Natural Language Engineering (2024), **30**, pp. 943–972
doi:10.1017/S1351324923000438

ARTICLE

Improving short text classification with augmented data using GPT-3

Salvador V. Balkus[1] and Donghui Yan[2]

[1]Program in Data Science, University of Massachusetts Dartmouth, Dartmouth, MA, USA and [2]Department of Mathematics, University of Massachusetts Dartmouth, Dartmouth, MA, USA
Corresponding author: Salvador V. Balkus; Email: sbalkus@g.harvard.edu

(Received 22 May 2022; revised 21 April 2023; accepted 20 July 2023; first published online 25 August 2023)

Abstract

GPT-3 is a large-scale natural language model developed by OpenAI that can perform many different tasks, including topic classification. Although researchers claim that it requires only a small number of in-context examples to learn a task, in practice GPT-3 requires these training examples to be either of exceptional quality or a higher quantity than easily created by hand. To address this issue, this study teaches GPT-3 to classify whether a question is related to data science by augmenting a small training set with additional examples generated by GPT-3 itself. This study compares two augmented classifiers: the Classification Endpoint with an increased training set size and the Completion Endpoint with an augmented prompt optimized using a genetic algorithm. We find that data augmentation significantly increases the accuracy of both classifiers, and that the embedding-based Classification Endpoint achieves the best accuracy of about 76%, compared to human accuracy of 85%. In this way, giving large language models like GPT-3 the ability to propose their own training examples can improve short text classification performance.

Keywords: GPT-3; Data augmentation; Text classification; Machine learning

1. Introduction

Text messages, social media posts, emails, and internet comments are just a few examples of "short text" data that people use to communicate every day (Song *et al.* 2014). Because of their ubiquity, building natural language processing (NLP) models that can classify the topic or category of short text is an important business problem (Li *et al.* 2020). For example, classification models could automatically detect and censor offensive text, identify when customers ask questions that should be forwarded to a specific contact, or determine the sentiment of a message—whether it is considered positive or negative. Previously, building text classification models has been challenging as it requires collecting vast quantities of training examples to learn subtle differences in context—a quantity that may not have been feasible to collect for individuals, academics, or small businesses.

In recent years, however, new developments have made text classification accessible for everyone. *Transfer learning* for NLP is a process by which researchers pretrain a large neural network on vast quantities of unstructured text data to be easily tuned or generalized to many different types of problems such as text classification, question answering, and summarization. Some examples of pretrained transfer learning models include Google's BERT (Devlin *et al.* 2019) and Microsoft's Turing NLR (Bajaj *et al.* 2022), as well as OpenAI's GPT-3, a public model released in November 2020 that is the main subject of this work (Brown *et al.* 2020; Dale 2020; Pilipiszyn 2021).

Since this study began, the public GPT-3 API has offered two different methods of text classification, both of which involve trade-offs. The **Completion Endpoint** uses a text prompt followed by example-label pairs as input. It only requires a few (5–10) examples, but its performance is highly sensitive to *which* examples are included in the context. The **Classification Endpoint** uses text embedding with a JSON file of example-label pairs. It has more reliable performance, but requires hundreds or thousands of examples to perform well—more than could be quickly be constructed by hand. After our research was conducted, OpenAI merged the Classification Endpoint into the Embeddings Endpoint, but the functionality remains similar (OpenAI 2022).

Because GPT-3 requires much less training data to learn a specific task than a typical language model, Brown *et al.* (2020) describe it as "few-shot learning." That means developers can use GPT-3 to solve text classification problems without needing large domain-specific sets of training examples. Developers have already created dozens of applications using GPT-3, and this number will likely only grow (Pilipiszyn 2021). However, because of the trade-off between the Completion Endpoint's sensitivity to example quality and Classification Endpoint's still relatively large training data requirement, the few-shot learning ability of GPT-3 is currently limited for text classification. Those who have access to only small datasets with fewer than 100 observations need a way to use GPT-3 and other transfer learning models more effectively.

This raises the research question of this study: **how can we construct a better set of training examples to improve GPT-3's few-shot learning capabilities?** This process is often called *data augmentation*, and we are especially interested in its practical application to empirical text classification problems faced by organizations with small datasets.

This study evaluates two potential optimization-based data augmentation methods. Both involve using GPT-3 to generate new questions of a given class based on the existing ones. The first method increases the *quantity* of the sample size for the embedding-based Classification Endpoint, with the goal of improving accuracy after optimizing hyperparameters. The second selects optimal generated training examples to be included in the Completion Endpoint using a genetic algorithm, in hopes of improving the *quality* of the input context.

We evaluate these two methods in a case study: predicting question topics related to data science in the English language. At the University of Massachusetts Dartmouth, a student organization called the Big Data Club collected questions asked by students through the organization's server on Discord, an instant messaging application. To help track which members participated the most in the group, the Big Data Club wanted to use GPT-3 to identify when members asked questions related to data science, the group's academic focus. Using data from this case study, our research evaluates the performance of two data augmentation methods with the GPT-3 Classification and Completion Endpoints for solving a practical, real-world text classification problem.

By providing methods to train natural language models on small short text datasets, our research benefits developers looking to apply GPT-3 and other pretrained generative transfer learning models in practice. Previous studies on data augmentation techniques for NLP do not leverage GPT-3 and are typically not evaluated on practical classification problems beyond highly sanitized NLP benchmarks. They also focus on generating examples that are highly similar to existing training data, rather than formulating new, diverse samples and selecting the ones that provide the most valuable training. Overall, the goal of this study is to evaluate whether GPT-3 can be used to improve its own performance, rather than relying on external methods. This would promote the implementation and widespread use of NLP models for automating mundane business tasks requiring text classification, from commercial chatbots to the detection of harmful or misleading social media posts.

The remaining content is summarized as follows. Section 2 reviews the existing literature on data augmentation for NLP problems and provides readers with necessary background information on GPT-3 and the other methods we use for augmentation. In Section 3, we explain the workings of the proposed augmentation methods implemented in this study. Section 4 describes how these methods were evaluated, including the data and model parameters. Then, Section 5 presents our evaluation results, while a discussion on these results, their limitations, and potential future work is included in Section 6. Finally, Section 7 summarizes our conclusions.

2. Background and related work

2.1 Natural language processing and transfer learning

Text classification is defined as "the procedure of designating predefined labels for text" (Li *et al.* 2020). Language classifiers take tokens—the most basic components of words—as input in order to yield labels as output. Previously, traditional machine learning models such as the Naive Bayes, Support Vector Machine, K-Nearest Neighbor, and Random Forest algorithms have been combined with text representation methods such as N-gram, Bag-Of-Words, and word2vec for text classification (Kowsari, Heidarysafa, and Mendu 2019; Li *et al.* 2020). However, these methods often require feature engineering and suffer performance limitations. Classification is especially challenging for short text because of the sparseness of the training examples, which contain very few words to inform the model (Song *et al.* 2014). This is also the reason that techniques like keyword or topic extraction (Onan, Korukoğlu, and Bulut 2016; Onan 2019b) are more difficult to use in short text, as they typically rely on characteristics only present in longer documents.

More recent research focuses on deep learning, which uses large-scale neural networks to better capture the complex, nonlinear relationships between words (Zulqarnain *et al.* 2020; Minaee *et al.* 2022). However, deep neural networks often require time consuming and expensive training processes to develop. To overcome this problem, researchers have implemented the technique of transfer learning (Pan and Yang 2010). Transfer learning for NLP involves constructing a pretrained deep learning model for some general task, which can then be fine-tuned for a specific task using a smaller dataset (Dai and Le 2015; Radford *et al.* 2018). Using pretraining reduces the time and data necessary to achieve quality performance in a model. It can even allow the construction of "few-shot" learning models, which require only a small number of training examples to attain acceptable performance.

Transfer learning models for NLP are trained to predict tokens in a sequence of text. Training on this task allows large language models to generate text by repeatedly predicting a sequence of tokens given an input (Radford *et al.* 2018; Devlin *et al.* 2019; Brown *et al.* 2020). To develop a transfer learning model, a neural network with millions or billions of neurons is trained on a large corpus of unstructured text data, usually from the internet, to predict masked tokens on unseen passages. This allows the model to, in essence, learn the relationships between thousands of different words in different contexts. Then this model, in turn, can be applied to numerous natural language problems either through fine-tuning or by providing a prompt structured in a specific pattern, which the model will attempt to continue (Devlin *et al.* 2019; Brown *et al.* 2020).

Some transfer learning models for NLP include BERT (Devlin *et al.* 2019), which has spawned numerous spin-off models (Liu *et al.* 2019; Adhikari *et al.* 2019); GPT-3 (Brown *et al.* 2020), which has been deployed for dozens of commercial products; and even more recent models like ERNiE (Sun *et al.* 2021), ST-MoE (Zoph *et al.* 2022), and Turing NLR (Bajaj *et al.* 2022). Our paper focuses on GPT-3 since it was made available to the public through an API in 2021, allowing anyone to use it for practical applications.

2.2 GPT-3

GPT-3 is a recent transfer learning model developed for natural language problems (Brown *et al.* 2020). Its deep neural network architecture features layers of *transformers*, which are deep learning layers that use self-attention (modeling the relationship between each word and all other words in the sequence) to learn the complex relationships between words in different contexts (Vaswani *et al.* 2017). Several versions of the model were trained, with the largest having up to 175 billion neurons. The model was trained on the Common Crawl dataset (Brown *et al.* 2020; Raffel *et al.* 2022), which contains unstructured text data scraped from the web, including sites like Wikipedia.

GPT-3 is commercially available through an application programming interface (API) offered by OpenAI (Pilipiszyn 2021). It features several different engines, of which this study considers two: **ada** and **davinci**. The **ada** engine provides the fastest and cheapest inference but limited predictive performance, while **davinci** offers the most accurate predictions, but is the most expensive and is limited by slow inference time.

Short text classification can be performed using the OpenAI GPT-3 API in two main ways. The first is the **Completion Endpoint**, which is traditionally used for generating human-sounding text given a prompt (OpenAI 2021b). By structuring the prompt in a specific pattern, the GPT-3 Completion Endpoint can also be used for classification. If users provide a prompt that consists of a series of examples, each followed by a label, and leave the label of the final example blank the Completion Endpoint will follow the pattern provided by the user and attempt to predict the label of the final example. Users can also restrict the output to ensure the only possible token outputs are the known classes. This process is explained further in Section 3.2 and depicted visually in Figure 2. However, because the input is restricted to 2,049 tokens, and since more tokens induce greater expense, the number of possible training examples that can be used with the Completion Endpoint is limited.

The second short text classification method with GPT-3 is the **Classification Endpoint**. This method, designed specifically for classification, converts sequences of tokens into vectors of real numbers. When called to classify a short text input, the Classification Endpoint searches for examples most semantically similar to the input (based on their distance), ranks them based on their relevance, and then selects the class with the highest likelihood of occurring based on the labels of the most similar examples (OpenAI 2021a). OpenAI recently merged its function into the Embeddings Endpoint OpenAI (2022).

