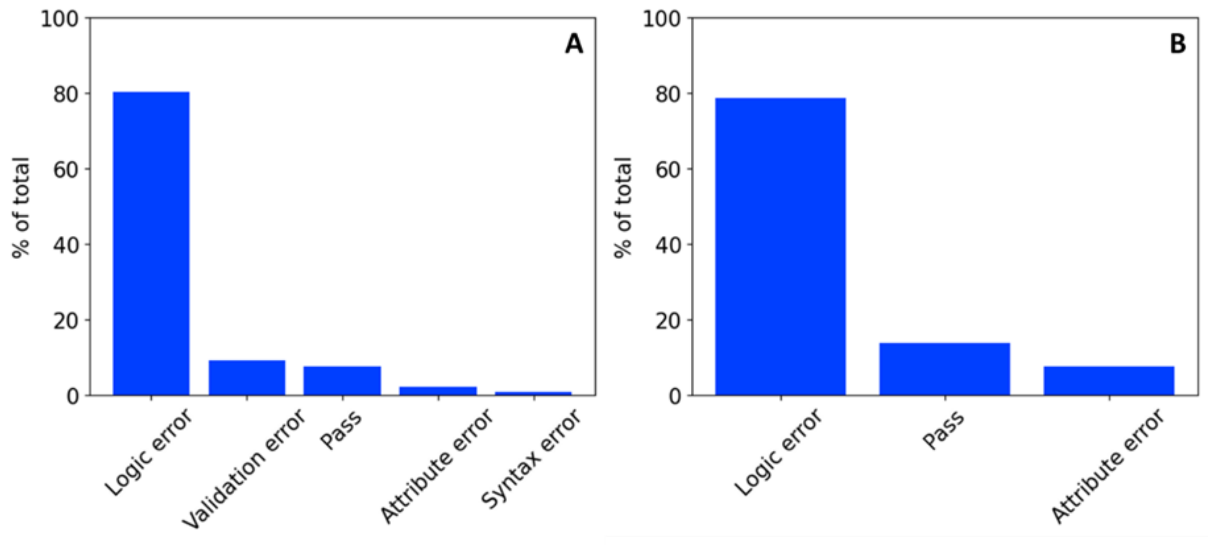


Figure 4.20 A percentage breakdown of A) GPT-J-6B and B) NL4DV outputs when one output is generated per problem. All GPT-J-6B results were collected with a threshold P of 0.3.

The proportion of problems output by both models which failed each logical test is shown in Figure 4.21. Analogous to earlier experiments, both model's primary reasons for a logic error were incorrect encodings for x/y axis (GPT-J-6B & NL4DV position channel error rate: 76.5% and 72.0%, respectively), row/column properties (GPT-J-6B & NL4DV facet channel error rate: 22.0% and 21.2%, respectively) and mark colour (GPT-J-6B and NL4DV mark channel error rate: 35.6% and 25.0%, respectively). Taken altogether, these errors mean that problems related to more complex visualisations such as stacked bar charts were unsuccessful. Table 4.13 shows the pass rates for each type of visualisation. For GPT-J-6B, only problems related to histograms, unsorted bar charts and scatter plots (multiples) were successful. Whilst only unsorted bar charts, line charts, scatter plots and coloured scatter plots were created successfully with NL4DV.

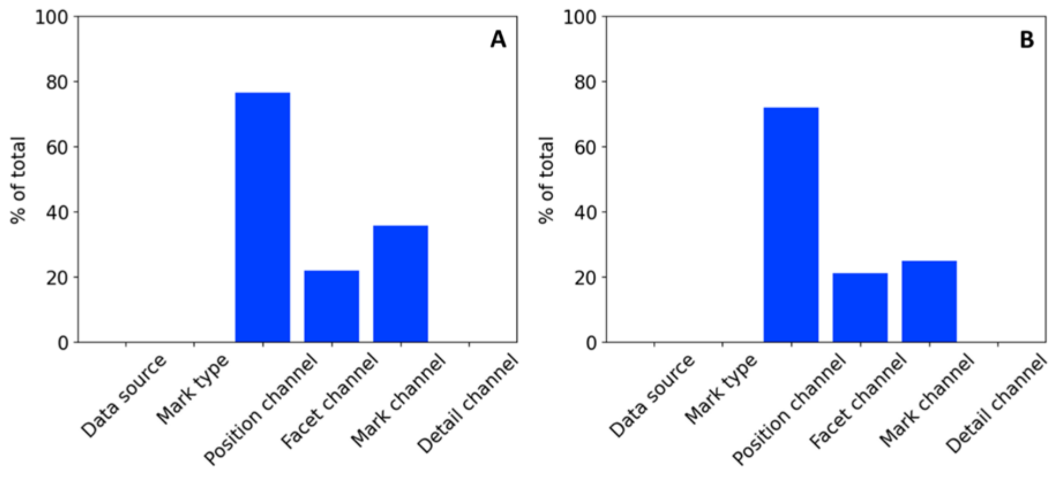


Figure 4.21 The proportion of problems generated by A) GPT-J-6B and B) NL4DV which failed each logical test when one output is generated per problem. All GPT-J-6B results were collected with a threshold p of 0.3 and three in-context examples (i.e. n=3).

Visualisation	GPT-J-6B	NL4DV
Bar chart (multiples)	0.0	0.0
Bar chart (sorted)	0.0	0.0
Bar chart (unsorted)	7.1	21.4
Histogram	33.3	0.0
Line chart	40.0	60.0
Line chart (multiples)	0.0	0.0
Line chart (multi series)	0.0	0.0
Scatter plot	0.0	60.0
Scatter plot (coloured)	0.0	50.0
Scatter plot (multiples)	20.0	0.0
Stacked bar chart	0.0	0.0
Stacked bar chart (normalised)	0.0	0.0
Strip plot	0.0	0.0

Table 4.13 Pass@1 by chart type. All GPT-J-6B results were collected with a threshold p of 0.3 and three in-context examples (i.e. n=3).

The HE test data is out-of-domain as in-context examples were drawn from the cdCorpus training set. Interestingly GPT-J-6B exhibited a substantially lower pass@1 with this data than for the out-of-domain test data from the cdCorpus assessed as part of the second research objective (GPT-J-6B: 30%). Given NL4DV also performed poorly on HE data, this suggests differences exist between the heCorpus and cdCorpus. Interestingly, the median length of queries in the cdCorpus was 9 words compared to 16 words for the heCorpus. The decline in performance may therefore be due to poorer semantic parsing for more verbose NL queries. It should also be noted the user studies used to curate the corpora differed in terms of target audience. The heCorpus was collected using the 'sampleSize' Reddit forum. Whilst the cdCorpus was collected using visualisation-specific forums. For instance, it included Tableau user groups on LinkedIn, the 'visualization' Reddit forum and the 'PowerBI Reddit forum' (Srinivasan et al., 2021). An investigative hypothesis which could be tested in the future is that users with technical expertise provide more concise and easily parsed NL queries.

4.4 Conclusion

This chapter has provided an overview of key experimental results for each research objective. The next section will contextualise findings in terms of the wider literature and original research objectives.

Chapter 5: Discussion

"Can state-of-the-art transformer-based autoregressive language models, pre-trained using large web-based corpora, be used to generate data visualisations from natural language statements?"

The overall research question for this project is stated above. Experiments demonstrated successful visualisation generation from NL queries using four autoregressive models pre-trained using the Pile (Gao et al., 2021): GPT-Neo-125M, GPT-Neo-1.3B, GPT-Neo-2.7B and GPT-J-6B. This was achieved without any additional task-specific fine tuning of model parameters. To the best of the author's knowledge, this is the first example of pre-trained autoregressive models being applied to NL2VIS and is therefore a useful proof of concept for the NL2VIS community. The level of performance depended on model size and problem type. It is therefore useful to frame further discussions with respect to the original research objectives.

Research objective 1: Design and create an end-to-end model which will generate Vega-Lite visualisations from NL queries for a given dataset.

"Vega-Lite visualisations can be generated from natural language statements by using a pre-trained GPT-Neo-125M model in an end-to-end, modular model."

The Hugging Face library enabled development of an end-to-end model where pre-trained autoregressive models could easily be interchanged. Average performance using prompt-based learning was assessed using (NL query, Vega-Lite specification) pairs from a previously published corpus (cdCorpus). When in-context examples can be associated with the same dataset as a given problem (i.e. in-domain problems), the performance of GPT-Neo-125M (pass@1: 73.1%) was superior to NL4DV (45.0%). Importantly McNemar's test enabled the rejection of the null hypothesis: "*There is no significant difference in pass@1 between GPT-Neo-125M and NL4DV when applied to* *in-domain problems from the cdCorpus*". The key success criterion outlined in the introduction was therefore met.

