Original Article

Sociological Methods & Research
1–48

DOI: 10.1177/00491241251340608
journals.sagepub.com/home/smr
Sage

Cheap Learning: Maximizing Performance of Language Models for Social Data Science Using Minimal Data

Leonardo Castro-González[1,2], Yi-Ling Chung[1], Hannah Rose Kirk[1,3], John Francis[1], Angus R. Williams[1], Pica Johansson[1], and Jonathan Bright[1,4]

Abstract

The field of machine learning has recently made significant progress in reducing the requirements for labeled training data when building new models. These "cheaper" learning techniques hold significant potential for the social sciences, where development of large labeled training datasets is often a significant practical impediment. In this article we review three "cheap" techniques that have developed in recent years: Weak supervision, transfer learning and prompt engineering. For the latter, we also review the particular case of zero-shot prompting of large language models. For each technique, we provide a guide of how it works and demonstrate its application and the presence of systematic biases across two different and realistic social

[1]Public Policy Programme, The Alan Turing Institute, London, UK
[2]School of Geographical Sciences, University of Bristol, Bristol, UK
[3]Oxford Internet Institute, University of Oxford, Oxford, UK
[4]Zug Institute for Blockchain Research, University of Lucerne, Luzern, Switzerland

Corresponding Author:
Leonardo Castro-Gonzalez, Geography Department, School of Geographical Sciences, University of Bristol, Beacon House, Queens Rd, Bristol BS8 1QU, UK.
Email: leonardo.castrogonzalez@bristol.ac.uk

Data Availability Statement included at the end of the article

science tasks paired with three different dataset makeups. We show good performance for all techniques and we demonstrate how prompting of large language models can achieve high accuracy at very low cost, but biases must be considered.

Keywords

Social data science, text classification, week supervision, transfer learning, prompt engineering, language models

Introduction

The systematic analysis of large-scale text data is a growing part of sociology and social science research. The rapid digitisation of social life has meant that daily activity has increasingly left digital "traces" (Boyd and Crawford 2011). Across the social sciences, many of the core objects of disciplinary study can be interrogated through analysis of large datasets created by these traces. Examples abound: Large-scale corpora of news articles; datasets of political manifestoes and speeches; textual interactions between individuals around romantic relationships, job searches, discussions around health and climate change, to name but a few (Beigman Klebanov and Madnani 2020; Burscher et al. 2015; Gangula et al. 2019; Jensen et al. 2022; Scharkow and Vogelgesang 2011; Souma et al. 2019; Stede and Patz 2021).

In the past, large-scale text corpora were addressed through equally large-scale manual coding efforts (see, for example, Lehmann et al. (2023)), whereby texts are reviewed by teams of trained coders. However, around a decade ago, work emerged pointing to the potential for automatic content analysis to shed light on the social dynamics contained within these datasets, making use of insights from the field of machine learning (Grimmer and Stewart 2013; Grimmer et al. 2021; Wiedemann 2013). By now a considerable body of work has developed methods for automating content analysis in the social sciences (Lichtenstein and Rucks-Ahidiana 2021; Nelson and Taneja 2018; Németh et al. 2020), which offers obvious potential benefits in terms of saving time and effort, and thus allowing for analyses at a scale that would be beyond most teams of manual coders (Burscher et al. 2014).

However, while the potential opened up by this automatic work is important, there remains a weakness in the automatic content analysis based on the "conventional" paradigm of machine learning (as we will refer to it here): It still in reality contains a considerable manual component. As we will describe

more fully below, in order for a classifier to be trained to identify a given concept, labeled training data is required, often in large quantities. Hence a significant time investment is required before such a classifier can be developed. Whilst such upfront work can be justified in industry contexts that may require millions of data points to be labeled automatically on an ongoing basis, many analytical research tasks in the social sciences could actually be achieved with this initially labeled dataset, rendering the training of the model somewhat irrelevant. Hence, the practical advantages of this type of machine learning approach in the social sciences are not always clear cut, with some saying that the time and financial savings from automated methods are exaggerated (De Grove et al. 2020; Németh et al. 2020) and others preferring methods granting researchers greater measurement validity (Baden et al. 2022).