Both the Completion and Classification Endpoints enable users to develop *few-shot* natural language classification models—in other words, GPT-3 requires only a small number of examples to learn a domain-specific task, rather than the terabytes of data necessary for the original pretraining. At the time of release, GPT-3 had achieved state-of-the-art performance on the SuperGLUE natural language benchmark tasks (Brown *et al.* 2020), including reading comprehension and question answering. Current research has applied GPT-3 to other tasks, such as understanding and drafting emails (Thiergart, Huber, and Ubellacker 2021). It also attracted media attention due to its human-like performance in writing fake news articles (GPT-3 2020).

Despite its promise, using GPT-3 is still challenging. The performance of GPT-3 highly depends on the choice of examples provided in the few-shot learning scenario (Zhao *et al.* 2021; Liu *et al.* 2022). Hence, a method for training example selection is necessary. *Contextual calibration* provides a method to select where in the prompt to place different answers to avoid instability in responses (Zhao *et al.* 2021). However, this method only selects *where* to place the example, not which example to use. KATE, proposed by Liu *et al.* (2022), selects optimal in-context examples for the Completion Endpoint by retrieving examples semantically similar to the test sample before constructing the prompt for GPT-3. However, this requires modifying the prompt of GPT-3 for each prediction, may result in very expensive API calls, and is already performed efficiently by the Classification Endpoint.

As such, a better method is needed for improving the classification performance of GPT-3 outside of the context of benchmarks like SuperGLUE (Wang *et al.* 2019) that contain many accurately labeled snippets of text. What happens if the user does not have a dataset with a large number of labeled training examples? In this case, data augmentation methods can improve the performance of GPT-3.

2.3 Data augmentation

Data augmentation is the process of improving a model's input data by "selecting important samples, adding more useful data samples or adding extra information to adapt the model" to a specific domain (Guo and Yu 2022). Previous research has proposed several techniques, though compared to other domains such as image processing (Shorten and Khoshgoftaar 2019), augmentation of text data is still in its infancy.

Guo and Yu (2022) survey the types of domain adaptation available for large natural language models, including data augmentation. One type of data augmentation is *importance sampling*, which attempts to select only relevant samples for training to improve performance. Another is *pseudo-labeling*, which attempts to increase the amount of training data by applying pseudo-labels to previously unlabeled data. An example of this is the SentAugment technique, which retrieves previously unlabeled sentences from a text bank and labels them to increase the number of training examples (Du *et al.* 2020). Finally, *prompting* provides additional information such as task descriptions to the language model—which is how the GPT-3 Completion Endpoint to classifies text.

As Guo and Yu (2022) discuss, input data can also be augmented using adversarial training. Qu *et al.* (2020) discuss several variations of data augmentation for text which rely on adversarial training methods to generate new examples. They also propose CoDA, which combines data transformations to augment training data. Though successful, these methods are complex to implement and do not necessarily allow the model to improve by proposing examples that are *better* than the existing ones.

Feng *et al.* (2021) also survey data augmentation for NLP and describe a multitude of specific methods that have been developed to generate new training examples. These include rule-based approaches, interpolation approaches which adopt mixup (Zhang *et al.* 2018) for NLP, techniques for modifying existing text like Backtranslation (Sennrich, Haddow, and Birch 2016), and even techniques which generate new text.

Several cutting-edge data augmentation approaches rely heavily on the generative capabilities of transfer learning models to expand the number of training samples. Kobayashi (2018) discusses contextual augmentation, where words in existing examples are modified. LAMBADA, proposed by Anaby-Tavor *et al.* (2020), generates new labeled data using GPT-2, filters the new data to reduce noise and potential error, and then provides it as input to another language model. Similarly, GPT3Mix uses GPT-3 to select samples from training data and generate additional samples by mixing previous sentence components together into new, yet still plausible, examples (Yoo *et al.* 2021). Quteineh *et al.* (2020) use Monte Carlo Tree Search to generate optimal examples for model training. Generating new training data effectively combines the ideas behind importance sampling, pseudo-labeling, and prompting into one powerful technique.

Kumar *et al.* (2020) extend these ideas by using LAMBADA with a variety of NLP transfer learning models—including BERT, GPT-2, and BART—to augment the training data of multiple text classifiers. Evaluating training sets with 10 examples per class, they showed that pairing each sample in the training data with one additional synthetic sample improved performance on NLP benchmarks for sentiment analysis, intent classification, and question classification. Each language model evaluated achieved similar accuracies across tasks. They also evaluated the semantic fidelity, or how well-generated examples retained the meaning and class information of the input examples, for each model. They noted that GPT-2 exhibited much lower semantic fidelity than other models.

One interesting feature of the previous text data augmentation literature is its focus on trying to generate new examples that are very similar to preexisting ones. Kumar *et al.* (2020) even imply that GPT's inability to preserve the meaning of existing examples is a weakness. Yet, why should augmented examples try to duplicate existing examples in meaning? If language models can generate entirely new questions in a given class that are completely unrelated to existing training examples, would that not be beneficial in providing more information about the problem? We hypothesize that such new questions, as long as they preserve enough labels, would expand the coverage of the training data and help better capture unique edge cases within classes.

Therefore, inspired by Kumar *et al.* (2020), our study attempts to develop an improved method of data augmentation. First, we leverage the newer GPT-3 model, which has much better text-generation capabilities. Second, our training data focuses on a practical, empirical case study with highly limited data availability and a specific domain focus, which goes beyond the highly sanitary NLP benchmark datasets used in the literature. Third, we evaluate the model's performance after generating far more synthetic examples than previous studies—up to 10,000 more new examples for the Classification Endpoint. And finally, unlike Kumar *et al.* (2020) and Anaby-Tavor *et al.* (2020), we allow GPT-3 to creatively generate any example, allowing new examples to be unrelated to old ones. To ensure only quality examples are selected for the Completion Endpoint, we evaluate a genetic algorithm that select the highest-quality examples based on how much they improve the classifier's performance.

2.4 Genetic algorithms

2.4.1 Motivation

As discussed earlier, GPT-3 is sensitive to the in-context examples selected as training data—especially the Completion Endpoint, which must use limited examples since larger inputs are more expensive and their size is capped at 2049 tokens. Using the GPT-3 Completion Endpoint for few-shot natural language classification requires selecting the best examples to include in its context. This amounts to an optimization problem, wherein the feature space is defined by text examples rather than numeric values. Because of this, traditional optimization algorithms such as gradient descent cannot be applied directly.

Most modern machine learning techniques for text classification overcome this problem by relying on embeddings (Minaee *et al.* 2022), which transform sequences of text into numerical vectors based on their similarly. In fact, the GPT-3 Classification Endpoint uses embeddings to select the best examples for classification. When trying to select examples that optimize the predictive performance of the Completion Endpoint; however, we hypothesize that embeddings might not be the best choice. The acute sensitivity of the Completion Endpoint to the input means that its performance can behave erratically as a function of the input. The function is nonsmooth, and the global maximum could occur anywhere on the discrete gradient. Even two prompts that are similar semantically can produce very different few-shot performance.

Genetic algorithms overcome this problem. Erratic functions (like we believe the Completion Endpoint to be) contain many local optima that can trap typical optimization algorithms, preventing them from finding the global optimum. Genetic algorithms are designed to avoid getting trapped in local optima, and unlike other popular techniques, they operate even on nonsmooth functions where the overall gradient does not necessarily inform the location of the global optimum. Selecting the optimal subset of training examples is a problem more closely related to NP-Hard combinatorial problems, at which genetic algorithms excel (Katoch, Chauhan, and Kumar 2020), rather than function optimization. This is why we believe it might be more appropriate than an embedding-based approach.

In NLP, genetic algorithms have been applied for feature selection problems in text classification, especially problems like our own that involve challenges with embeddings (Deng *et al.* 2018). Chen *et al.* (2021) employ genetic algorithms for extractive summarization due to their ability to be customized to specific problem domains. Onan, Korukoğlu, and Bulut use genetic algorithms and other evolutionary and similar meta-heuristic algorithms for optimizing feature selection and ensemble pruning in multiple types of text classification tasks (Onan and Korukoğlu 2016; Onan, Korukoğlu, and Bulut 2017; Onan 2018). Hence, we believe they could be applicable to our similar problem.

2.4.2 Overview

A *genetic algorithm* is an iterative optimization technique inspired by the biological mechanism of natural selection. The algorithm initializes and maintains a population of potential solution candidates. At each iteration (termed *generation*), the algorithm evaluates the *fitness* of each candidate, which is the value to be optimized. The candidates with the highest fitness are preserved, and their genetic information is recombined using a crossover operator to produce genetically similar offspring, which are evaluated at the next iteration. As this process continues, the fitness of the population rises, thereby maximizing the fitness function (Srinivas and Patnaik 1994; Katoch *et al.* 2020).

In a genetic algorithm, each candidate is defined as a set of *alleles*, each of which describes a feature of the candidate. Genetic algorithms also feature several *operators* which are applied at each iteration. After the population is initialized, they are applied in the following order (Srinivas and Patnaik 1994; Katoch *et al.* 2020):

1. *Encoding*. Candidate solutions can be encoded in multiple ways, such as binary digits or, in the case of this study, a value like a string of characters.
2. *Fitness Evaluation*. At the start, each possible candidate is evaluated based on the function to optimize. For example, when seeking to optimize a machine learning algorithm, a performance metric like accuracy or F1 score can be used as the fitness function.
3. *Selection*. This operator selects the best candidates to provide offspring based on their fitness. There are many selection mechanisms, including rank based as well as *tournament selection*, the method used in this study. In tournament selection, groups of candidates are randomly created, and the candidate with the best fitness survives. In addition, *elitist* selection allows previous candidates to remain in the population.
4. *Crossover*. The process by which remaining candidates produce offspring by combining their sets of alleles into one new set. *Partially matched crossover* is the most common for sequence data (like text), as it preserves the order of observations.
5. *Mutation*. To avoid premature convergence, some alleles are randomly modified using a technique appropriate to the data representation. For example, a new value can be randomly sampled from existing possible values.

Genetic algorithms have been applied to numerous problems, including logistics, information security, image processing, agriculture, gaming, and wireless communications (Katoch *et al.* 2020). We use a genetic algorithm to select optimal examples for the GPT-3 Completion Endpoint.

3. Proposed augmentation methods

This study evaluates two methods for improving short text classification by augmenting training data using GPT-3. The first augments the GPT-3 Classification Endpoint, which classifies each short text input by searching for the most semantically similar training examples and comparing the probabilities of each class. In this method, additional training examples for the Classification Endpoint are generated using GPT-3 and added to the set of training examples in an attempt to improve performance. The second method augments the GPT-3 Completion Endpoint using a genetic algorithm to select the optimal examples to be included in the input context. These methods are described in detail as follows.