Research objective 2: Design and create an end-to-end model which will generate Vega-Lite visualisations from NL queries for a given dataset.

"Scaling up the size of pre-trained transformed-based models will improve Vega-Lite visualisation generation performance."

The average performance of GPT-Neo-125M, GPT-Neo-1.3B, GPT-Neo-2.7B and GPT-J-6B using prompt-based learning was assessed using (NL query, Vega-Lite specification) pairs from a previously published corpus (cdCorpus). When applied to in-domain problems, no significant differences in performance were observed. In contrast, for out-of-domain problems, a significant positive correlation was observed between model size (i.e. no. of parameters) and average performance. This reflects the difficulty of the task. As in-context examples cannot be associated with the same dataset as problems, a model must infer the change in data source and attributes. In this instance, experiments showed GPT-Neo-125M was unable to generate a visualisation that passed all unit tests. In contrast, GPT-J-6B achieved an overall pass rate of 30.0%. The following null hypothesis could therefore be rejected using McNemar's test: "*There is no significant difference in pass@1 between GPT-Neo-125M and NL4DV when applied to out-of-domain problems from the cdCorpus*". This improvement in performance was expected as this phenomena has, in recent years, motivated substantial increases in model size (e.g. GPT-3>GPT-2>GPT) (Brown et al., 2020, Radford et al., 2018).

It should be noted these results do also show GPT-J-6B performs worse than NL4DV when prompts do not contain in-context examples associated with the same data as the problem (i.e. for out-of-domain problems). This suggests NL4DV is more generalisable.

Research objective 3: Assess the sensitivity of the best performing model to the query description.

"Vega-Lite visualisations can be generated from Spanish natural language commands using pre-trained transformer-based autoregressive language models."

GPT-J-6B demonstrated successful visualisation generation for Spanish in-domain problems from the cdCorpus (pass@1: 72.5%). Interestingly, performance was similar to that achieved using the English version of the test set (pass@1: 73.1%). Experiments demonstrated NL4DV is not currently optimised for non-English NL. The results therefore supported rejection of the null hypothesis: *"There is no significant difference in pass@1 between GPT-J-6B and NL4DV when applied to the Spanish in-domain problems from the cdCorpus"*. Unfortunately, time did not permit experiments with out-of-domain problems. Nevertheless, the results suggest autoregressive language models, pretrained using multilingual text, can solve non-English NL2VIS tasks without any changes in architecture. This addresses a key constraint of rule-based systems.

Research objective 4: Evaluate the optimal model's ability to generate visualisations from Higher Education (HE) data.

"Vega-Lite visualisations related to Higher Education data can be generated from natural language commands using pre-trained transformer-based autoregressive language models."

Unfortunately, GPT-J-6B performed poorly on queries from the heCorpus(pass@1: 7.6%). This is substantially lower than that observed for out-of-domain problems from the cdCorpus (pass@1: 30.0%). The key success criterion outlined in section 1.2 was comparable performance to NL4DV. Given NL4DV's performance on HE queries was also weak (pass@1: 13.6%), this was achieved. In reality, neither model provides adequate performance. The decline in performance can perhaps be attributed to differences in the heCorpus and cdCorpus. The former was collected using the 'sampleSize' Reddit forum. Whilst cdCorpus was collected using visualisation-specific forums. For instance, it included Tableau user groups on LinkedIn, the 'visualization' Reddit forum and the 'PowerBI' Reddit forum (Srinivasan et al., 2021). These results may therefore reflect the difficulty of generalising to non-specialist audiences. A user study on HE visualisations where participants are domain experts would help test this hypothesis. Larger pre-trained autoregressive models may also improve semantic parsing for queries from lay users.

Chapter 6: Evaluation, Reflections, and Conclusions

The results presented prove that NL2VIS using pre-trained autoregressive models is possible. Furthermore, visualisations have been generated from non-English NL commands, which is a key limitation of rule-based systems. Despite these positives, GPT-J-6B was inferior to NL4DV when tested on out-of-domain problems. Therefore, further optimisation is required. The analysis revealed several limitations of the research design and therefore areas of potential interest in future studies. These are summarised below.

- A user study should be used to collect a new, larger corpus containing (NL command, Vega-Lite specification) pairs related to HE. Critically, survey participants should include lay users; users with experience using visualisation tools and users with knowledge of the HE domain. Ideally, participants would also be asked to assess the usefulness of the visualisations generated. A drawback of the current approach to evaluation is that match based metrics may fail visualisations which differ from the expected visualisation, but produce a visualisation which answers the query nonetheless.
- The performance of GPT-J-6B on in-domain Spanish problems was promising. However, a user study should be used to collect multilingual (NL command, Vega-Lite specification) pairs. This would eliminate the need for Google Translate, whose accuracy could not be

verified, and enable improved estimates of performance on colloquial non-English phrases. A priority of experiments should be optimising models for out-of-domain non-English phrases as this is key to generalisation.

- Larger pre-trained autoregressive models should be tested. The original research proposal included plans to use GPT-3 (~175 billion parameters)(Brown et al., 2020) . However, this was not possible due to cost. Given improvements in performance were observed for out-of-domain problems as model size increases, larger models such as GPT-3 should now be tested.
- Performance should be assessed after pre-training of models using different datasets. Text in the Pile is ~97.4% English. Furthermore, code from github only accounts for 8% of the dataset (Gao et al., 2021). Performance may be improved using models pretrained with a higher proportion of code and non-English text (Chen et al., 2021).
- The manually designed prompts may not be optimal. Alternative designs should be tested. Automated approaches to design have been demonstrated and these should be explored (Jiang et al., 2020, Lu et al., 2021, Liu et al., 2021).

Glossary

Autoregressive: Inference based on previously generated outputs.

Declarative: Declarative approaches to visualising data involve the specification of what should be plotted. The required steps are determined by the compiler using predefined rules. Vega-Lite provides a declarative approach to visualisation.

Feed forward neural network: A type of neural network which is not recurrent. An example is the multi-layer perceptron.

Generalisation: A machine model that is generalisable performs well with out-of-distribution test data.

In-domain problem: For the purposes of this research project, an in-domain problem is one where in-context examples can be associated with the same dataset as the given problem's NL query.

Imperative: Imperative approaches to visualising data involve the specification of what should be plotted and how this should be executed (i.e. plotting steps). Plotting charts with the MatplotLib library is imperative.

Out-of-domain problem: For the purposes of this research project, an out-of-domain problem is one where in-context examples cannot be associated with the same dataset as the given problem's NL query.

References

Bolton, P. 2021, Higher Education funding in England, House of Commons Libary.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J.D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I. & Amodei, D. 2020, "Language Models are Few-Shot Learners", Advances in Neural Information Processing Systems, eds. H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan & H. Lin, Curran Associates, Inc, , pp. 1877.

Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H.P.d.O., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., Ray, A., Puri, R., Krueger, G., Petrov, M., Khlaaf, H., Sastry, G., Mishkin, P., Chan, B., Gray, S., Ryder, N., Pavlov, M., Power, A., Kaiser, L., Bavarian, M., Winter, C., Tillet, P., Such, F.P., Cummings, D., Plappert, M., Chantzis, F., Barnes, E., Herbert-Voss, A., Guss, W.H., Nichol, A., Paino, A., Tezak, N., Tang, J., Babuschkin, I., Balaji, S., Jain, S., Saunders, W., Hesse, C., Carr, A.N., Leike, J., Achiam, J., Misra, V., Morikawa, E., Radford, A., Knight, M., Brundage, M., Murati, M., Mayer, K., Welinder, P., McGrew, B., Amodei, D., McCandlish, S., Sutskever, I. & Zaremba, W. 2021, Evaluating Large Language Models Trained on Code, arXiv.

Devlin, J., Chang, M., Lee, K. & Toutanova, K. 2019, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding", Proceedings of the 2019 Conference of the North {A}merican Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)Association for Computational Linguistics, Minneapolis, Minnesota, jun, pp. 4171.

Gao, T., Dontcheva, M., Adar, E., Liu, Z., Karahalios, K. & ACM 2015, "DataTone: Managing Ambiguity in Natural Language Interfaces for Data Visualization", ACM, NEW YORK, pp. 489.

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V. & Zettlemoyer, L. 2019, "BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension", .

Li, Y., Choi, D., Chung, J., Kushman, N., Schrittwieser, J., Leblond, R., Eccles, T., Keeling, J., Gimeno, F., Lago, A.D., Hubert, T., Choy, P., d'Autume, C.d.M., Babuschkin, I., Chen, X., Huang, P., Welbl, J., Gowal, S., Cherepanov, A., Molloy, J., Mankowitz, D.J., Robson, E.S., Kohli, P., de Freitas, N., Kavukcuoglu, K. & Vinyals, O. 2022, Competition-Level Code Generation with AlphaCode, arXiv.