The heavy data requirements of conventional machine learning approaches have led to considerable research interest in techniques that might reduce data requirements without sacrificing performance or analytical rigor (Al-Jarrah et al. 2015; Galal et al. 2024; Goldsteen et al. 2022). In computer science, such techniques are now developing very actively, with considerable advances and a wide array of new techniques developing. However, only a few papers have emerged trying to make these techniques accessible to social science researchers (Costantini et al. 2023; Macanovic 2022; Molina and Garip 2019; Wankmüller 2022), and the uptake remains patchy at best.

The aim of this article is to remedy this deficit and offer an introduction to these "cheap" learning techniques, as compared to their more expensive conventional counterparts. We discuss three main approaches. First, we look at weak supervision, a method for formalizing background knowledge of a subject into a systematic approach to data labeling, making use largely of keyword-based approaches. While it might not be the technique that offers the highest performance, it is transparent and easy to implement, retaining the simplicity of the dictionary technique but assuaging some of the criticisms of how dictionaries are both arbitrary and difficult to validate (Grimmer and Stewart 2013). Second, we discuss transfer learning, a technique which involves taking a model trained in one context and applying it to another, making use of a minimal amount of retraining data. We show how good results can be achieved with small amounts of novel data by building on pre-existing models. Finally, we discuss "prompt engineering": Prompting large language models (LLMs), such as the GPT family created by OpenAI, as a means of content classification. We look at both direct use of generative models for such classification (sometimes called "zero-shot" learning), and also how these techniques can be combined with some elements of transfer

learning. LLMs in particular have generated much excitement as a very cheap method of content classification (Gilardi et al. 2023); however, we show that at the moment they do not outperform more traditional approaches for all tasks.

Each one of these techniques offers the potential for realizing the benefits of supervised machine learning, but at less cost in terms of human labeling, hence offering the tantalizing potential of finally realizing the potential of automatic content analysis in sociology and the social sciences in general. For each one of these techniques, we describe how it functions, discuss approaches to validation, and discuss common potential use cases. In an empirical section, we then illustrate using existing annotated datasets how good performance can be achieved with a minimal amount of labeled data (up to 1,024 data points), as well as showing how performance improvements relate to increases in labeled data. We also compare all of these techniques to the Naïve Bayes approach and to a logistic regression classifier, two classic methods from conventional machine learning Kibriya et al. (2005) and Aborisade and Anwar (2018). Although these methods were developed more than two decades ago, their simplicity of training and deployment makes them a good baseline to compare with newer and more sophisticated methods in a low-resource context.

We complement our performance analysis with an additional experiment to study the biases each technique can have in classifying text for each task. While the most recent OpenAI's GPT models have been used to annotate text (Gilardi et al. 2023), the potential systemic biases they have at an annotation task can lead to incorrect results (Ashwin et al. 2023; Spirling 2023). With this bias analysis, closed models like OpenAI's GPTs can be compared to more classical machine learning techniques in another useful dimension for social scientists besides their performances.

We should note the existence of a handful of other papers that have made similar contributions to our own (Terechshenko et al. 2020; Treviso et al. 2023). Whilst these are valuable works, our paper is distinctive in a number of ways. First, these papers address one technique exclusively, whereas ours compares three at the same time. This comparison allows us to explore the "labelling budget" (the amount of manually labeled data required for each technique), as well as discussing how different methods perform well in different circumstances. Second, our analysis is not limited to the performance of labeling data but also studies the systematic bias of each technique. Finally, alongside our paper we release code that will allow other researchers with a practical working knowledge of the Python language to implement these techniques in their own research.[1]

With this paper, we contribute by giving sociologists and social scientists tools and strategies to address challenges where data scarcity is found. By offering available online resources and a performance and bias analysis done on two well-known examples in the literature (Maas et al. 2011; Wulczyn et al. 2017), we hope for social scientists to apply these strategies in key challenges where data scarcity and bias might be embedded into the problem, like interview analysis of under-represented groups (Ashwin et al. 2023; Atari et al. 2023; Bonikowski and Nelson 2022; Deterding and Waters 2021; Small and Calarco 2022) or text analysis and labeling using historical text archives (Jaillant 2022; Marciano et al. 2018; Rahal et al. 2024).