3.1 Classification Endpoint Augmentation

As mentioned in Section 2, the OpenAI GPT-3 Classification Endpoint performs classification by comparing the input text to a labeled training set. A semantic search first identifies a specific

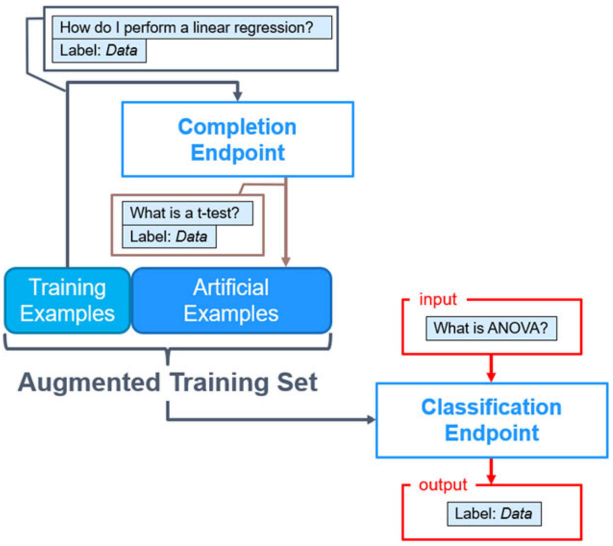


Figure 1. Our Classification Endpoint Augmentation. First, new artificial questions are created using the ability of the GPT-3 Completion Endpoint to generate text based on existing examples. Then, newly generated questions are used to train the Classification Endpoint to, given an input, produce a more accurate output.

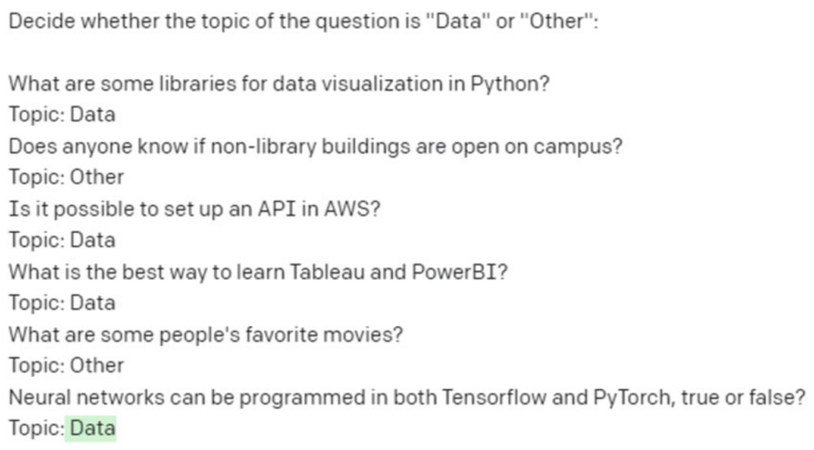
number of relevant training samples up to a user-specified hyperparameter *max_examples*. Then, these results are ranked based on their relevance. Finally, the input is classified based on the probabilities of the labels for the selected results. This process is provided by the OpenAI API (OpenAI 2021a 2022).

To augment the GPT-3 Classification Endpoint, this study generates additional training examples using GPT-3 itself—specifically, using the GPT-3 Completion Endpoint, which is designed to generate text given a prompt. To generate each additional training example for Classification Endpoint Augmentation, we first provide the GPT-3 Completion Endpoint with the prompt "Generate a similar question:" followed by three questions of the same class ("data" or "other") each preceded by the "Q:" token. These questions are randomly selected from the original training set. Finally, the output question is labeled with the same class as the input questions. The process is repeated to generate the desired number of questions with which to augment the Classification Endpoint.

This is similar to the method used by Kumar *et al.* (2020) who also generate synthetic examples using a large language model but employ filtering and fine-tuning to only include examples semantically similar to the existing training set. This method differs in that it allows all examples at first, regardless of quality, and instead selects the best examples for a given classification based on whichever ones yield the highest accuracy (instead of ones which are similar to the original training set). Figure 1 demonstrates this process graphically.

Once the desired number of example questions are generated, they are added to the training set, which is then uploaded to the OpenAI Classification Endpoint. With this training set, represented as a JSON file, users can call the GPT-3 API to classify any given text input. Of course, to enhance performance, users should compare the performances of models that use different numbers of additional training examples, as well as test different parameters for the Classification Endpoint such as *temperature* and *max_examples*—these are known as **hyperparameters**.

Different hyperparameters can result in slightly different performance. The *temperature* controls how deterministic the model's prediction is—a temperature of 0 ensures questions with clear labels are classified the same every time, while temperature greater than 0 allows some degree of



Decide whether the topic of the question is "Data" or "Other":

What are some libraries for data visualization in Python?
Topic: Data
Does anyone know if non-library buildings are open on campus?
Topic: Other
Is it possible to set up an API in AWS?
Topic: Data
What is the best way to learn Tableau and PowerBI?
Topic: Data
What are some people's favorite movies?
Topic: Other
Neural networks can be programmed in both Tensorflow and PyTorch, true or false?
Topic: Data

Figure 2. Example of an input to the GPT-3 Completion Endpoint interface. By organizing text in the *question-topic-question-topic* pattern, GPT-3 can be instructed to output labels classifying the topic of a question. The output of the model is highlighted.

guessing to better handle examples with less clear labels. As mentioned previously, *max_examples* determines the number of training examples to which the Classification Endpoint compares each question to be classified—a higher number provides more information to inform the question's label, but having too many slows prediction and could provide misleading, irrelevant information. There are many ways to optimize hyperparameters (Bischl *et al.* 2023); the methods used in this study are described in Section 4.

3.2 Completion Endpoint Augmentation

3.2.1 Classification using the Completion Endpoint

In the first augmentation method, we just described, we use the Completion Endpoint to generate new examples to attempt to augment the Classification Endpoint. Recall from Section 2, however, that the Completion Endpoint can also be directly applied to classification problems. To classify the topic of a question, we prompt the Completion Endpoint with the phrase "Decide whether the topic of the question is 'Data' or 'Other'," and append several training examples in a *question-topic-question-topic* pattern. An example of this input to the Completion Endpoint that results in classification is shown in Figure 2.

The examples provided, often referred to as "in-context examples," serve as a miniature training set for the Completion Endpoint. To classify a question, we simply append it to the end of the prompt and add a "Topic:" token with no label after it. When provided as input, this prompts GPT-3 to predict the next token as the topic of the previous question. The API even allows us to restrict the output of GPT-3 to specified tokens, so we can ensure that only the possible classes (in this case, "Data" or "Other") can be output as predictions. Hence, the Completion Endpoint can be used as a classifier after being provided with only a few training examples.

Like the Classification Endpoint, the Completion Endpoint can also be provided augmented training examples, but input prompts are limited to only 2,049 tokens. This means only a small number of training examples can be used. In addition, limited training examples are desirable because the more tokens included in the prompt, the more expensive the model is to call from the API. Therefore, rather than augmenting the Completion Endpoint by generating additional training examples, this study uses an optimization algorithm to select which subset of examples from a larger training set yields the best accuracy when provided as a prompt.

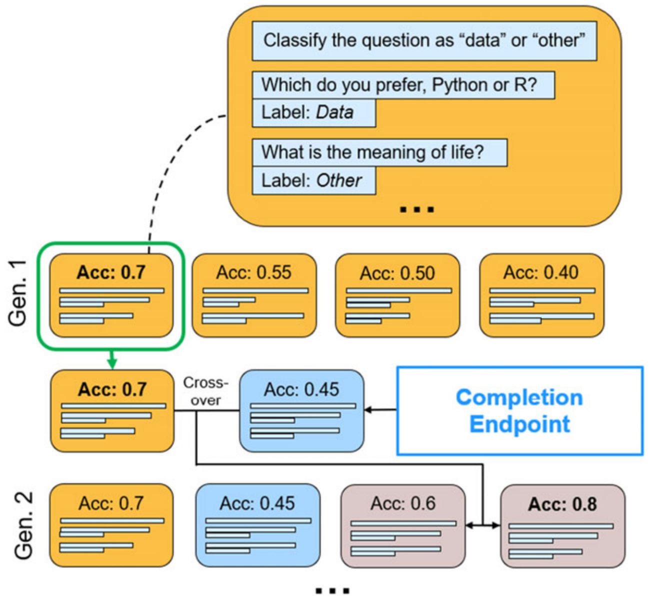


Figure 3. Graphical depiction of the genetic algorithm for selecting optimal augmented in-context examples for the GPT-3 Completion Endpoint. Each candidate consists of a set of alleles representing questions provided to the Completion Endpoint prompt. At each generation, the candidates with the best accuracy are selected to produce offspring with new candidates containing augmented examples generated by GPT-3.

3.2.2 Performing augmentation by optimizing training example selection

Few-shot learning with the Completion Endpoint is limited to only a small number of examples, so we need some way to select the optimal subset of examples to provide as a prompt. To optimize which augmented examples are chosen to include in the Completion Endpoint prompt, this work employs a genetic algorithm, as described in the Section 2 literature review. Recall that a genetic algorithm maintains a population of *candidates* with certain *alleles* (traits). Each candidate represents a possible prompt for the Completion Endpoint, consisting of a sequence of example-label pairs. An example of a single candidate was shown previously in Figure 2. Each allele within a candidate represents a single example-label pair, encoded as a string of text, in the sequence. For instance, the text snippet "Is it possible to set up an API in AWS? Topic: Data" in Figure 2 is an allele. The gene pool, then, is the set of all possible alleles, or all possible example-label pairs available for training.

The genetic algorithm process is depicted in Figure 3. As mentioned previously, a genetic algorithm applies a number of operators at each iteration. Operators apply some transformation to the existing population of candidates, and at each iteration, the fitness of the candidates improves. The specific operators we use in the genetic algorithm for optimizing Completion Endpoint training examples are described as follows (note that Step 1 is applied only in the first iteration) and are listed in Figure 4.

Step 1: Population Initialization. To begin, each candidate in the population is initialized as a set of random alleles sampled from the training set. We sample eight alleles, four from each class ("Data" or "Other"). We chose eight alleles to balance performance with API cost for few-shot learning, as the Completion Endpoint limits the number of tokens; other numbers could also be chosen. The alleles are sampled uniformly without replacement (only being replaced when the training set is empty), as the algorithm should ensure that no duplicate alleles are placed together in the same candidate (which would be inefficient). Then, each candidate is evaluated using the fitness function.

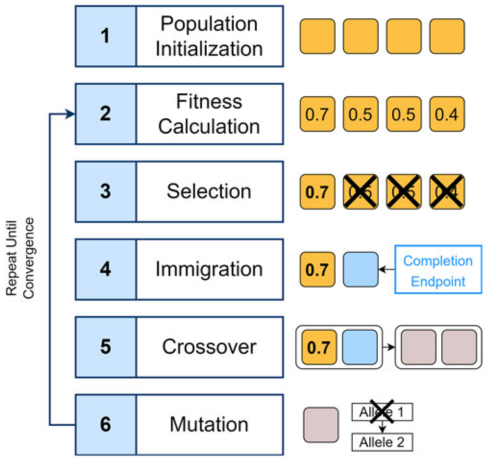


Figure 4. Steps in the genetic algorithm for Completion Endpoint augmentation.

Step 2: Fitness Calculation. The fitness function is defined as GPT-3's predictive performance on the validation set. In this study, we define fitness as the model's accuracy on the validation set since it is easily interpretable. Accuracy is also commonly used for NLP model performance evaluation (Brown *et al.* 2020). Other measures such as sensitivity or F1 score could also be chosen depending on the task. To evaluate the fitness for a given candidate, its alleles are provided as the prompt to the Completion Endpoint, which is used to predict the label of each observation in the validation set. The accuracy of these validation predictions is calculated as the fitness. Hence, we seek to maximize candidate accuracy.