Liu, C., Han, Y., Jiang, R. & Yuan, X. 2021, "ADVISor: Automatic Visualization Answer for Natural-Language Question on Tabular Data", 2021 Ieee 14th Pacific Visualization Symposium (Pacificvis 2021), , pp. 11-20.

Luo, Y., Tang, N., Li, G., Chai, C., Li, W., Qin, X. & ASSOC COMP MACHINERY 2021, "Synthesizing Natural Language to Visualization (NL2VIS) Benchmarks from NL2SQL Benchmarks", Assoc Computing Machinery, NEW YORK, pp. 1235.

Luo, Y., Tang, N., Li, G., Tang, J., Chai, C. & Qin, X. 2022, "Natural Language to Visualization by Neural Machine Translation", IEEE Transactions on Visualization and Computer Graphics, vol. 28, no. 1, pp. 217-226.

McCoy, T., Pavlick, E. & Linzen, T. 2019, "Right for the Wrong Reasons: Diagnosing Syntactic Heuristics in Natural Language Inference", Proceedings of the 57th Annual Meeting of the Association for Computational LinguisticsAssociation for Computational Linguistics, Florence, Italy, jul, pp. 3428.

Microsoft 2021, Feb 9,-last update, Q&A for Power BI business users. Available: https://learn.microsoft.com/en-us/power-bi/consumer/end-user-q-and-a [2022, December 18,].

Min, B., Ross, H., Sulem, E., Veyseh, A.P.B., Nguyen, T.H., Sainz, O., Agirre, E., Heinz, I. & Roth, D. 2021, Recent Advances in Natural Language Processing via Large Pre-Trained Language Models: A Survey, arXiv.

Mitra, R., Narechania, A., Endert, A. & Stasko, J. 2022, "Facilitating Conversational Interaction in Natural Language Interfaces for Visualization", .

Narechania, A., Srinivasan, A. & Stasko, J. 2021, "NL4DV: A Toolkit for Generating Analytic

Specifications for Data Visualization from Natural Language Queries", IEEE Transactions on Visualization and Computer Graphics, vol. 27, no. 2, pp. 369-379.

Office for Students 2022a, Oct 7,-last update, About the TEF. Available: https://www.officeforstudents.org.uk/advice-and-guidance/teaching/about-the-tef/ [2022, December 18,].

Office for Students 2022b, Oct 6,-last update, Access and participation plans. Available: https://www.officeforstudents.org.uk/advice-and-guidance/promoting-equal-opportunities/access-and-participation-plans/ [2022, December 18,].

Office for Students 2022c, Securing student success:
Regulatory framework for
higher education in England, Office for Students.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D. & Sutskever, I. 2018, "Language Models are Unsupervised Multitask Learners", .

Setlur, V., Tory, M. & Djalali, A. 2019, "Inferencing Underspecified Natural Language Utterances in Visual Analysis", Proceedings of Iui 2019, , pp. 40-51.

Shin, R. & Van Durme, B. 2021, "Few-Shot Semantic Parsing with Language Models Trained On Code", .

Srinivasan, A. & Stasko, J. 2017, "Natural Language Interfaces for Data Analysis with Visualization: Considering What Has and Could Be Asked", EuroVis 2017 - Short Papers, .

Sun, Y., Leigh, J., Johnson, A. & Lee, S. 2010, "Articulate: A Semi-automated Model for Translating Natural Language Queries into Meaningful Visualizations" in SMART GRAPHICS, PROCEEDINGS, eds. R. Taylor, P. Boulanger, A. Kruger & P. Olivier, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 184-195.

Tableau Software 2022, , Automatically Build Views with Ask Data. Available: https://help.tableau.com/current/pro/desktop/en-us/ask_data.htm [2022, December 18,].

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L. & Polosukhin, I. 2017, "Attention is All You Need", Proceedings of the 31st International Conference on Neural Information Processing SystemsCurran Associates Inc, Red Hook, NY, USA.

Alammar, J. 2018, Jun 27,-last update, The Illustrated Transformer. Available: https://jalammar.github.io/illustrated-transformer/ [2022, December 18,].

Austin, J., Odena, A., Nye, M., Bosma, M., Michalewski, H., Dohan, D., Jiang, E., Cai, C., Terry, M., Le, Q. & Sutton, C. 2021, Program Synthesis with Large Language Models, arXiv.

Black, S., Gao, L., Wang, P., Leahy, C. & Biderman, S. 2021, "GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow", .

Bolton, P. 2021, Higher Education funding in England, House of Commons Libary.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J.D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I. & Amodei, D. 2020, "Language Models are Few-Shot Learners", Advances in Neural Information Processing Systems, eds. H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan & H. Lin, Curran Associates, Inc, , pp. 1877.

Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H.P.d.O., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., Ray, A., Puri, R., Krueger, G., Petrov, M., Khlaaf, H., Sastry, G., Mishkin, P., Chan, B., Gray, S., Ryder, N., Pavlov, M., Power, A., Kaiser, L., Bavarian, M., Winter, C., Tillet, P., Such, F.P., Cummings, D., Plappert, M., Chantzis, F., Barnes, E., Herbert-Voss, A., Guss, W.H., Nichol, A., Paino, A., Tezak, N., Tang, J., Babuschkin, I., Balaji, S., Jain, S., Saunders, W., Hesse, C., Carr, A.N., Leike, J., Achiam, J., Misra, V., Morikawa, E., Radford, A., Knight, M., Brundage, M., Murati, M., Mayer, K., Welinder, P., McGrew, B., Amodei, D., McCandlish, S., Sutskever, I. & Zaremba, W. 2021, Evaluating Large Language Models Trained on Code, arXiv.

Devlin, J., Chang, M., Lee, K. & Toutanova, K. 2019, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding", Proceedings of the 2019 Conference of the North {A}merican Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)Association for Computational Linguistics, Minneapolis, Minnesota, jun, pp. 4171.

Dodge, J., Sap, M., Marasović, A., Agnew, W., Ilharco, G., Groeneveld, D., Mitchell, M. & Gardner, M. 2021, "Documenting Large Webtext Corpora: A Case Study on the Colossal Clean Crawled Corpus", .

Gao, L., Biderman, S., Black, S., Golding, L., Hoppe, T., Foster, C., Phang, J., He, H., Thite, A., Nabeshima, N., Presser, S. & Leahy, C. 2021, The Pile: An 800GB Dataset of Diverse Text for Language Modeling, arXiv.

Gao, T., Dontcheva, M., Adar, E., Liu, Z., Karahalios, K. & ACM 2015, "DataTone: Managing Ambiguity in Natural Language Interfaces for Data Visualization", ACM, NEW YORK, pp. 489.

Hendrycks, D., Basart, S., Kadavath, S., Mazeika, M., Arora, A., Guo, E., Burns, C., Puranik, S., He, H., Song, D. & Steinhardt, J. 2021, Measuring Coding Challenge Competence With APPS, arXiv.

Hugging Face 2022, , Auto Classes. Available: https://huggingface.co/docs/transformers/model_doc/auto [2022, December 18,].

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V. & Zettlemoyer, L. 2019, "BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension", .

Li, Y., Choi, D., Chung, J., Kushman, N., Schrittwieser, J., Leblond, R., Eccles, T., Keeling, J., Gimeno, F., Lago, A.D., Hubert, T., Choy, P., d'Autume, C.d.M., Babuschkin, I., Chen, X., Huang, P., Welbl, J., Gowal, S., Cherepanov, A., Molloy, J., Mankowitz, D.J., Robson, E.S., Kohli, P., de Freitas, N., Kavukcuoglu, K. & Vinyals, O. 2022, Competition-Level Code Generation with AlphaCode, arXiv.

Liu, C., Han, Y., Jiang, R. & Yuan, X. 2021, "ADVISor: Automatic Visualization Answer for Natural-Language Question on Tabular Data", 2021 Ieee 14th Pacific Visualization Symposium (Pacificvis 2021), , pp. 11-20.

Lu, Y., Bartolo, M., Moore, A., Riedel, S. & Stenetorp, P. 2021, Fantastically Ordered Prompts and Where to Find Them: Overcoming Few-Shot Prompt Order Sensitivity, arXiv.

Luo, Y., Tang, N., Li, G., Chai, C., Li, W., Qin, X. & ASSOC COMP MACHINERY 2021, "Synthesizing Natural Language to Visualization (NL2VIS) Benchmarks from NL2SQL Benchmarks", Assoc Computing Machinery, NEW YORK, pp. 1235.