While acknowledging the importance of unsupervised machine learning approaches (Hutto and Gilbert 2014; Shelke et al. 2022), we chose not to work with these techniques. Given that unsupervised learning discovers structure in data rather than labeling according to prior theoretical concepts of interest, we consider unsupervised learning techniques working fundamentally different to the features of supervised learning we are interested in testing, thus falling out of the scope of this study. In the same way, we are also not including open-source LLMs for this study. On the one hand, we are interested in showing the monetary costs, performance and potential biases of using the API of closed models like the OpenAI's GPT-3.5 (ChatGPT) and GPT-4.0, which have grabbed much attention (Kalla et al. 2023) in the academic and non-academic world. On the other hand, we decided to centre our attention on the mentioned closed LLMs to avoid this study becoming another comparison between the different LLMs now available for researchers (Yu et al. 2023).

Our paper is structured in the following way. In Section "Cheap Learning Techniques" we give a general overview of how we define cheap learning techniques, and also justify our focus on the techniques we are looking at. Then, we provide a guide to each technique, describing how they function, discussing approaches to validation, and discussing common potential use cases. In Section "Methods" we describe our approach to the empirical illustration of the models and describe the data we make use of. Finally, in Section "Results", we illustrate the use of these techniques using existing annotated datasets to showcase how good performance can be achieved with a minimal amount of labeled data (as well as showing how improvements in performance relate to increases in labeled data). We also discuss concerns around balanced and unbalanced data, showing how these might affect preferences for models. We conclude by providing perspective on directions in the field, especially in the light of increasingly powerful and available LLMs.

Cheap Learning Techniques

In this paper, we address methods for automating content analysis in the social sciences using techniques from the field of machine learning. Our focus is on algorithms that can be implemented "cheaply" when compared to what we will define as the "conventional" approach to automatic content analysis, which involves training a model from scratch with a large labeled training dataset (though we refer to it as "conventional," we should also say that it is an approach that is less and less the norm within computer science). As highlighted in the introduction, we address three main approaches: Weak supervizion, transfer learning and prompt engineering. We regard these algorithms as "cheap" to implement because they typically require much less labeled data to achieve good performance than the conventional approach (though all of them still require some such data, particularly for validation). In this section, we will provide an overview of the three techniques we are introducing in the paper. For each technique, we provide a general overview of the history and motivation, and provide a guide on how to implement it, including notes on how to approach evaluation of the technique. We also reflect on the cost of each algorithm, and the amount of computing power required to implement it (with the proviso that, of course, computing power increases at a regular pace). Finally, we provide our thoughts on how each technique might adapt to different problem sets in the social sciences.

We should note that we do not address "unsupervised" machine learning, a technique which involves no manual labeling (at least in the input stage). We omit this technique here partly because it has already been covered by a relatively extensive set of social science applications (e.g. Berry et al. 2019; Molina and Garip 2019; Waggoner 2021), and also because we do not consider it a direct competitor to supervised machine learning. While unsupervised learning offers the potential to group text documents into lexically similar sets (often referred to as "topic modelling") it does not provide the direct link to theoretical categories of interest which is offered by supervised machine learning: That is to say there is no guarantee that identified topics will map neatly to theoretical concepts and hence be directly applicable to an analytical social science question. Therefore, while the technique is an excellent approach for exploratory analysis of a large dataset, the lack of ability to define concepts at the outset can make it challenging to use topic models for explanatory hypothesis testing (though we do note the emergence of seeded topic models (Watanabe and Baturo 2023) as a potential partial remedy to this

problem), and of course post hoc labeling of topics can allow them to be built into an explanatory framework.

We contrast all of these approaches to the conventional approach to text classification through machine learning, which is worth briefly reviewing here (the subject has already been treated extensively in, for example, Kotsiantis et al. (2006), Jordan and Mitchell (2015), and Mahesh (2020)). This approach seeks to produce a model that, given an input text, estimates the probability that this text belongs to a given analytical category. Such models are based on input data: A corpus of text that has been labeled by hand such that each piece of text is assigned to a category. One classic example, that we make further use of below, would be to label a given piece of text as containing "hate speech" or not, to produce a text-based classifier that can automatically detect hateful text.