Step 3: Selection. Once a fitness has been calculated for each candidate, selection is performed using *4-way tournament selection*. Candidates are allocated randomly into groups of four, and only the candidate with the best accuracy moves on to the next generation. This ensures that the algorithm uses elitist selection, allowing the best individual from the previous population to carry over to the next population. This prevents the best solutions from being lost. However, its random nature also promotes genetic diversity by allowing a small number of candidates that are not the best to possibly survive as well.

Step 4: Immigration. For such a small training set, it is necessary to introduce augmented examples—otherwise, the lack of genetic diversity will cause the algorithm to converge prematurely. This is performed using *immigration*, an operator which introduces candidates with entirely new sets of alleles into the gene pool (Yang 2004). The immigration operator in this algorithm employs the GPT-3 Completion Endpoint to generate new alleles in the same manner as in Classification Endpoint data augmentation: by prompting the Completion Endpoint with three random questions of the same class and asking it to generate a new question of the same class.

Step 5: Crossover. Genetic algorithms optimize by proposing new candidates likely to have high accuracy at each iteration. This is done by creating "offspring" that combine observations from different existing candidates. In this study, each candidate selected in Step 3 performs crossover with an immigrant to generate two offspring. We accomplish this using partially matched crossover which randomly selects one or more alleles to swap between the sets of alleles of the selected candidate and the immigrant (Katoch *et al.* 2020). These alleles are selected in such a manner as to prevent duplication of the same allele. The newly created sets become the alleles for each offspring. After this step, the number of candidates will have been multiplied by 4, reverting the effect of the selection process and returning the population to its original size (provided it was divisible by four; otherwise, the size will be approximately the original).

Step 6: Mutation. Finally, each allele in the offspring has a chance to be modified with some mutation probability. The mutation operator replaces the allele with a random allele sampled from the total gene pool—the set of all alleles in the population, including those newly generated in each immigrant population and excluding those already contained in the candidate. This promotes genetic diversity and prevents the optimization algorithm from being limited to the local optima of the best candidates found so far.

After this process, a new population is created which contains about the same number of candidates as the original. Then, steps 2 through 6 are repeated for the desired number of iterations (termed "generations") until the best candidate's desired accuracy is achieved, or some other stopping criteria such as a time or spending limit is met.

Through this procedure, the best candidates reproduce with new candidates, searching for new possible solutions that are similar to yet different from the best found so far. Using mutation and immigration to preserve genetic diversity prevents the population from becoming overrun with candidates that have the same alleles, thereby avoiding convergence to a mere local optimum. Instead, the algorithm better seeks a global optimum containing candidate sets of alleles that allow the Completion Endpoint to achieve the highest accuracy on the classification task.

4. Numerical evaluation

This section describes how the above methods were evaluated on an English language case study to compare their performance. Though the overall algorithms for data augmentation were described previously, here we detail which specific data and parameters were used for the models and the basic reasoning behind their selection in the evaluation process.

4.1 Data collection

To train the model, we collected a dataset of short text from the University of Massachusetts Dartmouth. This dataset consisted of questions asked in the Discord instant messaging app by undergraduate and graduate members of the University of Massachusetts Dartmouth Big Data Club. The Big Data Club Discord server is used by students at the university to specifically discuss data science topics as well as engage in casual conversation over text-based chat. As a result, we were able to collect both questions related to data science as well as questions related to other topics.

The dataset contained 72 questions that we labeled either "data" or "other" to indicate whether the topic was related to data science. Of these, 45 were collected directly from Discord messages with the members' permission. The remaining 27 were proposed by club members and edited by the research team with the goal of covering a broad range of data science topics, such as statistics, machine learning, databases, and cloud computing. They also included 8 counterintuitive examples that use data science terms in a non-data-related context, as well as 2 "junk" questions. For example, a counterintuitive question might be "How many neurons are contained within the human nervous system?"—although neural networks are a type of machine learning model, in this context the term "neuron" does not relate to data science. An example of a junk question would be a single-word question like "What?" or a string of random characters. Not only did these expand the distribution of possible topics in the training set, but they may also have served as "adversarial examples" that would have been exceptionally hard to classify or that would have represented boundary conditions in the data. In the end, about 14% of the data consisted of these human-proposed adversarial examples.

These questions were divided into three sets. The **training set** and **validation set** contained 26 questions each, with a random allocation of questions that occurred naturally and that were proposed by club members. The **test set** contained 20 questions and included only those originally asked in the Discord server. All sets were randomly selected to contain the same number of questions of the "data" and "other" classes to avoid any class imbalance issues.

The training set was used as input to each endpoint; it included the examples from which the algorithm actually learned. The validation set was used to evaluate performance and optimize parameters, such as the examples selected for the prompt in the genetic algorithm. The test set was also used to evaluate performance—it represented questions that the algorithm had never seen before, so performance on this set is the most important metric.

4.2 Implementation

Data processing, experimentation, and programming of the genetic algorithm were performed in Python 3.8.5. The Python API for OpenAI was used to call predictions from GPT-3. All simulations were run on a Dell XPS 15 9560 with an Intel (R) Core (TM) i7-7700HQ processor. API calls to GPT-3 used the default frozen model, with no fine-tuning.

Except for example generation, all GPT-3 API calls used the **ada** engine since it is currently the least expensive and yielded the fastest inference time—necessary characteristics for a real-time deployed machine learning application. New questions were generated using the **davinci** engine since it is optimized by performance. After data augmentation, once enough questions were generated, the model could be used continuously without needing to generate any more questions, negating any considerations of speed or cost. This is why we used davinci for generating new questions during data augmentation, but not for the actual classification task.

In evaluating the performance of the Completion Endpoint, we generated new question-label pairs for immigrant candidates on the fly. To evaluate the Classification Endpoint, to save costs, we sampled without replacement from a set of about 11,000 questions generated by the GPT-3 davinci engine during code development that were not used in the final Completion Endpoint tests.

4.3 Model parameters

4.3.1 Classification Endpoint Augmentation parameters

To evaluate augmentation on the Classification Endpoint Augmentation, we ran a battery of tests. In each test, n question-label pairs were first sampled from the set of 11,000 questions generated by the GPT-3 Completion Endpoint (using the procedure specified in Section 3.1) and added to the 26-question training set to create the augmented training set. Then, the augmented training set was formatted into a JSON file and uploaded to the API. Finally, using each augmented training set, we evaluated the performance of the model on the validation set and the test set.

Tests of the model's performance were run for $n = 0$ (for a baseline), 10, 100, 1000, and 10,000 augmented examples added. Each test was repeated five times for each hyperparameter setting. All augmented Classification Endpoint models were first evaluated on the validation set using different sets of hyperparameters to determine which hyperparameters were the best.

We optimized two hyperparameters, *temperature* and *max_examples*, using a grid search (Bischl *et al.* 2023). As mentioned previously, the *temperature* controls the determinism of the algorithm, with high values allowing GPT-3 to take more risks and be more creative, and low values limiting its answers to the most well-defined answer (OpenAI 2021a, b). For *temperature*, we tested values of 0, 0.1, and 0.5. The *max_examples* controls the number of training examples selected in the first step of the Classification Endpoint against which to compare the text being classified (OpenAI 2021a). For *max_examples*, we tested values of 5, 10, 15, 20, 25, and for >100 added examples, 100, in order to evaluate a large range of feasible selections.

After this, we selected the set of hyperparameters that achieved the best average accuracy out of the 5 evaluations on the validation set. A model using these hyperparameters was then evaluated

Table 1. Genetic algorithm parameters for Completion Endpoint in-context example selection optimization

Parameter	Value	Parameter	Value
Encoding	String	Population size	32
Selection method	Tournament	Tournament size	4
Crossover method	Partially matched	Crossover probability	1.0
Mutation method	Uniform	Mutation rate	0.1
Fitness function	Accuracy	Validation set size	26
GPT-3 engine	Ada	GPT-3 temperature	0
Header	Yes	Number of alleles	8

five times on the test set. This provides a description of the model's performance on unseen data, which more closely approximates how well it would perform if it were deployed in practice.

4.3.2 Completion Endpoint Augmentation parameters

To evaluate the augmentation method for the Completion Endpoint, we employed the genetic algorithm described in Section 3.2 to optimize the endpoint's accuracy on the validation set. The genetic algorithm was run for 40 generations. Eight alleles (possible training examples) were provided to each candidate. The algorithm was evaluated under the same setting three times. In the first, the candidates of the starting generation were made entirely of augmented examples generated by GPT-3 based on the training set. In the second and third, in order to increase convergence speed, the starting generation was restricted to only examples from the original training set, with no augmented examples at the beginning. We chose these quantities to balance assurance of reproducibility with available funds, which limited the total number of generations that could be run.

Table 1 displays the parameters used in the genetic algorithm. The fitness function calculated the accuracy of the algorithm on the validation set. Temperature was set to 0 to ensure classifications are deterministic. Partially matched crossover ensured that no duplicate examples were placed into the context (Katoch *et al.* 2020).

The uniform mutation rate of 0.1, although high, was chosen to ensure that most offspring would obtain at least one mutation since there are only eight alleles to be mutated. Many mutations are desirable because it permits random search in the algorithm. Combined with a population size of 32 with a tournament size of 4 and certain crossover, this ensures rapid replacement of the existing population. We believe rapid replacement is necessary because, while the original training set is small, there are nearly infinite possible examples to generate from them—hence, spending some time randomly searching the space of training examples rapidly will result in higher accuracy. It also prevents convergence to a local minimum.

Larger populations, higher mutation rates, and larger tournament sizes bias the algorithm toward random search, meaning that the algorithm can find new combinations that are closer to the global minimum more effectively. However, they also modify the convergence rate. Changing the hyperparameters could change how quickly the algorithm converges, prevent it from reaching the same results that we did within 40 generations.

Regarding the population dynamics, in each of the three tests, a population of 32 candidates with 8 alleles (question-label pairs) each was randomly initialized by sampling sets of alleles from the 26 training questions, with no duplicate questions permitted. These numbers were chosen to ensure that different combinations of the sample examples could be represented in the starting population. This also ensured that when each tournament winner produced two offspring with a new immigrant, the original population size would be recovered. At each generation, after 4-way tournament selection culled all but 8 candidates, 8 new immigrants were introduced, and crossover yielded 16 new offspring; as a result, we attain the same population size (32) as that prior to selection.

At the end of the 40 generations, we selected the best candidate in terms of fitness (accuracy) that occurred. We then took the examples from the best candidate as the Completion Endpoint model. Finally, the accuracy of this model was evaluated on the test set. This yielded the algorithm's performance on examples it had never seen before, to ensure that it continued to perform well in practice. The performance of our augmentation methods on the validation and test sets is described in the next section.

4.4 A note on baselines

To our knowledge, this is the first study using an optimization algorithm to augment training data based on newly generated examples from a large language model. While there are no baseline methods that solve exactly the same problem against which to compare our own, we can define a baseline for each endpoint based on a slight modification of existing techniques in the literature for ease of implementation.

Classification Endpoint Baseline: A method nearly identical to KATE (Liu *et al.* 2022) is implemented in the GPT-3 Classification Endpoint to select examples. Therefore, we take the mean accuracy of the hyperparameter-optimized Classification Endpoint with 0 newly generated examples added as our baseline. We expect this would mimic the performance of KATE while adapting it to our problem.