Luo, Y., Tang, N., Li, G., Tang, J., Chai, C. & Qin, X. 2022, "Natural Language to Visualization by Neural Machine Translation", IEEE Transactions on Visualization and Computer Graphics, vol. 28, no. 1, pp. 217-226.

McCoy, T., Pavlick, E. & Linzen, T. 2019, "Right for the Wrong Reasons: Diagnosing Syntactic Heuristics in Natural Language Inference", Proceedings of the 57th Annual Meeting of the Association for Computational LinguisticsAssociation for Computational Linguistics, Florence, Italy, jul, pp. 3428.

Microsoft 2021, Feb 9,-last update, Q&A for Power BI business users. Available: https://learn.microsoft.com/en-us/power-bi/consumer/end-user-q-and-a [2022, December 18,].

Min, B., Ross, H., Sulem, E., Veyseh, A.P.B., Nguyen, T.H., Sainz, O., Agirre, E., Heinz, I. & Roth, D. 2021, Recent Advances in Natural Language Processing via Large Pre-Trained Language Models: A Survey, arXiv.

Mitra, R., Narechania, A., Endert, A. & Stasko, J. 2022, "Facilitating Conversational Interaction in Natural Language Interfaces for Visualization", .

Narechania, A., Srinivasan, A. & Stasko, J. 2021, "NL4DV: A Toolkit for Generating Analytic Specifications for Data Visualization from Natural Language Queries", IEEE Transactions on Visualization and Computer Graphics, vol. 27, no. 2, pp. 369-379.

Oates, B.J. 2006, Researching information systems and computing, SAGE Publications, London.

Office for Students 2022a, Oct 7,-last update, About the TEF. Available: https://www.officeforstudents.org.uk/advice-and-guidance/teaching/about-the-tef/ [2022, December 18,].

Office for Students 2022b, Oct 6,-last update, Access and participation plans. Available: https://www.officeforstudents.org.uk/advice-and-guidance/promoting-equal-opportunities/access-and-participation-plans/ [2022, December 18,].

Office for Students 2022c, Securing student success:
Regulatory framework for

higher education in England, Office for Students.
Radford, A., Wu, J., Child, R., Luan, D., Amodei, D. & Sutskever, I. 2018, "Language Models are Unsupervised Multitask Learners", .
Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W. & Liu, P.J. 2020, "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer", Journal of machine learning research, vol. 21.
Reimers, N. & Gurevych, I. 2019, "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks", .
Rush, A., Nguyen, V. & Klein, G. 2018, Apr 3,-last update, The Annotated Transformer. Available: https://nlp.seas.harvard.edu/2018/04/03/attention.html#encoder [2022, December 18,].
Satyanarayan, A., Moritz, D., Wongsuphasawat, K. & Heer, J. 2017, "Vega-Lite: A Grammar of Interactive Graphics", IEEE Transactions on Visualization and Computer Graphics, vol. 23, no. 1, pp. 341-350.
Setlur, V., Tory, M. & Djalali, A. 2019, "Inferencing Underspecified Natural Language Utterances in Visual Analysis", Proceedings of Iui 2019, , pp. 40-51.
Shin, R. & Van Durme, B. 2021, "Few-Shot Semantic Parsing with Language Models Trained On Code", .
Srinivasan, A., Nyapathy, N., Lee, B., Drucker, S.M., Stasko, J. & ASSOC COMP MACHINERY 2021, "Collecting and Characterizing Natural Language Utterances for Specifying Data Visualizations", Assoc Computing Machinery, NEW YORK.
Srinivasan, A. & Stasko, J. 2017, "Natural Language Interfaces for Data Analysis with Visualization: Considering What Has and Could Be Asked", EuroVis 2017 - Short Papers, .
Sun, Y., Leigh, J., Johnson, A. & Lee, S. 2010, "Articulate: A Semi-automated Model for Translating Natural Language Queries into Meaningful Visualizations" in SMART GRAPHICS, PROCEEDINGS, eds. R. Taylor, P. Boulanger, A. Kruger & P. Olivier, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 184-195.
Tableau Software 2022, , Automatically Build Views with Ask Data. Available: https://help.tableau.com/current/pro/desktop/en-us/ask_data.htm [2022, December 18,].
Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L. & Polosukhin, I. 2017, "Attention is All You Need", Proceedings of the 31st International Conference on Neural Information Processing SystemsCurran Associates Inc, Red Hook, NY, USA.
von Platen, P. 2020, Oct 8,-last update, Transformers-based Encoder-Decoder Models. Available: https://huggingface.co/blog/encoder-decoder [2022, December 18,].
Wang, B. & Komatsuzaki, A. 2021, GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model.
Yang, Y., Cer, D., Ahmad, A., Guo, M., Law, J., Constant, N., Abrego, G.H., Yuan, S., Tar, C., Sung, Y., Strope, B. & Kurzweil, R. 2019, "Multilingual Universal Sentence Encoder for Semantic Retrieval", .
Alammar, J. 2018, Jun 27,-last update*, The Illustrated Transformer*. Available: https://jalammar.github.io/illustrated-transformer/ [2022, December 18,].
Austin, J., Odena, A., Nye, M., Bosma, M., Michalewski, H., Dohan, D., Jiang, E., Cai, C., Terry, M., Le, Q. & Sutton, C. 2021, *Program Synthesis with Large Language Models*, arXiv.
Black, S., Gao, L., Wang, P., Leahy, C. & Biderman, S. 2021, "GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow", .
Bolton, P. 2021, *Higher Education funding in England*, House of Commons Libary.
Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J.D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I. & Amodei, D. 2020, "Language Models are Few-Shot Learners", *Advances in Neural Information Processing Systems*, eds. H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan & H. Lin, Curran Associates, Inc, , pp. 1877.
Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H.P.d.O., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., Ray, A., Puri, R., Krueger, G., Petrov, M., Khlaaf, H., Sastry, G., Mishkin, P., Chan, B., Gray, S., Ryder, N., Pavlov, M., Power, A., Kaiser, L., Bavarian, M., Winter, C., Tillet, P.,

Such, F.P., Cummings, D., Plappert, M., Chantzis, F., Barnes, E., Herbert-Voss, A., Guss, W.H., Nichol, A., Paino, A., Tezak, N., Tang, J., Babuschkin, I., Balaji, S., Jain, S., Saunders, W., Hesse, C., Carr, A.N., Leike, J., Achiam, J., Misra, V., Morikawa, E., Radford, A., Knight, M., Brundage, M., Murati, M., Mayer, K., Welinder, P., McGrew, B., Amodei, D., McCandlish, S., Sutskever, I. & Zaremba, W. 2021, *Evaluating Large Language Models Trained on Code*, arXiv.

Devlin, J., Chang, M., Lee, K. & Toutanova, K. 2019, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding", *Proceedings of the 2019 Conference of the North {A}merican Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*Association for Computational Linguistics, Minneapolis, Minnesota, jun, pp. 4171.

Dodge, J., Sap, M., Marasović, A., Agnew, W., Ilharco, G., Groeneveld, D., Mitchell, M. & Gardner, M. 2021, "Documenting Large Webtext Corpora: A Case Study on the Colossal Clean Crawled Corpus", .

Gao, L., Biderman, S., Black, S., Golding, L., Hoppe, T., Foster, C., Phang, J., He, H., Thite, A., Nabeshima, N., Presser, S. & Leahy, C. 2021, *The Pile: An 800GB Dataset of Diverse Text for Language Modeling*, arXiv.

Gao, T., Dontcheva, M., Adar, E., Liu, Z., Karahalios, K. & ACM 2015, "DataTone: Managing Ambiguity in Natural Language Interfaces for Data Visualization", ACM, NEW YORK, pp. 489.

Hendrycks, D., Basart, S., Kadavath, S., Mazeika, M., Arora, A., Guo, E., Burns, C., Puranik, S., He, H., Song, D. & Steinhardt, J. 2021, *Measuring Coding Challenge Competence With APPS*, arXiv.

Hugging Face 2022, *, Auto Classes*. Available: https://huggingface.co/docs/transformers/model_doc/auto [2022, December 18,].

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V. & Zettlemoyer, L. 2019, "BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension", .

Li, Y., Choi, D., Chung, J., Kushman, N., Schrittwieser, J., Leblond, R., Eccles, T., Keeling, J., Gimeno, F., Lago, A.D., Hubert, T., Choy, P., d'Autume, C.d.M., Babuschkin, I., Chen, X., Huang, P., Welbl, J., Gowal, S., Cherepanov, A., Molloy, J., Mankowitz, D.J., Robson, E.S., Kohli, P., de Freitas, N., Kavukcuoglu, K. & Vinyals, O. 2022, *Competition-Level Code Generation with AlphaCode*, arXiv.