The corpus of text data is typically divided into three groups: Training data, development data, and test data. Training data are the data used to train the model to begin with: The model's decision making will be based on patterns learnt from these data. There are a wide variety of approaches to learning a model from training data that we will not seek to review here (good overviews are provided by Cherkassky and Mulier (2007); Jordan and Mitchell (2015); Kotsiantis et al. (2006); Mahesh (2020); N and Gupta (2020)). Broadly speaking, they will rely on identifying differences in linguistic patterns between the different classes of data for which a classification is sought (for example, hateful text is likely to contain different words to non-hateful text). Development data are data used during the process of model training to provide an estimate of the model's performance (with a variety of metrics typically calculated for this evaluation—see Fatourechi et al. 2008; Hossin and Sulaiman 2015). These development data allow for decisions to be made during the process of model training, for example selecting appropriate hyperparameters (Kandel and Castelli 2020; Keskar et al. 2017; Luo 2016). Test data, finally, are used to make an assessment of model performance once the process of training is complete. Test data are typically kept separate until the final model evaluation to guard against "overfitting," a phenomenon whereby (through repeated trial and error with different models and different hyperparameters) a model is developed that performs well on the development dataset but generalizes poorly to unseen data.

When implementing such a process within the social sciences, the amount of data required (the "labelling budget," Guan and Koudas 2022; Lindstrom et al. 2013) is the core practical concern. High-quality labeled data is expensive and time-consuming to generate. Firstly, coders need to be recruited and trained on the concepts of interest, and need to establish a common

understanding of the concepts they are coding (often measured through high inter-coder reliability scores). Once this process is complete, labeling needs to take place, which is a time-consuming and repetitive manual task (and gold standard scientific journals often require data to have been coded by multiple coders, Geva et al. 2019; Hube et al. 2019). The exact amount of time will of course be task-dependent (varying by, for example, the amount of text that needs to be reviewed, the amount of potential classes, and the complexity of the coding task). However, there is a general expectation that the process will be time-consuming, driven by the fact that the data requirements of high-performance classification models are typically large (Anaby-Tavor et al. 2020; Sun et al. 2019; Zheng and Jin 2020), and that increasing the size of the training dataset is the standard recommendation for improving performance (Hsieh et al. 2023; Raffel et al. 2020; Shams 2014).

The approaches we review below are all motivated by a desire to reduce the cost of the labeling budget to produce a high-performing model, particularly the training and development datasets (a high-quality test dataset for evaluation remains essential to the process however). Each of the techniques approach this task in a different way: Weak supervision does not necessarily reduce the size of the labeling budget, but makes large training datasets cheaper to put together; transfer learning seeks to leverage other pre-existing models to reduce the requirements for novel labeled data; while prompt engineering seeks to exploit "emergent" properties of LLMs that have been trained for generic language production tasks but can also be leveraged for classification.

As a core element of our paper is cost, we should also mention some other costs that are relevant here beyond those needed for the creation of labeled data. There are hardware costs implied in some of our work below: Though much of what we describe can be implemented on a generic personal laptop, some may require the use of an institutional or cloud server. For commercial language models, there are also costs in terms of API access that could mount up if there are requirements to label large volumes of data. Finally, there are some time costs required for training models, that can mount up if a long cycle of experimentation is required to find the ideal configuration of hyperparameters. We will also comment on these costs for all of the techniques we discuss.

Weak Supervision

Proposed in Ratner et al. (2016) and extended in Ratner et al. (2019), weak supervision is an approach to text classification that seeks to cheaply build labeled data using heuristic rules based on pre-existing background

knowledge of the concepts being labeled. Data labeled in this fashion can either be applied directly to an analytical task, or leveraged to build a further classification model.

As a simple example of weak supervision, we could imagine a task based on classifying international news articles based on which country they relate to. A first step to build a model through the conventional machine learning approach would be to take a dataset of news articles and label by hand the country to which they relate (conscious that an article might relate to more than one country). However, this approach essentially ignores background knowledge that will be largely obvious to people with domain expertise: We know that (for example) an article containing the keyword "Canberra" is most likely about Australia, whilst one containing the keyword "Boston" likely refers to the United States. Integrating this knowledge would seem like a valuable efficiency.