Completion Endpoint Baseline: Kumar *et al.* (2020) and Anaby-Tavor *et al.* (2020) use LAMBADA to expand the size of the training data directly, but our goal is to avoid doing this, and to instead use optimization to select the best generated examples. Since both the genetic algorithm's first iteration and LAMBADA both similarly generate new examples and filter for quality, we take the best solution produced by the first generation, measured by mean accuracy, as our baseline for the Completion Endpoint.

5. Results

5.1 Classification Endpoint Augmentation results

Data augmentation for the Classification Endpoint successfully resulted in increased model accuracy on text classification for the problem of classifying whether a question is related to data science in the English language. Table 2 reports the mean μ and standard error of the endpoint accuracy for both the validation set and the test set, with the model using optimized hyperparameters. A permutation test for difference in means is used to calculate p-values comparing the mean of the baseline (0 examples added) to each scenario with n examples added, to examine whether each difference in accuracy is statistically significant. The time taken to train each model was fairly consistent across simulations—it took about 18–20 minutes on average to optimize GPT-3's hyperparameters, no matter the amount of additional examples added. This time does not count generating the examples.

As more generated examples were added to the training set, the Classification Endpoint accuracy tended to increase. Without augmentation, the Classification Endpoint with just the 26-question training set performed comparably to random guessing, only classifying about 49% of questions correctly on average for the validation set and 58% on average for the test set. However, on the validation set, accuracy continually increased as more examples were added, reaching about 73% accuracy after adding 10,000 new examples.

Table 2. GPT-3 Classification Endpoint performance on data science question topic classification, additional examples generated using GPT-3 Davinci Completion. *p*-values test for significance from results with 0 additional examples using a permutation test for difference in means

n added	Validation accuracy			Test accuracy			Time (min)	
Examples	μ	SE	*p*	μ	SE	*p*	μ	SE
0	0.49	(±0.028)	–	0.58	(±0.023)	–	20.4	6.6
10	0.52	(±0.018)	0.21	0.56	(±0.048)	0.690	18.8	3.1
100	0.62	(±0.022)	0.012	0.62	(±0.041)	0.349	16.8	5.4
1000	0.67	(±0.008)	0.004	0.76	(±0.017)	0.004	19.3	5.7
10,000	0.73	(±0.000)	0.004	0.73	(±0.030)	0.008	18.6	8.5

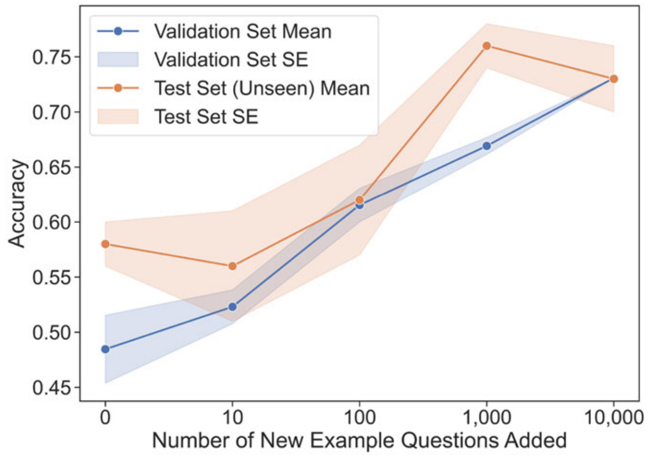


Figure 5. GPT-3 Classification Endpoint mean performance with standard errors on data science question topic classification. Training data are augmented by adding different quantities of new examples generated with GPT-3 Davinci Completion.

While, for the validation set, accuracy was positively related to the number of questions generated, the same was not true for the test set. Figure 5 plots the relationship between accuracy and number of example questions added across both validation and test sets, with the shaded regions representing the standard error. Note that the x-axis is a log scale. On the test set, accuracy scarcely increased at all until the number of questions added reached about 1000, at which point it increased to 76%. This represented peak accuracy; augmented training sets with 10,000 new questions averaged only 73% accuracy, a slight drop.

Overall, data augmentation yielded statistically significant increases in accuracy, based on permutation tests between accuracies for each set of *n* additional artificial examples and the baseline. On the validation set, only 100 examples were necessary to observe statistically significant increases in accuracy at $\alpha = 0.05$. However, adding 100 examples did not yield significant increases in accuracy on the test set, as the baseline accuracy was much higher. With the more strict $\alpha = 0.01$ requirement for significance, the increase in accuracy became statistically significant after adding 1000 examples or more for both the validation and test set.

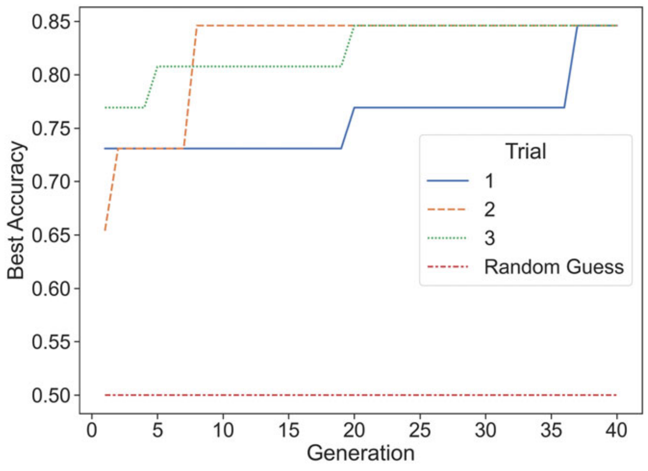


Figure 6. Genetic algorithm performance for selecting best in-context examples for the GPT-3 Completion Endpoint. Results from each individual trial are compared to the baseline, which represents the expected performance of random guessing, in terms of classification accuracy on the 26-question validation set.

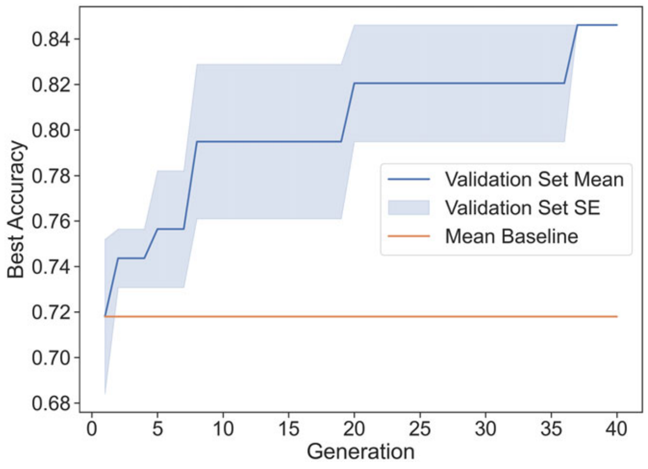


Figure 7. Averaged genetic algorithm performance for selecting best in-context examples for the GPT-3 Completion Endpoint with standard error across three trials. Performance is measured in terms of classification accuracy on the 26-question validation set.

5.2 Completion Endpoint Augmentation results

To evaluate the performance of the augmentation method for the Completion Endpoint, we first examine the changes in validation set accuracy across each generation of the genetic algorithm. Three trials of the genetic algorithm experiment were completed. Figure 6 displays the validation accuracy across generations for each individual trial, while Figure 7 shows the averaged results across trials, with the shaded regions representing standard error. We take the first generation's performance as a baseline, approximating what we might expect from another data-generation-based augmentation algorithm (Kumar *et al.* 2020). Trials 1–3 took 105, 119, and 112 minutes to run, respectively ($\mu = 112 \pm 3.3$), counting example generation.

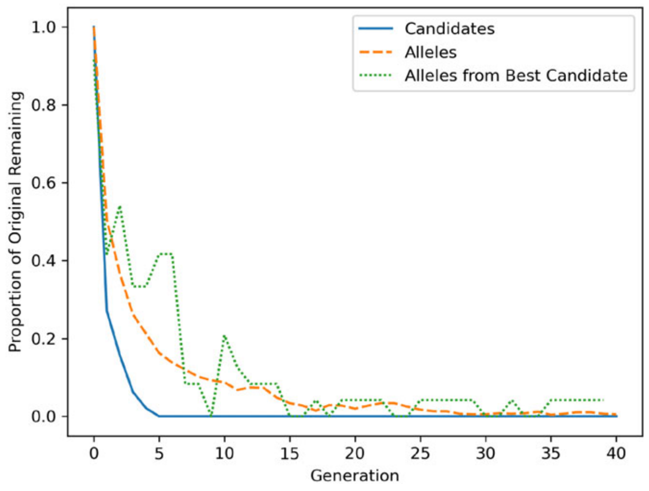


Figure 8. Average proportion of original population of alleles, candidates, and alleles from the best candidate that remain in the population at each subsequent generation across three trials. This plot shows the average replacement rate of each over time.

Figures 6 and 7 demonstrate that, during the genetic algorithm, the *validation* accuracy of the best candidate continually increased—rapidly at the start, and then more slowly as the algorithm converges. Moreover, simply selecting the best candidate from 32 random subsets, which yielded the validation accuracy at generation 0, resulted in an average validation accuracy of about 72%. Hence, before generating any new candidates, the Completion Endpoint achieved a validation accuracy comparable to the Classification Endpoint augmented with 10,000 new training examples, which obtained about 73% accuracy.

Using data augmentation on the Completion Endpoint by selecting the optimal set of examples yielded a validation accuracy of about 85%—much better than augmentation on the Classification Endpoint. It is important to note that large spikes in accuracy often occurred in a single generation in the genetic algorithm. These spikes represented when a new best combination of training questions was found, which could have corrected several errors from the previous set at once. In addition, although generations 10–35 featured large standard errors, by the end of each of the three trials, all simulations reached the same final best validation accuracy of about 85%.

Figure 8 shows the average replacement rate of the population in the genetic algorithm. None of the starting candidates were the best at classification, and they were quickly replaced by offspring. Alleles from the original population tended to be replaced by newly generated examples over time, though a small number lingered in the population across generations. Alleles from the best candidate did tend to last slightly longer than others.

Results for the Completion Endpoint worsen on the unseen test set. Table 3 compares the best test accuracy as well as validation accuracy for both the Completion Endpoint and the Classification Endpoint. Unlike augmenting the Classification Endpoint, augmenting the Completion Endpoint with a set of optimal examples failed to achieve a testing accuracy comparable to the validation accuracy. We see that the test accuracy, 67%, was much lower than the validation accuracy of 85%.

This indicates that the in-context example set solutions produced by the genetic algorithm did *not* generalize well to unseen questions. As a result, when augmented with additional in-context examples, the Completion Endpoint was far less consistent in performance compared to the Classification Endpoint. Furthermore, since unseen questions more closely approximated those the algorithm would have been asked to classify in a practical environment, it is arguable

Table 3. Comparison of augmented GPT-3 Completion Endpoint performance on data science question classification, in-context examples selected using genetic algorithm ($n_{\text{trials}} = 3$), to augmented Classification Endpoint

	Completion Endpoint	Classification Endpoint
Best validation accuracy	0.85	0.73
Best test accuracy	0.67 (±0.024)	0.76 (±0.017)
Mean training time (min)	112 (±3.3)	19 (±0.5)
Mean proportion of differing alleles	0.94 (±0.009)	–

that the augmented Classification Endpoint performed better, with a 76% accuracy on the test set compared to the Completion Endpoint's 67% accuracy.