Liu, C., Han, Y., Jiang, R. & Yuan, X. 2021, "ADVISor: Automatic Visualization Answer for Natural-Language Question on Tabular Data", *2021 Ieee 14th Pacific Visualization Symposium (Pacificvis 2021),* , pp. 11-20.

Lu, Y., Bartolo, M., Moore, A., Riedel, S. & Stenetorp, P. 2021, *Fantastically Ordered Prompts and Where to Find Them: Overcoming Few-Shot Prompt Order Sensitivity*, arXiv.

Luo, Y., Tang, N., Li, G., Chai, C., Li, W., Qin, X. & ASSOC COMP MACHINERY 2021, "Synthesizing Natural Language to Visualization (NL2VIS) Benchmarks from NL2SQL Benchmarks", Assoc Computing Machinery, NEW YORK, pp. 1235.

Luo, Y., Tang, N., Li, G., Tang, J., Chai, C. & Qin, X. 2022, "Natural Language to Visualization by Neural Machine Translation", *IEEE Transactions on Visualization and Computer Graphics,* vol. 28, no. 1, pp. 217-226.

McCoy, T., Pavlick, E. & Linzen, T. 2019, "Right for the Wrong Reasons: Diagnosing Syntactic Heuristics in Natural Language Inference", *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*Association for Computational Linguistics, Florence, Italy, jul, pp. 3428.

Microsoft 2021, Feb 9,-last update*, Q&A for Power BI business users*. Available: https://learn.microsoft.com/en-us/power-bi/consumer/end-user-q-and-a [2022, December 18,].

Min, B., Ross, H., Sulem, E., Veyseh, A.P.B., Nguyen, T.H., Sainz, O., Agirre, E., Heinz, I. & Roth, D. 2021, *Recent Advances in Natural Language Processing via Large Pre-Trained Language Models: A Survey*, arXiv.

Mitra, R., Narechania, A., Endert, A. & Stasko, J. 2022, "Facilitating Conversational Interaction in Natural Language Interfaces for Visualization", .

Narechania, A., Srinivasan, A. & Stasko, J. 2021, "NL4DV: A Toolkit for Generating Analytic Specifications for Data Visualization from Natural Language Queries", *IEEE Transactions on*

Visualization and Computer Graphics, vol. 27, no. 2, pp. 369-379.

Oates, B.J. 2006, *Researching information systems and computing,* SAGE Publications, London.

Office for Students 2022a, Oct 7,-last update*, About the TEF*. Available: https://www.officeforstudents.org.uk/advice-and-guidance/teaching/about-the-tef/ [2022, December 18,].

Office for Students 2022b, Oct 6,-last update*, Access and participation plans*. Available: https://www.officeforstudents.org.uk/advice-and-guidance/promoting-equal-opportunities/access-and-participation-plans/ [2022, December 18,].

Office for Students 2022c, *Securing student success: Regulatory framework for higher education in England*, Office for Students.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D. & Sutskever, I. 2018, "Language Models are Unsupervised Multitask Learners", .

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W. & Liu, P.J. 2020, "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer", *Journal of machine learning research,* vol. 21.

Reimers, N. & Gurevych, I. 2019, "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks", .

Rush, A., Nguyen, V. & Klein, G. 2018, Apr 3,-last update*, The Annotated Transformer*. Available: https://nlp.seas.harvard.edu/2018/04/03/attention.html#encoder [2022, December 18,].

Satyanarayan, A., Moritz, D., Wongsuphasawat, K. & Heer, J. 2017, "Vega-Lite: A Grammar of Interactive Graphics", *IEEE Transactions on Visualization and Computer Graphics,* vol. 23, no. 1, pp. 341-350.

Setlur, V., Tory, M. & Djalali, A. 2019, "Inferencing Underspecified Natural Language Utterances in Visual Analysis", *Proceedings of Iui 2019,* , pp. 40-51.

Shin, R. & Van Durme, B. 2021, "Few-Shot Semantic Parsing with Language Models Trained On Code", .

Srinivasan, A., Nyapathy, N., Lee, B., Drucker, S.M., Stasko, J. & ASSOC COMP MACHINERY 2021, "Collecting and Characterizing Natural Language Utterances for Specifying Data Visualizations", Assoc Computing Machinery, NEW YORK.

Srinivasan, A. & Stasko, J. 2017, "Natural Language Interfaces for Data Analysis with Visualization: Considering What Has and Could Be Asked", *EuroVis 2017 - Short Papers,* .

Sun, Y., Leigh, J., Johnson, A. & Lee, S. 2010, "Articulate: A Semi-automated Model for Translating Natural Language Queries into Meaningful Visualizations" in *SMART GRAPHICS, PROCEEDINGS*, eds. R. Taylor, P. Boulanger, A. Kruger & P. Olivier, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 184-195.

Tableau Software 2022, *, Automatically Build Views with Ask Data*. Available: https://help.tableau.com/current/pro/desktop/en-us/ask_data.htm [2022, December 18,].

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L. & Polosukhin, I. 2017, "Attention is All You Need", *Proceedings of the 31st International Conference on Neural Information Processing Systems*Curran Associates Inc, Red Hook, NY, USA.

von Platen, P. 2020, Oct 8,-last update*, Transformers-based Encoder-Decoder Models*. Available: https://huggingface.co/blog/encoder-decoder [2022, December 18,].

Wang, B. & Komatsuzaki, A. 2021, *GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model*.

Yang, Y., Cer, D., Ahmad, A., Guo, M., Law, J., Constant, N., Abrego, G.H., Yuan, S., Tar, C., Sung, Y., Strope, B. & Kurzweil, R. 2019, "Multilingual Universal Sentence Encoder for Semantic Retrieval", .

Alammar, J. 2018, Jun 27,-last update, The Illustrated Transformer. Available: https://jalammar.github.io/illustrated-transformer/ [2022, December 18,].

Austin, J., Odena, A., Nye, M., Bosma, M., Michalewski, H., Dohan, D., Jiang, E., Cai, C., Terry, M., Le, Q. & Sutton, C. 2021, Program Synthesis with Large Language Models, arXiv.

Black, S., Gao, L., Wang, P., Leahy, C. & Biderman, S. 2021, "GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow", .

Bolton, P. 2021, Higher Education funding in England, House of Commons Libary.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J.D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I. & Amodei, D. 2020, "Language Models are Few-Shot Learners", Advances in Neural Information Processing Systems, eds. H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan & H. Lin, Curran Associates, Inc, , pp. 1877.

Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H.P.d.O., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., Ray, A., Puri, R., Krueger, G., Petrov, M., Khlaaf, H., Sastry, G., Mishkin, P., Chan, B., Gray, S., Ryder, N., Pavlov, M., Power, A., Kaiser, L., Bavarian, M., Winter, C., Tillet, P., Such, F.P., Cummings, D., Plappert, M., Chantzis, F., Barnes, E., Herbert-Voss, A., Guss, W.H., Nichol, A., Paino, A., Tezak, N., Tang, J., Babuschkin, I., Balaji, S., Jain, S., Saunders, W., Hesse, C., Carr, A.N., Leike, J., Achiam, J., Misra, V., Morikawa, E., Radford, A., Knight, M., Brundage, M., Murati, M., Mayer, K., Welinder, P., McGrew, B., Amodei, D., McCandlish, S., Sutskever, I. & Zaremba, W. 2021, Evaluating Large Language Models Trained on Code, arXiv.

Devlin, J., Chang, M., Lee, K. & Toutanova, K. 2019, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding", Proceedings of the 2019 Conference of the North {A}merican Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)Association for Computational Linguistics, Minneapolis, Minnesota, jun, pp. 4171.

Dodge, J., Sap, M., Marasović, A., Agnew, W., Ilharco, G., Groeneveld, D., Mitchell, M. & Gardner, M. 2021, "Documenting Large Webtext Corpora: A Case Study on the Colossal Clean Crawled Corpus", .

Gao, L., Biderman, S., Black, S., Golding, L., Hoppe, T., Foster, C., Phang, J., He, H., Thite, A., Nabeshima, N., Presser, S. & Leahy, C. 2021, The Pile: An 800GB Dataset of Diverse Text for Language Modeling, arXiv.

Gao, T., Dontcheva, M., Adar, E., Liu, Z., Karahalios, K. & ACM 2015, "DataTone: Managing Ambiguity in Natural Language Interfaces for Data Visualization", ACM, NEW YORK, pp. 489.

Hendrycks, D., Basart, S., Kadavath, S., Mazeika, M., Arora, A., Guo, E., Burns, C., Puranik, S., He, H., Song, D. & Steinhardt, J. 2021, Measuring Coding Challenge Competence With APPS, arXiv.

Hugging Face 2022, , Auto Classes. Available: https://huggingface.co/docs/transformers/model_doc/auto [2022, December 18,].