Using keywords to classify content in this fashion, sometimes called the "dictionary" approach, does of course have a long history in the social sciences and indeed many early approaches to content classification were based on keywords (Hulth and Megyesi 2006; Onan et al. 2016). However, such approaches received strong criticism as being hard to evaluate properly (Grimmer and Stewart 2013). In the absence of manually labeled data, it was hard to measure common evaluation metrics such as the precision and recall,[2] or understand how many false positives and negatives might be generated (for example, an article on Boston, Lincolnshire in the UK would be obviously misclassified by the above approach).

Weak supervision can be viewed as a method that seeks to circumvent some of these difficulties: Benefiting from the simple but powerful domain knowledge that experts can bring to bear on classification, without sacrificing the rigor or evaluation metrics used in more conventional machine-learning approaches. It has already been applied in a wide variety of different contexts like text data in the fields of medicine and health sciences (Fries et al. 2021; Silva et al. 2022), chemistry (Mallory et al. 2020), detection of fake news (Shu et al. 2021), and more generally to study sentiment analysis (Jain 2021), and ontology and computational linguistics problems (Berger and Goldstein 2021; Maresca et al. 2021).

Weak supervision is based on two main components: Labeling functions (LFs) that encode background knowledge into formalized rules; and a probabilistic model that is trained with the noisy, overlapping labels obtained from applying the functions to the labeled data set, and its respective prior labels.

LFs are abstract rules that take an unlabeled data point and either assign a label if a criterion is met or abstain from assigning it otherwise. The presence

or absence of keywords contained in text data is one of the most obvious types of such rules, however, there is no need to only make use of keywords. Regular expressions, Named-Entity Recognition (NER) systems, results from other models such as sentiment classifiers, the length of the document to be classified, or its source; any of these document features can be incorporated into a LF. This diversity of potential criteria makes it possible to have topic-specific sets of LFs obtained from the particularities of the topic to be classified, going a step beyond the dictionary approach. Some examples of simple LFs that we make use of in the experimental section of the paper are presented below. For further examples, the list of all the LFs used for the classification tasks presented in the paper can be found in the Online Appendix A.4 of the online Supplemental Material.

- LF1: `return NEGATIVE if polarity(text) < -0.25 else ABSTAIN`
- LF2: `return POSITIVE if regex found in text else ABSTAIN`
- LF3: `return ATTACK if length(text) > LIMIT else ABSTAIN`

In the examples above, variables `NEGATIVE` and `POSITIVE` refer to the classes used for the binary movie sentiment classification exercise, while the `ATTACK` refers to one of the classes used for the Wiki Personal Attacks exercise explained below. In the same way, the variable `ABSTAIN` refers to the value that the function returns when it abstains from classifying a text. The polarity function refers to the sentiment analysis function to measure the polarity of a given text between -1 and 1. `regex` is used in this example as any regular expression that would refer to a non-abusive text, e.g. "This is such a good idea!". LF3 is a good example of how domain knowledge can be used in weak labeling. As seen in the Wiki Personal Attacks exercise (Wulczyn et al. 2017), cyberbullying can take many forms (Sathyanarayana Rao et al. 2018). One of them is to repeat a sentence or a set of words in a comment until the maximum number of characters allowed is hit. LF3 accounts for that, labeling unusually long texts as personal attacks by measuring the length of the text and checking if it is greater than the maximum length allowed (`LIMIT`).

The exploration subset of the dataset serves to fine-tune or debug the set of LFs. Weak supervision enables measuring *coverage* and *overlap* of these functions in the different sets (details of the LFs used in this work are

presented in Online Appendix A of the online Supplemental Material). Coverage for a specific LF indicates the percentage of data points it labels without abstaining. On the other hand, overlap represents the percentage of data points where two or more LFs do not abstain. Ideally, for a given class, we aim for a set of LFs that collectively cover the entire exploration set without significant redundancy in their actions, measured by a high overlap.

Once the LFs are created, they are then applied to the training set, generating a label matrix. Each row in the matrix represents a data point, and each column represents a LFs. Since the training data is already classified, this label matrix is used to compute weights for each LF using a Bayesian probabilistic model, called a `LabelModel` by the authors (Ratner et al. 2019). The overall model, incorporating the LFs and their respective weights, is then used on the test set for classification.

The outcome of a weak supervision process will be a partially labeled dataset (as we expect the LFs to abstain in some cases rather than suggesting a label). Depending on the coverage achieved, the partially labeled set may already be enough to answer the question at hand. If it is not, a common next step would be to train a conventional machine-learning model to label the remaining data points. The success of the approach will depend on whether the partially labeled data is good enough to produce a strong classification model, though as we will present below the approach shows good potential for a number of tasks.