Granted, if the algorithm were run for significantly longer than 40 generations, it is possible that a better solution could have been found. The *Mean Proportion of Differing Alleles* displayed at the bottom of Table 3 measures the relative genetic diversity of the candidate population in generation 40. On average, 94% of questions in any given candidate were not shared by any other given candidate, indicating that the sets of alleles among different candidates in the final population at generation 40 contained mostly different questions. Hence, since the technique maintained genetic diversity across generations, we know it did not converge prematurely. Therefore, it could have been possible for a search to find a better solution. However, given the performance degradation between validation and test sets with the Completion Endpoint, this would probably be an inefficient use of resources.

5.3 Reviewer agreement

While accuracies less than 80% may seem less than desirable, in reality, categories are subjective, meaning that performance will never reach very high accuracies. For example, a question that one person believed to be data science may be seen as unrelated by another person, meaning that even humans cannot achieve perfect performance. To measure the subjectivity of answers, after completing the training process, we solicited additional labels for the training, validation, and test sets from 3 participants who previously contributed questions.

The results, stratified by dataset, are included in Table 4. Only 69% of questions in total had complete label agreement across reviewers. The greatest agreement occurred for the training set, while reviewers had the least agreement on the validation set. **Most importantly, the reviewers achieved only about 85% accuracy on the test set and 76% on validation.** This puts the performance of data augmentation with GPT-3 into context; our two methods achieved comparable or better accuracy on the validation set, and only 10% less accuracy on the test set.

How did the GPT-3's struggles compare to those of the reviewers? The three questions that were always misclassified by every GPT-3 model are included below. They are labeled with their "true" category in square brackets, followed by the fraction of reviewers which agreed with the original label.

1. *"Can someone help me understand how to SSH into the computing cluster on Friday?"* [Data] [1/3]
2. *"Here's a link on how to make custom Jupyter notebook themes. Big Data Club-themed notebooks, anyone?"* [Data] [0/3]
3. *"Are you coming to Big Data Club tomorrow?"* [Other] [3/3]

Table 4. Proportion of questions for which a given fraction of post hoc reviewers agreed with the original label. Human accuracy is reported as the fraction of post hoc labels that matched the original label

	Training	Validation	Test
3/3 reviewers agreed	0.77	0.58	0.75
$\geq$ 2/3 reviewers agreed	0.92	0.77	0.85
$\geq$ 1/3 reviewers agreed	1.00	0.88	0.95
Human accuracy	0.90	0.74	0.85

Compared to questions like “What are some libraries for data visualization in python?” (3/3 agreement among reviewers), these questions are less directly related to data science. In question 1, although data scientists use computing clusters, such language is not necessarily specific to data science as a field, as two of the post hoc reviewers evidently believed. Even though question 2 was labeled “Data” since Jupyter notebooks are frequently used in the data science domain, the post hoc reviewers believed that such a question regarding the appearance of a particular software was not relevant to the actual field of data science. Finally, even though every reviewer agreed that question 3 should have been labeled “Other,” since the club in question is dedicated to data science, GPT-3 always classified it as “Data.” Hence, while GPT-3 and the reviewers were both frequently in disagreement on the labels of some questions, not every question that GPT-3 classified incorrectly received mixed labels.

6. Discussion

Meaning, as expressed by human language, is highly subjective. To one person, a question might be very relevant to a certain topic, while to another, it could be completely unrelated. Hence, labeling text with a category or topic depends on the complex and minute contextual associations between the words and phrases it contains. This study has demonstrated that, while GPT-3 cannot entirely overcome the limitations of this subjectivity, using data augmentation, it can capture strong enough contextual relationships between words to classify short text topics in a practical real-world deployment setting with limited data.

6.1 The efficacy of embedding

This study compared data augmentation techniques for the GPT-3 Classification and Completion Endpoints, using GPT-3 to generate its own original training data examples based on observations from an existing training dataset. Augmenting the Classification Endpoint improved performance significantly, increasing validation accuracy from 49% to about 73% and increasing test set accuracy from 58% to 76%. In comparison, humans agreed with the original labels about 76% of the time on the validation set and about 85% of the time on the test set. Augmenting the Completion Endpoint yielded even better validation accuracy of about 85%, but this performance was inconsistent and likely overfitted; accuracy dropped to 67% when evaluated on the test set, which represented unseen questions.

For text classification with GPT-3, augmentation using embedding-based models like the Classification Endpoint appears to be preferable to using genetic algorithms. We hypothesize two possible reasons for this.

First, despite that genetic algorithms excel on combinatorial problems with many local minima, it appears that even the small text prompt of the Completion Endpoint is too high-dimensional to be solved efficiently—especially starting with such a small training set. This is why genetic algorithms are generally not popular in NLP. Embeddings were developed to handle this problem (Minaee *et al.* 2022), and it is logical for embedding-based approaches to be less overfitted.

Second, due to cost, the Classification Endpoint method was able to select different examples for every classification, while the Completion Endpoint was restricted to using the same prompt every time. So, while employing an embedding-based approach to optimize the Completion Endpoint prompt might help avoid overfitting, it also might not improve its performance overall. The success of the Classification Endpoint may have been more due to its ability to draw information from many more than the 8 examples available to the Completion Endpoint, rather than its use of embedding itself.

6.2 Remaining questions

Why was there a loss in accuracy between validation and test sets for the Completion Endpoint but not for the Classification Endpoint? First, the optimization performed in the Classification Endpoint only modified the hyperparameters, *temperature* and *max_examples*, while for the Completion Endpoint, the genetic algorithm optimized the in-context examples themselves. If the accuracy of GPT-3 depends more on the in-context examples in the training set than on the chosen hyperparameters—which it should so that it can adapt to different problem domains—then this may have caused the genetic algorithm to overfit to the validation data. Overfitting was not avoided even in Trial 1, where the training set was augmented with generated questions in the first generation. Since augmented examples that improve on existing ones are rare, we do not hypothesize providing augmented examples to the initial population in the genetic algorithm to be effective. Furthermore, it is possible that the smaller training set in the Completion Endpoint may simply have been a poorer representation of the broad distribution of possible questions in general, causing worse performance.

More broadly, why did our technique for data augmentation improve performance? We hypothesize that it worked because GPT-3 is better at generating text than classifying (Brown *et al.* 2020). This means that it can produce new examples that closely match the given category better than it can match a category to an example. Hence, it follows that relying more heavily on the algorithm's generative capabilities can improve its ability to identify topics correctly. In general, having more samples to describe the true population space tends to result in improved performance of machine learning models (Hestness *et al.* 2017). Data augmentation increases training set size; providing more samples allowed GPT-3 to learn from question-label pairs that more closely approximated the new questions that it was asked to classify.

6.3 Limitations and future work

6.3.1 Language

The greatest limitation of this study is that its results are confined to the English language. Because the data were sampled from a solely English-speaking population, the proposed method's performance could only be evaluated on questions in English. Further work must expand the performance of data augmentation for GPT-3 to other languages, especially morphologically rich languages that may pose unique challenges to model (Gerz *et al.* 2018). While the current work demonstrates the applicability of this method to applications in English, additional research is necessary to confirm its genericity and applicability to other languages.

6.3.2 Sample size

Of course, this study is limited by the sample size of the validation and test sets. Testing data augmentation on a practical scenario where only a small domain-specific dataset is available meant the test set could only be limited to 20 questions, which obviously reduces the robustness of the results. It also limited the study to one particular classification problem.

Although we specifically collected and included a subset of 27 questions meant to cover a wide variety of data science topics and represent boundary cases, the set of all possible questions is broad and the impact of including these is therefore unclear. Random sampling of questions-label pairs into training and testing sets during evaluation and random selection for topic generation resulted in performance differences, as evidenced by the standard error in the results. These variations were not extremely large, though—standard errors only reached about 1–5% differences in accuracy. Still, additional research could address whether a starting training set of more diverse questions would also improve data augmentation efforts.

6.3.3 Task diversity

Future research in this area should also assess the performance of data augmentation on different types of classification problems, such as those in other domains or with several possible classes. One particularly pertinent problem would be data augmentation on data containing a class imbalance (Onan 2019a), which may require better methods for selecting effective training examples. This is because if a specific class is underrepresented in the prompt or training data compared to others, the model may fail to classify any text with that given label. It would also be useful to assess the model's performance in even more limited-data scenarios, such as having only 4–5 labeled examples available, to fully assess how much of the performance is attributable to having a small validation set available for hyperparameter optimization.

The Completion Endpoint-based data augmentation approach is not limited to just classification problems. As discussed by Brown *et al.* (2020), few-shot learning with GPT-3 can be applied to other language benchmark tasks, including SuperGLUE (Wang *et al.* 2019), as well as translation, common sense reasoning, reading comprehension, Winograd-style tasks, and others. Hence, genetic algorithms can also be used to engineer optimal prompts as well, as long as users adapt the types of examples and fitness functions used in the genetic algorithm to match the type of task being performed.

In fact, some tasks may require more complicated training procedures. Multihop question-answering requires large language models to answer several related questions in sequence (Jiang *et al.* 2022). Future research could attempt data augmentation on more complex problems like multihop question answering by providing each training example as a set of multiple questions in sequence (or, potentially, even a tree of questions) and optimizing for multianswer evaluation metrics such as exact match as the fitness function.

6.3.4 Subjectivity and quality control

Since all three post hoc reviewers agreed on the original question label only about 71% of the time across all of the data, it is necessary to mitigate the effect of incorrect labels on prediction. One way to do this is using data quality control. For example, a software application that deployed data augmentation on GPT-3 for text classification could periodically poll users, asking the topic of a given question. This would allow the application to both determine which questions have more subjective topics and ensure that existing training examples are classified correctly.

Future work or implementations could also include additional labels such as "Unsure" or even "Sure Data" versus "Possibly Data," which might be applied if multiple annotators express disagreement on the true topic of the question. For instance, if a poll indicates great disagreement from human annotators on a particular category, it could be assigned an "Unsure" label which could be processed differently from examples that have more certain labels. This would allow GPT-3 to differentially classify examples whose labels are less clear.

6.3.5 Comparison to other data augmentation techniques

Finally, even though example optimization did not produce consistent results on the test set, future work may consider alternative optimization techniques for selecting the best example-label pairs to provide to GPT-3, such as reinforcement learning (Arulkumaran *et al.* 2017). For example, previous research has applied reinforcement learning to text generation for data augmentation (Liu *et al.* 2020), and similar approaches could be applied to transfer learning models like GPT-3 in examples like the one explored here. Similarly, other optimization methods could even be used to select how the Completion Endpoint prompt itself is formatted. We chose a *question-label* structure with a fixed prompt based on examples provided the documentation OpenAI (2021b), but an algorithm could also identify whether using a *label-question* structure or different sets of instructions could yield better performance.