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V. & Zettlemoyer, L. 2019, "BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension", .

Li, Y., Choi, D., Chung, J., Kushman, N., Schrittwieser, J., Leblond, R., Eccles, T., Keeling, J., Gimeno, F., Lago, A.D., Hubert, T., Choy, P., d'Autume, C.d.M., Babuschkin, I., Chen, X., Huang, P., Welbl, J., Gowal, S., Cherepanov, A., Molloy, J., Mankowitz, D.J., Robson, E.S., Kohli, P., de Freitas, N., Kavukcuoglu, K. & Vinyals, O. 2022, Competition-Level Code Generation with AlphaCode, arXiv.

Liu, C., Han, Y., Jiang, R. & Yuan, X. 2021, "ADVISor: Automatic Visualization Answer for Natural-Language Question on Tabular Data", 2021 Ieee 14th Pacific Visualization Symposium (Pacificvis 2021), , pp. 11-20.

Lu, Y., Bartolo, M., Moore, A., Riedel, S. & Stenetorp, P. 2021, Fantastically Ordered Prompts and Where to Find Them: Overcoming Few-Shot Prompt Order Sensitivity, arXiv.

Luo, Y., Tang, N., Li, G., Chai, C., Li, W., Qin, X. & ASSOC COMP MACHINERY 2021, "Synthesizing Natural Language to Visualization (NL2VIS) Benchmarks from NL2SQL Benchmarks", Assoc Computing Machinery, NEW YORK, pp. 1235.

Luo, Y., Tang, N., Li, G., Tang, J., Chai, C. & Qin, X. 2022, "Natural Language to Visualization by Neural Machine Translation", IEEE Transactions on Visualization and Computer Graphics, vol. 28, no. 1, pp. 217-226.

McCoy, T., Pavlick, E. & Linzen, T. 2019, "Right for the Wrong Reasons: Diagnosing Syntactic Heuristics in Natural Language Inference", Proceedings of the 57th Annual Meeting of the Association for Computational LinguisticsAssociation for Computational Linguistics, Florence,

Italy, jul, pp. 3428.
Microsoft 2021, Feb 9,-last update, Q&A for Power BI business users. Available: https://learn.microsoft.com/en-us/power-bi/consumer/end-user-q-and-a [2022, December 18,].
Min, B., Ross, H., Sulem, E., Veyseh, A.P.B., Nguyen, T.H., Sainz, O., Agirre, E., Heinz, I. & Roth, D. 2021, Recent Advances in Natural Language Processing via Large Pre-Trained Language Models: A Survey, arXiv.
Mitra, R., Narechania, A., Endert, A. & Stasko, J. 2022, "Facilitating Conversational Interaction in Natural Language Interfaces for Visualization", .
Narechania, A., Srinivasan, A. & Stasko, J. 2021, "NL4DV: A Toolkit for Generating Analytic Specifications for Data Visualization from Natural Language Queries", IEEE Transactions on Visualization and Computer Graphics, vol. 27, no. 2, pp. 369-379.
Oates, B.J. 2006, Researching information systems and computing, SAGE Publications, London.
Office for Students 2022a, Oct 7,-last update, About the TEF. Available: https://www.officeforstudents.org.uk/advice-and-guidance/teaching/about-the-tef/ [2022, December 18,].
Office for Students 2022b, Oct 6,-last update, Access and participation plans. Available: https://www.officeforstudents.org.uk/advice-and-guidance/promoting-equal-opportunities/access-and-participation-plans/ [2022, December 18,].
Office for Students 2022c, Securing student success:
Regulatory framework for
higher education in England, Office for Students.
Radford, A., Wu, J., Child, R., Luan, D., Amodei, D. & Sutskever, I. 2018, "Language Models are Unsupervised Multitask Learners", .
Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W. & Liu, P.J. 2020, "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer", Journal of machine learning research, vol. 21.
Reimers, N. & Gurevych, I. 2019, "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks", .
Rush, A., Nguyen, V. & Klein, G. 2018, Apr 3,-last update, The Annotated Transformer. Available: https://nlp.seas.harvard.edu/2018/04/03/attention.html#encoder [2022, December 18,].
Satyanarayan, A., Moritz, D., Wongsuphasawat, K. & Heer, J. 2017, "Vega-Lite: A Grammar of Interactive Graphics", IEEE Transactions on Visualization and Computer Graphics, vol. 23, no. 1, pp. 341-350.
Setlur, V., Tory, M. & Djalali, A. 2019, "Inferencing Underspecified Natural Language Utterances in Visual Analysis", Proceedings of Iui 2019, , pp. 40-51.
Shin, R. & Van Durme, B. 2021, "Few-Shot Semantic Parsing with Language Models Trained On Code", .
Srinivasan, A., Nyapathy, N., Lee, B., Drucker, S.M., Stasko, J. & ASSOC COMP MACHINERY 2021, "Collecting and Characterizing Natural Language Utterances for Specifying Data Visualizations", Assoc Computing Machinery, NEW YORK.
Srinivasan, A. & Stasko, J. 2017, "Natural Language Interfaces for Data Analysis with Visualization: Considering What Has and Could Be Asked", EuroVis 2017 - Short Papers, .
Sun, Y., Leigh, J., Johnson, A. & Lee, S. 2010, "Articulate: A Semi-automated Model for Translating Natural Language Queries into Meaningful Visualizations" in SMART GRAPHICS, PROCEEDINGS, eds. R. Taylor, P. Boulanger, A. Kruger & P. Olivier, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 184-195.
Tableau Software 2022, , Automatically Build Views with Ask Data. Available: https://help.tableau.com/current/pro/desktop/en-us/ask_data.htm [2022, December 18,].
Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L. & Polosukhin, I. 2017, "Attention is All You Need", Proceedings of the 31st International Conference on Neural Information Processing SystemsCurran Associates Inc, Red Hook, NY, USA.
von Platen, P. 2020, Oct 8,-last update, Transformers-based Encoder-Decoder Models. Available: https://huggingface.co/blog/encoder-decoder [2022, December 18,].
Wang, B. & Komatsuzaki, A. 2021, GPT-J-6B: A 6 Billion Parameter Autoregressive Language

Model.
Yang, Y., Cer, D., Ahmad, A., Guo, M., Law, J., Constant, N., Abrego, G.H., Yuan, S., Tar, C., Sung, Y., Strope, B. & Kurzweil, R. 2019, "Multilingual Universal Sentence Encoder for Semantic Retrieval", .
Alammar, J. 2018, Jun 27,-last update, The Illustrated Transformer. Available: https://jalammar.github.io/illustrated-transformer/ [2022, December 18,].
Austin, J., Odena, A., Nye, M., Bosma, M., Michalewski, H., Dohan, D., Jiang, E., Cai, C., Terry, M., Le, Q. & Sutton, C. 2021, Program Synthesis with Large Language Models, arXiv.
Black, S., Gao, L., Wang, P., Leahy, C. & Biderman, S. 2021, "GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow", .
Bolton, P. 2021, Higher Education funding in England, House of Commons Libary.
Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J.D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I. & Amodei, D. 2020, "Language Models are Few-Shot Learners", Advances in Neural Information Processing Systems, eds. H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan & H. Lin, Curran Associates, Inc, , pp. 1877.
Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H.P.d.O., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., Ray, A., Puri, R., Krueger, G., Petrov, M., Khlaaf, H., Sastry, G., Mishkin, P., Chan, B., Gray, S., Ryder, N., Pavlov, M., Power, A., Kaiser, L., Bavarian, M., Winter, C., Tillet, P., Such, F.P., Cummings, D., Plappert, M., Chantzis, F., Barnes, E., Herbert-Voss, A., Guss, W.H., Nichol, A., Paino, A., Tezak, N., Tang, J., Babuschkin, I., Balaji, S., Jain, S., Saunders, W., Hesse, C., Carr, A.N., Leike, J., Achiam, J., Misra, V., Morikawa, E., Radford, A., Knight, M., Brundage, M., Murati, M., Mayer, K., Welinder, P., McGrew, B., Amodei, D., McCandlish, S., Sutskever, I. & Zaremba, W. 2021, Evaluating Large Language Models Trained on Code, arXiv.
Devlin, J., Chang, M., Lee, K. & Toutanova, K. 2019, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding", Proceedings of the 2019 Conference of the North {A}merican Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)Association for Computational Linguistics, Minneapolis, Minnesota, jun, pp. 4171.
Dodge, J., Sap, M., Marasović, A., Agnew, W., Ilharco, G., Groeneveld, D., Mitchell, M. & Gardner, M. 2021, "Documenting Large Webtext Corpora: A Case Study on the Colossal Clean Crawled Corpus", .
Fan, A., Lewis, M. & Dauphin, Y. 2018, "Hierarchical Neural Story Generation", .
Gao, L., Biderman, S., Black, S., Golding, L., Hoppe, T., Foster, C., Phang, J., He, H., Thite, A., Nabeshima, N., Presser, S. & Leahy, C. 2021, The Pile: An 800GB Dataset of Diverse Text for Language Modeling, arXiv.
Gao, T., Dontcheva, M., Adar, E., Liu, Z., Karahalios, K. & ACM 2015, "DataTone: Managing Ambiguity in Natural Language Interfaces for Data Visualization", ACM, NEW YORK, pp. 489.
Hendrycks, D., Basart, S., Kadavath, S., Mazeika, M., Arora, A., Guo, E., Burns, C., Puranik, S., He, H., Song, D. & Steinhardt, J. 2021, Measuring Coding Challenge Competence With APPS, arXiv.
Holtzman, A., Buys, J., Du, L., Forbes, M. & Choi, Y. 2019, "The Curious Case of Neural Text Degeneration", .
Hugging Face 2022, , Auto Classes. Available: https://huggingface.co/docs/transformers/model_doc/auto [2022, December 18,].
Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V. & Zettlemoyer, L. 2019, "BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension", .
Li, Y., Choi, D., Chung, J., Kushman, N., Schrittwieser, J., Leblond, R., Eccles, T., Keeling, J., Gimeno, F., Lago, A.D., Hubert, T., Choy, P., d'Autume, C.d.M., Babuschkin, I., Chen, X., Huang, P., Welbl, J., Gowal, S., Cherepanov, A., Molloy, J., Mankowitz, D.J., Robson, E.S., Kohli, P., de Freitas, N., Kavukcuoglu, K. & Vinyals, O. 2022, Competition-Level Code Generation with AlphaCode, arXiv.
Liu, C., Han, Y., Jiang, R. & Yuan, X. 2021, "ADVISor: Automatic Visualization Answer for