In the future, it would be wise to compare the proposed data augmentation technique for GPT-3 prompt optimization to other techniques that ensure higher quality examples. As surveyed by Guo and Yu (2022), importance sampling and pseudo-labeling may be combined to produce better prompts and training sets. If more data were collected, model optimization techniques—including continual learning, adversarial learning, and metric learning—could also be used to construct better input prompts and training sets. Other text-specific data augmentation methods like LAMBADA (Anaby-Tavor *et al.* 2020) feature methods to mitigate potential noise and error from generated training examples which could be incorporated into our optimization-based approach. Furthermore, personalization—methods that help the model better understand personal habits of language usage within a domain—could help optimize prompts for a specific setting. For instance, personalization could help the model avoid classifying a term like "Big Data Club" as related to data science when used in the organization's Discord server, even though though such a classification might be correct in a different context.

7. Conclusion

This study finds that through the process of optimization-based data augmentation, the generative capabilities of GPT-3 allow small ($n < 30$) short text data sets to be used for developing effective text classification models. Expanding the size of the training set in the GPT-3 Classification Endpoint by generating new examples using GPT-3 itself was found to increase both validation and testing accuracy. Adding 1000 artificial examples to a Classification Endpoint model increased classification accuracy from the baseline of 0.58 to about 0.76—a 31% increase, and only about 10% less than the estimated human accuracy of 0.85. We also explored the use of genetic algorithms for optimizing in-context examples for few-shot learning with the Completion Endpoint, but the model performed inconsistently, achieving a validation accuracy of about 85% but a test accuracy (on unseen questions) of only 67%. As such, optimization-based data augmentation is more effective when using an embedding-based model that can draw from a large augmented training set, instead of being restricted to a small prompt.

Augmenting training data for short text classification using the generative capabilities of GPT-3 allows NLP models to be constructed even for solving problems with limited available data. This overcomes a common problem with transfer learning wherein models for domain-specific practical problems often require either high-quality observations or training sets too large to be collected or annotated by a single individual. If these models are coupled with methods to check, validate, or even better optimize artificial examples generated automatically, the few-shot capabilities of transfer learning models such as GPT-3 can be fully realized. Using data augmentation with GPT-3 would allow businesses and organizations to more easily construct bespoke NLP models for problems unique to their own domains.

Acknowledgments. First, we must thank the Program in Data Science at the University of Massachusetts Dartmouth for financially supporting this project. In addition, we would also like to thank members of the University of Massachusetts

Dartmouth Big Data Club for contributing questions to our training, validation, and test data. Finally, we especially thank Benjamin Pfeffer, McCord Murray, and John Willy for generously volunteering to perform post hoc annotations of the questions used in this study.

Competing interests. Salvador Balkus previously served as a volunteer on the UMass Dartmouth Big Data Club executive board.

References

Adhikari A., Ram A., Tang R. and Lin J. (2019). Docbert: Bert for document classification. arXiv:1904.08398.

Anaby-Tavor A., Carmeli B., Goldbraich E., Kantor A., Kour G., Shlomov S., Tepper N. and Zwerdling N. (2020). Do not have enough data? Deep learning to the rescue! In *AAAI Conference on Artificial Intelligence*.

Arulkumaran K., Deisenroth M. P., Brundage M. and Bharath A. A. (2017). Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine* **34**(6), 26–38.

Bajaj P., Xiong C., Ke G., Liu X., He D., Tiwary S., Liu T.-Y., Bennett P., Song X. and Gao J. (2022). Metro: Efficient denoising pretraining of large scale autoencoding language models with model generated signals. arXiv:2204.06644.

Bischl B., Binder M., Lang M., Pielok T., Richter J., Coors S., Thomas J., Ullmann T., Becker M., Boulesteix A.-L., Deng D. and Lindauer M. (2023). Hyperparameter optimization: Foundations, algorithms, best practices, and open challenges. *WIREs Data Mining and Knowledge Discovery* **13**(2), 7–16.

Brown T. B., Mann B., Ryder N., Subbiah M., Kaplan J., Dhariwal P., Neelakantan A., Shyam P., Sastry G., Askell A., Agarwal S., Herbert-Voss A., Krueger G., Henighan T. J., Child R., Ramesh A., Ziegler D. M., 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. and Amodei D. (2020). Language models are few-shot learners. arXiv:2005.14165.

Chen W., Ramos K., Mullaguri K. N. and Wu A. S. (2021). Genetic algorithms for extractive summarization. arXiv:2105.02365.

Dai A. M. and Le Q. V. (2015). Semi-supervised sequence learning. arXiv:1511.01432.

Dale R. (2020). GPT-3: What's it good for? *Natural Language Engineering* **27**(1), 113–118.

Deng X., Li Y., Weng J. and Zhang J. (2018). Feature selection for text classification: A review. *Multimedia Tools and Applications* **78**(3), 3797–3816.

Devlin J., Chang M.-W., Lee K. and Toutanova K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv:1810.04805.

Du J., Grave E., Gunel B., Chaudhary V., Çelebi O., Auli M., Stoyanov V. and Conneau A. (2020). Self-training improves pre-training for natural language understanding. In *North American Chapter of the Association for Computational Linguistics*.

Feng S., Gangal V., Wei J., Chandar S., Vosoughi S., Mitamura T. and Hovy E. (2021). A survey of data augmentation approaches for NLP. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*. Association for Computational Linguistics.

Gerz D., Vulić I., Ponti E., Naradowsky J., Reichart R. and Korhonen A. (2018). Language modeling for morphologically rich languages: Character-aware modeling for word-level prediction. *Transactions of the Association for Computational Linguistics* **6**, 451–465.

GPT-3. (2020). A robot wrote this entire article. Are you scared yet, human? *The Guardian*. Available at https://www.theguardian.com/commentisfree/2020/sep/08/robot-wrote-this-article-gpt-3.

Guo X. and Yu H. (2022). On the domain adaptation and generalization of pretrained language models: A survey. arXiv:2211.03154.

Hestness J., Narang S., Ardalani N., Diamos G., Jun H., Kianinejad H., Patwary M. M. A., Yang Y. and Zhou Y. (2017). Deep learning scaling is predictable, empirically. arXiv:1712.00409.

Jiang Z., Araki J., Ding H. and Neubig G. (2022). Understanding and improving zero-shot multi-hop reasoning in generative question answering. In *International Conference on Computational Linguistics*.

Katoch S., Chauhan S. S. and Kumar V. (2020). A review on genetic algorithm: past, present, and future. *Multimedia Tools and Applications* **80**(5), 8091–8126.

Kobayashi S. (2018). Contextual augmentation: Data augmentation by words with paradigmatic relations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*. New Orleans, LA: Association for Computational Linguistics, pp. 452–457.

Kowsari K., Meimandi K.J., Heidarysafa M., Mendu S., Barnes L. and Brown D. (2019). Text classification algorithms: A survey. *Information* **10**(4), 150.

Kumar V., Choudhary A. and Cho E. (2020). Data augmentation using pre-trained transformer models. In *Proceedings of the 2nd Workshop on Life-long Learning for Spoken Language Systems*. Suzhou, China: Association for Computational Linguistics, pp. 18–26.

Li Q., Peng H., Li J., Xia C., Yang R., Sun L., Yu P. S. and He L. (2020). A survey on text classification: From shallow to deep learning. *ACM Transactions on Intelligent Systems and Technology* **13**(2), 1–41.

Liu J., Shen D., Zhang Y., Dolan B., Carin L. and Chen W. (2022). What makes good in-context examples for GPT-3?. In *Proceedings of Deep Learning Inside Out (DeeLIO 2022): The 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*. Dublin and Online: Association for Computational Linguistics, pp. 100–114.

Liu R., Xu G., Jia C., Ma W., Wang L. and Vosoughi S. (2020). Data boost: Text data augmentation through reinforcement learning guided conditional generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP).* Online: Association for Computational Linguistics, pp. 9031–9041.

Liu Y., Ott M., Goyal N., Du J., Joshi M., Chen D., Levy O., Lewis M., Zettlemoyer L. and Stoyanov V. (2019). Roberta: A robustly optimized bert pretraining approach. arXiv:1907.11692.

Minaee S., Kalchbrenner N., Cambria E., Nikzad N., Chenaghlu M. and Gao J. (2022). Deep learning–based text classification. *ACM Computing Surveys* **54**(3), 1–40.

Onan A. (2018). Biomedical text categorization based on ensemble pruning and optimized topic modelling. *Computational and Mathematical Methods in Medicine* **2018**, 1–22.

Onan A. (2019a). Consensus clustering-based undersampling approach to imbalanced learning. *Scientific Programming* **2019**, 1–14.

Onan A. (2019b). Two-stage topic extraction model for bibliometric data analysis based on word embeddings and clustering. *IEEE Access* **7**, 145614–145633.

Onan A. and Korukoğlu S. (2016). A feature selection model based on genetic rank aggregation for text sentiment classification. *Journal of Information Science* **43**(1), 25–38.

Onan A., Korukoğlu S. and Bulut H. (2016). Ensemble of keyword extraction methods and classifiers in text classification. *Expert Systems with Applications* **57**, 232–247.

Onan A., Korukoğlu S. and Bulut H. (2017). A hybrid ensemble pruning approach based on consensus clustering and multi-objective evolutionary algorithm for sentiment classification. *Information Processing and Management* **53**(4), 814–833.

OpenAI (2021a). *Classification*. OpenAI Documentation. Available at https://beta.openai.com/docs/guides/classifications

OpenAI (2021b). *Completion*. OpenAI Documentation. Available at https://beta.openai.com/docs/guides/completion

OpenAI (2022). *Embeddings*. OpenAI Documentation. Available at https://platform.openai.com/docs/guides/embeddings/what-are-embeddings

Pan S. J. and Yang Q. (2010). A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering* **22**(10), 1345–1359.

Pilipiszyn A. (2021). *GPT-3 Powers the Next Generation*. OpenAI. Available at https://openai.com/blog/gpt-3-apps/

Qu Y., Shen D., Shen Y., Sajeev S., Han J. and Chen W. (2020). CoDA: Contrast-enhanced and diversity-promoting data augmentation for natural language understanding. arXiv:2010.08670.

Quteineh H., Samothrakis S. and Sutcliffe R. (2020). Textual data augmentation for efficient active learning on tiny datasets. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online. Association for Computational Linguistics, pp. 7400–7410.

Radford A., Narasimhan K., Salimans T. and Sutskever I. (2018). Improving language understanding by generative pre-training. *OpenAI*. Available at https://openai.com/research/language-unsupervised.

Raffel C., Shazeer N., Roberts A., Lee K., Narang S., Matena M., Zhou Y., Li W. and Liu P. J. (2022). Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research* **21**(1), 5–10.

Sennrich R., Haddow B. and Birch A. (2016). Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin: Association for Computational Linguistics, pp. 86–96.

Shorten C. and Khoshgoftaar T. M. (2019). A survey on image data augmentation for deep learning. *Journal of Big Data* **6**(1), 7–17.

Song G., Ye Y., Du X., Huang X. and Bie S. (2014). Short text classification: A survey. *Journal of Multimedia* **9**(5), 1–2.

Srinivas M. and Patnaik L. (1994). Genetic algorithms: A survey. *Computer* **27**(6), 17–26.

Sun Y., Wang S., Feng S., Ding S., Pang C., Shang J., Liu J., Chen X., Zhao Y., Lu Y., Liu W., Wu Z., Gong W., Liang J., Shang Z., Sun P., Liu W., Ouyang X., Yu D., Tian H., Wu H. and Wang H. (2021). Ernie 3.0: Large-scale knowledge enhanced pre-training for language understanding and generation. arXiv:2107.02137.

Thiergart J., Huber S. and Ubellacker T. (2021). Understanding emails and drafting responses – an approach using GPT-3. arXiv:2102.03062.

Vaswani A., Shazeer N. M., Parmar N., Uszkoreit J., Jones L., Gomez A. N., Kaiser L. and Polosukhin I. (2017). Attention is all you need. arXiv:1706.03762.

Wang A., Pruksachatkun Y., Nangia N., Singh A., Michael J., Hill F., Levy O. and Bowman S. R. (2019). SuperGLUE: A stickier benchmark for general-purpose language understanding systems. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems.* Curran Associates Inc, pp. 3266–3280.

Yang W. X. (2004). An improved genetic algorithm adopting immigration operator. *Intelligent Data Analysis* **8**(4), 385–401.

Yoo K. M., Park D., Kang J., Lee S.-W. and Park W. (2021). GPT3Mix: Leveraging large-scale language models for text augmentation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*. Association for Computational Linguistics.

Zhang H., Cisse M., Dauphin Y. N. and Lopez-Paz D. (2018). mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*.

Zhao T., Wallace E., Feng S., Klein D. and Singh S. (2021). Calibrate before use: Improving few-shot performance of language models. arXiv:2102.09690.

Zoph B., Bello I., Kumar S., Du N., Huang Y., Dean J., Shazeer N. and Fedus W. (2022). Designing effective sparse expert models. In *2022 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*. Lyon, France, pp. 1044.

Zulqarnain M., Ghazali R., Hassim Y. M. M. and Rehan M. (2020). A comparative review on deep learning models for text classification. *Indonesian Journal of Electrical Engineering and Computer Science* **19**(1), 325.

A. Appendix

A.1 Datasets used for evaluation

This section contains the original training, validation, and test sets that were augmented. Identifying information, such names of study participants or course numbers, has been redacted. The original label is included, along with the three labels assigned by post hoc reviewers.

A.1.1 Training

1. Should I use Plotly or matplotlib for visualization? [Data]; [Data, Data, Data]
2. would this be better if we had dedicated reactions for good/bad questions/answers? [Other]; [Other, Other, Other]
3. @_ Do you have any of the merch with the new logo done? If so can we see a link/picture? [Other]; [Other, Other, Other]
4. What are some libraries for data visualization in python? [Data]; [Data, Data, Data]
5. Has anyone seen the original Transformers movie? [Other]; [Other, Other, Other]
6. Neural networks can be programmed in both Tensorflow and PyTorch, true or false? [Data]; [Data, Data, Data]
7. What statistical distribution models intervals? [Data]; [Data, Data, Data]
8. How many neurons are contained within the human nervous system? [Other]; [Other, Other, Other]
9. why is it that a snowflake falls in the winter instead of the summer? [Other]; [Other, Other, Other]
10. What does a data science career path look like? [Data]; [Data, Data, Data]
11. When he attached the wires, they made a spark. Does he need more training? [Other]; [Other, Other, Other]
12. Could you explain the difference between Spark and Hadoop? [Data]; [Data, Other, Data]
13. Does anyone know if non-library buildings are open on campus? [Other]; [Other, Other, Other]
14. Is it possible to set up an API in AWS? [Data]; [Data, Other, Other]
15. Has anyone ever trained a transformer model for NLP? [Data]; [Data, Data, Data]
16. What is a simulation and how does it differ from machine learning modeling? [Data]; [Data, Data, Data]
17. Django or Flask? [Data]; [Data, Other, Data]
18. What's the difference between data science and data analytics? [Data]; [Data, Data, Data]
19. What are some people's favorite movies? [Other]; [Other, Other, Other]
20. The wise old oak tree helped you make a decision? [Other]; [Other, Other, Other]
21. What is the best way to learn Tableau and PowerBI? [Data]; [Data, Other, Data]

22. Which programming language is preferable for embedded systems, C or C++? [Other]; [Data, Other, Other]
23. should I just thumbs up every single message or what [Other]; [Other, Other, Other]
24. what is data science? [Data]; [Data, Data, Data]
25. and if we're not, why not? [Other]; [Other, Other, Other]
26. Where was the largest cluster of cases found this month? [Other]; [Data, Other, Data]

A.1.2 Validation

1. R u ready to go? [Other]; [Other, Other, Other]
2. Are people familiar with APIs? Just wondering if I should plan an explanation for how APIs work for Thursday. [Data]; [Data, Data, Other]
3. Does P = NP? [Data]; [Data, Data, Other]
4. I need to train a regression and a classification model. Can this be done in sci-kit learn? [Data]; [Data, Data, Data]
5. What? [Other]; [Other, Other, Other]
6. Is Blender used for creating 3D graphics? [Other]; [Other, Other, Other]
7. What are we going to do for the rest of the meeting? [Other]; [Other, Other, Other]
8. Is clustering an example of supervised or unsupervised learning? [Data]; [Data, Data, Data]
9. Can anyone help with I do not understand any of this at all. The lecture/slides are awful and I've been watching youtube videos getting even more lost. Thanks friends [Data]; [Data, Other, Data]
10. A SQL query walks into a bar, goes up to two tables, and asks "Can I join you?" [Other]; [Other, Other, Other]
11. ajskdl qwerjksd weknwf we wejirknwdfw? [Other]; [Other, Other, Other]
12. I'm looking at adding data science as a second major but I'm really confused by the _ course. Has anyone taken it or is taking it and could tell me what it is? I couldn't find it anywhere on coin. [Data]; [Data, Other, Other]
13. What do you guys think would be some difficult data science questions to answer? [Data]; [Data, Other, Other]
14. Data Science? [Other]; [Other, Data, Other]
15. Which programming language do you prefer: Python, R, Julia, or Matlab? [Data]; [Data, Other, Other]
16. Did you know that just under half of all data science puns are below average? [Other]; [Other, Data, Other]
17. If I perform a chi-squared test and obtain a p-value of 0.02, is that considered statistically significant? [Data]; [Data, Data, Data]
18. are we using my redesign? [Other]; [Other, Other, Other]
19. Or Streamlit? [Data]; [Other, Other, Other]
20. Is it better to do a data science bootcamp or get a master's degree in data science? [Data]; [Data, Data, Data]
21. What is the meaning of life? [Other]; [Other, Other, Other]
22. Who went to the club meeting today? [Other]; [Other, Other, Other]
23. How do I create an SQL database in Azure? [Data]; [Data, Data, Data]

24. Or what would be some questions that people outside the club would be wondering about? [Data]; [Other, Other, Other]
25. The base of the model will be made in CAD - can you create this on your computer? [Other]; [Other, Other, Other]
26. According to statistics is Michael Jordan the best basketball player ever? [Other]; [Data, Data, Data]

A.1.3 Test

1. Is the school open today? [Other]; [Other, Other, Other]
2. When is the meeting time this semester? [Other]; [Other, Other, Other]
3. What is the quantum Fourier transform [Data]; [Data, Data, Other]
4. I tried to push to GitHub but I got a merge conflict error. How do I fix this problem? [Data]; [Data, Other, Other]
5. Are you coming to Big Data Club tomorrow? [Other]; [Other, Other, Other]
6. Will members get those stickers for free? [Other]; [Other, Other, Other]
7. What data science books would recommend for understanding the basic principles? [Data]; [Data, Data, Data]
8. Where can I access datasets like MNIST for image analysis? [Data]; [Data, Data, Data]
9. Can someone help me understand how to SSH into the computing cluster on Friday? [Data]; [Data, Other, Other]
10. Where can I find the link? [Other]; [Other, Other, Other]
11. Which language should I use for matrix methods for data analysis [Data]; [Data, Data, Data]
12. Does anyone have any classes they would recommend for me to take next semester [Other]; [Other, Other, Other]
13. What is a Poisson distribution? [Data]; [Data, Data, Data]
14. Here's my idea: we combine lookup table and machine learning model. Thoughts? [Data]; [Data, Data, Data]
15. I've taken the class before, I really enjoyed it actually. Do you want me to send you the syllabus? [Other]; [Other, Other, Other]
16. Could everyone send times when you are available before the start of the semester? [Other]; [Other, Other, Other]
17. Here's a link on how to make custom Jupyter notebook themes. Big Data Club-themed notebooks, anyone? [Data]; [Other, Other, Other]
18. What's the best way to get user input in Python? [Data]; [Data, Other, Data]
19. Can you volunteer at the accepted students day @_ [Other]; [Other, Other, Other]
20. Which logo won the design contest? [Other]; [Other, Other, Other]

A.2 Final completion model prompts

This section includes final candidate prompts for the Completion Endpoint evaluation. Note that since we included a few "adversarial" examples (e.g. random characters) in the original data and placed no restrictions on the types of augmented examples that GPT-3 was allowed to generate, some of these may not be in the proper form of a question in the English language. Such occurrences are normal, as they still function the same as training examples that are in a question format for the Completion Endpoint (otherwise they would have been filtered out by the genetic algorithm's selection process).

A.2.1 Model 1

1. What is the difference between a neural network and a statistical regression model? [Data]
2. What is the first step when you are solving a math problem? [Other]
3. Explain how to use the R Package Deducer to cite a package. [Data]
4. I'm enjoying the new @pizzahut commercial playing on @midnight right now. "You can't spell 'delivery' without deliver" [Other]
5. What is the difference between a matrix and an array? [Data]
6. What are the two most pervasive problems in data science today? [Data]
7. What is the best way to get started in a new field of research? [Data]
8. How is a lobster like a dinosaur? [Other]

A.2.2 Model 2

1. What's the best way to learn data science? [Data]
2. Is the movie set in America or somewhere else? [Other]
3. What is a logistic regression? [Data]
4. When would I use the elif keyword in Python? [Data]
5. In what year was the first man on the moon? [Other]
6. Quick question: I am a girl who's dad never really had a relationship with me. He only sees me during the winter holidays and has not been a part of my life very much. I have a lot of resentment towards him and have always felt unwanted. However, I have never told him that I feel this way [Other]
7. What is the most powerful feature of yEd? What is a good use for it? [Data]
8. Is it true that a 10x programmer is 10 times more productive than an average one? [Data]

A.2.3 Model 3

1. What is the gradient descent algorithm and how does it work? [Data]
2. I get it. I'd say "no", too. . . [Other]
3. @Caspian I want to thank you and your team for the work you've done so far. The product is fantastic and it makes my life easier as a support engineer. Thanks again! Question: What tools/scripts do you use to build containers? [Other]
4. What is an aggregate function in SQL? [Data]
5. I have been working on my new script for a feature film, a. . . [Other]
6. What's the name of this interview? [Data]

7. I am curious what your thoughts are on the @Yale proposal to expand the legal meaning of sexual consent [Other]
8. Is there a reason why I should buy the sample business plan? [Other]

Cite this article: Balkus SV and Yan D (2024). Improving short text classification with augmented data using GPT-3. *Natural Language Engineering* **30**, 943–972. https://doi.org/10.1017/S1351324923000438