Natural-Language Question on Tabular Data", 2021 Ieee 14th Pacific Visualization Symposium (Pacificvis 2021), , pp. 11-20.
Lu, Y., Bartolo, M., Moore, A., Riedel, S. & Stenetorp, P. 2021, Fantastically Ordered Prompts and Where to Find Them: Overcoming Few-Shot Prompt Order Sensitivity, arXiv.
Luo, Y., Tang, N., Li, G., Chai, C., Li, W., Qin, X. & ASSOC COMP MACHINERY 2021, "Synthesizing Natural Language to Visualization (NL2VIS) Benchmarks from NL2SQL Benchmarks", Assoc Computing Machinery, NEW YORK, pp. 1235.
Luo, Y., Tang, N., Li, G., Tang, J., Chai, C. & Qin, X. 2022, "Natural Language to Visualization by Neural Machine Translation", IEEE Transactions on Visualization and Computer Graphics, vol. 28, no. 1, pp. 217-226.
McCoy, T., Pavlick, E. & Linzen, T. 2019, "Right for the Wrong Reasons: Diagnosing Syntactic Heuristics in Natural Language Inference", Proceedings of the 57th Annual Meeting of the Association for Computational LinguisticsAssociation for Computational Linguistics, Florence, Italy, jul, pp. 3428.
Microsoft 2021, Feb 9,-last update, Q&A for Power BI business users. Available: https://learn.microsoft.com/en-us/power-bi/consumer/end-user-q-and-a [2022, December 18,].
Min, B., Ross, H., Sulem, E., Veyseh, A.P.B., Nguyen, T.H., Sainz, O., Agirre, E., Heinz, I. & Roth, D. 2021, Recent Advances in Natural Language Processing via Large Pre-Trained Language Models: A Survey, arXiv.
Mitra, R., Narechania, A., Endert, A. & Stasko, J. 2022, "Facilitating Conversational Interaction in Natural Language Interfaces for Visualization", .
Narechania, A., Srinivasan, A. & Stasko, J. 2021, "NL4DV: A Toolkit for Generating Analytic Specifications for Data Visualization from Natural Language Queries", IEEE Transactions on Visualization and Computer Graphics, vol. 27, no. 2, pp. 369-379.
Oates, B.J. 2006, Researching information systems and computing, SAGE Publications, London.
Office for Students 2022a, Oct 7,-last update, About the TEF. Available: https://www.officeforstudents.org.uk/advice-and-guidance/teaching/about-the-tef/ [2022, December 18,].
Office for Students 2022b, Oct 6,-last update, Access and participation plans. Available: https://www.officeforstudents.org.uk/advice-and-guidance/promoting-equal-opportunities/access-and-participation-plans/ [2022, December 18,].
Office for Students 2022c, Securing student success:
Regulatory framework for
higher education in England, Office for Students.
Radford, A., Wu, J., Child, R., Luan, D., Amodei, D. & Sutskever, I. 2018, "Language Models are Unsupervised Multitask Learners", .
Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W. & Liu, P.J. 2020, "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer", Journal of machine learning research, vol. 21.
Reimers, N. & Gurevych, I. 2019, "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks", .
Ren, S., Guo, D., Lu, S., Zhou, L., Liu, S., Tang, D., Sundaresan, N., Zhou, M., Blanco, A. & Ma, S. 2020, "CodeBLEU: a Method for Automatic Evaluation of Code Synthesis", .
Rush, A., Nguyen, V. & Klein, G. 2018, Apr 3,-last update, The Annotated Transformer. Available: https://nlp.seas.harvard.edu/2018/04/03/attention.html#encoder [2022, December 18,].
Satyanarayan, A., Moritz, D., Wongsuphasawat, K. & Heer, J. 2017, "Vega-Lite: A Grammar of Interactive Graphics", IEEE Transactions on Visualization and Computer Graphics, vol. 23, no. 1, pp. 341-350.
Setlur, V., Tory, M. & Djalali, A. 2019, "Inferencing Underspecified Natural Language Utterances in Visual Analysis", Proceedings of Iui 2019, , pp. 40-51.
Shin, R. & Van Durme, B. 2021, "Few-Shot Semantic Parsing with Language Models Trained On Code", .
Srinivasan, A., Nyapathy, N., Lee, B., Drucker, S.M., Stasko, J. & ASSOC COMP MACHINERY 2021, "Collecting and Characterizing Natural Language Utterances for Specifying Data

Visualizations", Assoc Computing Machinery, NEW YORK.
Srinivasan, A. & Stasko, J. 2017, "Natural Language Interfaces for Data Analysis with Visualization: Considering What Has and Could Be Asked", EuroVis 2017 - Short Papers, .
Sun, Y., Leigh, J., Johnson, A. & Lee, S. 2010, "Articulate: A Semi-automated Model for Translating Natural Language Queries into Meaningful Visualizations" in SMART GRAPHICS, PROCEEDINGS, eds. R. Taylor, P. Boulanger, A. Kruger & P. Olivier, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 184-195.
Tableau Software 2022, , Automatically Build Views with Ask Data. Available: https://help.tableau.com/current/pro/desktop/en-us/ask_data.htm [2022, December 18,].
Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L. & Polosukhin, I. 2017, "Attention is All You Need", Proceedings of the 31st International Conference on Neural Information Processing SystemsCurran Associates Inc, Red Hook, NY, USA.
von Platen, P. 2020, Oct 8,-last update, Transformers-based Encoder-Decoder Models. Available: https://huggingface.co/blog/encoder-decoder [2022, December 18,].
Wang, B. & Komatsuzaki, A. 2021, GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model.
Wang, C., Cho, K. & Gu, J. 2019, "Neural Machine Translation with Byte-Level Subwords", .
Wiher, G., Meister, C. & Cotterell, R. 2022, "On Decoding Strategies for Neural Text Generators", Transactions of the Association for Computational Linguistics, vol. 10, pp. 997-1012.
Yang, Y., Cer, D., Ahmad, A., Guo, M., Law, J., Constant, N., Abrego, G.H., Yuan, S., Tar, C., Sung, Y., Strope, B. & Kurzweil, R. 2019, "Multilingual Universal Sentence Encoder for Semantic Retrieval", .
Zhang, H., Duckworth, D., Ippolito, D. & Neelakantan, A. 2020, "Trading Off Diversity and Quality in Natural Language Generation", .
Austin, J., Odena, A., Nye, M., Bosma, M., Michalewski, H., Dohan, D., Jiang, E., Cai, C., Terry, M., Le, Q. & Sutton, C. 2021, *Program Synthesis with Large Language Models*, arXiv.
Black, S., Gao, L., Wang, P., Leahy, C. & Biderman, S. 2021, "GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow", .
Bolton, P. 2021, *Higher Education funding in England*, House of Commons Libary.
Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J.D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I. & Amodei, D. 2020, "Language Models are Few-Shot Learners", *Advances in Neural Information Processing Systems*, eds. H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan & H. Lin, Curran Associates, Inc, , pp. 1877.
Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H.P.d.O., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., Ray, A., Puri, R., Krueger, G., Petrov, M., Khlaaf, H., Sastry, G., Mishkin, P., Chan, B., Gray, S., Ryder, N., Pavlov, M., Power, A., Kaiser, L., Bavarian, M., Winter, C., Tillet, P., Such, F.P., Cummings, D., Plappert, M., Chantzis, F., Barnes, E., Herbert-Voss, A., Guss, W.H., Nichol, A., Paino, A., Tezak, N., Tang, J., Babuschkin, I., Balaji, S., Jain, S., Saunders, W., Hesse, C., Carr, A.N., Leike, J., Achiam, J., Misra, V., Morikawa, E., Radford, A., Knight, M., Brundage, M., Murati, M., Mayer, K., Welinder, P., McGrew, B., Amodei, D., McCandlish, S., Sutskever, I. & Zaremba, W. 2021, *Evaluating Large Language Models Trained on Code*, arXiv.
Cox, K., Grinter, R.E., Hibino, S.L., Jagadeesan, L.J. & Mantilla, D. 2001, "A Multi-Modal Natural Language Interface to an Information Visualization Environment", *International Journal of Speech Technology,* vol. 4, no. 3, pp. 297-314.
Devlin, J., Chang, M., Lee, K. & Toutanova, K. 2019, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding", *Proceedings of the 2019 Conference of the North {A}merican Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*Association for Computational Linguistics, Minneapolis, Minnesota, jun, pp. 4171.
Dietterich, T.G. 1998, "Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms", *Neural computation,* vol. 10, no. 7, pp. 1895-1923.
Fan, A., Lewis, M. & Dauphin, Y. 2018, "Hierarchical Neural Story Generation", .

Freedman, D., Pisani, R. & Purves, R. *Statistics,* 4th Edition edn, W. W. Norton & Company Ltd.

Gao, L., Biderman, S., Black, S., Golding, L., Hoppe, T., Foster, C., Phang, J., He, H., Thite, A., Nabeshima, N., Presser, S. & Leahy, C. 2021, *The Pile: An 800GB Dataset of Diverse Text for Language Modeling*, arXiv.

Gao, T., Dontcheva, M., Adar, E., Liu, Z., Karahalios, K. & ACM 2015, "DataTone: Managing Ambiguity in Natural Language Interfaces for Data Visualization", ACM, NEW YORK, pp. 489.

Holtzman, A., Buys, J., Du, L., Forbes, M. & Choi, Y. 2019, "The Curious Case of Neural Text Degeneration", .

Hugging Face 2022, *, Auto Classes*. Available: https://huggingface.co/docs/transformers/model_doc/auto [2022, December 18,].

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V. & Zettlemoyer, L. 2019, "BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension", .

Li, Y., Choi, D., Chung, J., Kushman, N., Schrittwieser, J., Leblond, R., Eccles, T., Keeling, J., Gimeno, F., Lago, A.D., Hubert, T., Choy, P., d'Autume, C.d.M., Babuschkin, I., Chen, X., Huang, P., Welbl, J., Gowal, S., Cherepanov, A., Molloy, J., Mankowitz, D.J., Robson, E.S., Kohli, P., de Freitas, N., Kavukcuoglu, K. & Vinyals, O. 2022, *Competition-Level Code Generation with AlphaCode*, arXiv.

Liu, C., Han, Y., Jiang, R. & Yuan, X. 2021, "ADVISor: Automatic Visualization Answer for Natural-Language Question on Tabular Data", *2021 Ieee 14th Pacific Visualization Symposium (Pacificvis 2021),* , pp. 11-20.

Luo, Y., Tang, N., Li, G., Chai, C., Li, W., Qin, X. & ASSOC COMP MACHINERY 2021, "Synthesizing Natural Language to Visualization (NL2VIS) Benchmarks from NL2SQL Benchmarks", Assoc Computing Machinery, NEW YORK, pp. 1235.

Luo, Y., Tang, N., Li, G., Tang, J., Chai, C. & Qin, X. 2022, "Natural Language to Visualization by Neural Machine Translation", *IEEE Transactions on Visualization and Computer Graphics,* vol. 28, no. 1, pp. 217-226.

Microsoft 2021, Feb 9,-last update*, Q&A for Power BI business users*. Available: https://learn.microsoft.com/en-us/power-bi/consumer/end-user-q-and-a [2022, December 18,].

Min, B., Ross, H., Sulem, E., Veyseh, A.P.B., Nguyen, T.H., Sainz, O., Agirre, E., Heinz, I. & Roth, D. 2021, *Recent Advances in Natural Language Processing via Large Pre-Trained Language Models: A Survey*, arXiv.

Narechania, A., Srinivasan, A. & Stasko, J. 2021, "NL4DV: A Toolkit for Generating Analytic Specifications for Data Visualization from Natural Language Queries", *IEEE Transactions on Visualization and Computer Graphics,* vol. 27, no. 2, pp. 369-379.

Office for Students 2022a, Oct 7,-last update*, About the TEF*. Available: https://www.officeforstudents.org.uk/advice-and-guidance/teaching/about-the-tef/ [2022, December 18,].

Office for Students 2022b, Oct 6,-last update*, Access and participation plans*. Available: https://www.officeforstudents.org.uk/advice-and-guidance/promoting-equal-opportunities/access-and-participation-plans/ [2022, December 18,].

Office for Students 2022c, *Securing student success:
Regulatory framework for
higher education in England*, Office for Students.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D. & Sutskever, I. 2018, "Language Models are Unsupervised Multitask Learners", .

Reimers, N. & Gurevych, I. 2019, "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks", .

Rush, A., Nguyen, V. & Klein, G. 2018, Apr 3,-last update*, The Annotated Transformer*. Available: https://nlp.seas.harvard.edu/2018/04/03/attention.html#encoder [2022, December 18,].

Satyanarayan, A., Moritz, D., Wongsuphasawat, K. & Heer, J. 2017, "Vega-Lite: A Grammar of Interactive Graphics", *IEEE Transactions on Visualization and Computer Graphics,* vol. 23, no. 1, pp. 341-350.

Setlur, V., Tory, M. & Djalali, A. 2019, "Inferencing Underspecified Natural Language Utterances in

Visual Analysis", *Proceedings of Iui 2019,* , pp. 40-51.

Shin, R. & Van Durme, B. 2021, "Few-Shot Semantic Parsing with Language Models Trained On Code", .

Srinivasan, A., Nyapathy, N., Lee, B., Drucker, S.M., Stasko, J. & ASSOC COMP MACHINERY 2021, "Collecting and Characterizing Natural Language Utterances for Specifying Data Visualizations", Assoc Computing Machinery, NEW YORK.

Srinivasan, A. & Stasko, J. 2017, "Natural Language Interfaces for Data Analysis with Visualization: Considering What Has and Could Be Asked", *EuroVis 2017 - Short Papers,* .

Sun, Y., Leigh, J., Johnson, A. & Lee, S. 2010, "Articulate: A Semi-automated Model for Translating Natural Language Queries into Meaningful Visualizations" in *SMART GRAPHICS, PROCEEDINGS*, eds. R. Taylor, P. Boulanger, A. Kruger & P. Olivier, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 184-195.

Tableau Software 2022, *, Automatically Build Views with Ask Data*. Available: https://help.tableau.com/current/pro/desktop/en-us/ask_data.htm [2022, December 18,].

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L. & Polosukhin, I. 2017, "Attention is All You Need", *Proceedings of the 31st International Conference on Neural Information Processing Systems*Curran Associates Inc, Red Hook, NY, USA.

Wang, B. & Komatsuzaki, A. 2021, *GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model*.

Wiher, G., Meister, C. & Cotterell, R. 2022, "On Decoding Strategies for Neural Text Generators", *Transactions of the Association for Computational Linguistics,* vol. 10, pp. 997-1012.

Wu, Z. & Palmer, M. 1994, "Verb Semantics and Lexical Selection", *32nd Annual Meeting of the Association for Computational Linguistics: Proceedings of the Conference,* , pp. 133-138.

Yang, Y., Cer, D., Ahmad, A., Guo, M., Law, J., Constant, N., Abrego, G.H., Yuan, S., Tar, C., Sung, Y., Strope, B. & Kurzweil, R. 2019, "Multilingual Universal Sentence Encoder for Semantic Retrieval", .

Zhang, H., Duckworth, D., Ippolito, D. & Neelakantan, A. 2020, "Trading Off Diversity and Quality in Natural Language Generation", .