In addition to its very transparent and simple approach to producing labeled data, a further advantage of weak supervision lies in its lack of significant computing costs. Developing and applying LFs are computationally cheap tasks that can be tackled on any standard consumer laptop and labeling even a very large dataset in this fashion would take little time or processing power.

Transfer Learning

The approach to conventional machine learning described above starts from a blank page. Given a classification problem, and a large enough pre-existing labeled dataset, a model is trained from scratch, experimenting with different hyperparameters, trying to achieve the best possible accuracy on a development dataset. The final evaluation score is then calculated on previously unseen test data.

Transfer learning starts from a different premise (Pan and Yang 2010; Weiss et al. 2016; Zhuang et al. 2020). Rather than starting from scratch, it

looks for pre-existing models (potentially from closely related source domains), and seeks to leverage the information already contained within them (Bashath et al. 2022; Neyshabur et al. 2020; Weiss et al. 2016). A model built on this approach will still require some novel training and development data; but far less than in the conventional approach. Furthermore, given the variety of input data, transfer learning also offers a method for obtaining models with better generalisation properties. These advantages mean that transfer learning is a rapidly growing field of work (Chung et al. 2020; Conneau and Lample 2019). Transfer learning has shown great success in several real-world applications, such as text classification (Ali et al. 2022; Raffel et al. 2020), image recognition (Guo et al. 2019; Kolesnikov et al. 2020), and natural language generation (Golovanov et al. 2019; Raffel et al. 2020).

There are a variety of ways in which a pre-trained model might be selected. Domain specificity is obviously a concern: If seeking to build a classification model for hate speech, selecting a pre-existing hate speech classifier may be a sensible place to start. Beyond this, language ability (e.g., monolingual or multilingual), model architecture (e.g., model size, number of parameters and speed), computation power (e.g., hardware and number of GPUs required) and model performance (e.g., accuracy) all might be a consideration. With recent advancements in natural language processing, transformer-based LLMs that have been developed and pre-trained on massive corpora (e.g., Common Crawl (Raffel et al. 2020)) using various pre-training objectives (Devlin et al. 2019; Dong et al. 2019) are becoming more and more popular as the starting point of transfer models focused on text classification. These general-purpose models lack any domain specificity, but they have nevertheless demonstrated strong performances and can leverage knowledge from source data to downstream tasks through fine-tuning methods (Houlsby et al. 2019; Peters et al. 2019). As a general rule larger models mean better performance whilst requiring high computation power and memory consumption. A key question for many in the social sciences will be whether the model being selected can be run on a researcher's laptop or whether a cloud computing platform that can offer greater performance (such as AWS, Azure or GCP) will be required (Kamal et al. 2020; Pierleoni et al. 2019). However the technology here is evolving constantly, and the possibility of running ever larger models on contemporary consumer hardware is always increasing (Pierleoni et al. 2019).

Once a base model has been selected, it is then fine-tuned on the training set. The ideal size for a training set is not easy to know in advance, but in the experiments section below we will present a few rule of thumb examples for

the ideal size. This fine-tuning has the aim of improving model performance for the specific task at hand. There are a variety of different ways to approach this task, tuned through hyperparameters, and many of these have conventional values that have been found to work reasonably well in a variety of settings. Training epochs indicate the number of times a model sees or passes over the whole training dataset, often in the range between 1 and 5. Learning rate refers to how much to adjust the model (i.e. model weights) with respect to the knowledge learned each time the model weights are updated (Bengio 2012), with 0.0001 and 0.0003 sometimes suggested as working well for many problems.[3] A learning rate that is very large could make rapid divergence of model weights, likely resulting in unstable training and a sub-optimal solution, whereas a learning rate that is too small may require longer training time for models to converge or arrive only at a local minimum. Batch size decides the number of instances processed before the model weights are updated, which is generally a power of 2 in the range between 8 and 128. While increasing batch size generally requires less training time, it consumes more memory and may not lead to high accuracy (Kandel and Castelli 2020; Keskar et al. 2017). A random seed will also indicate the initial state of the model parameters.

Finally, the finished model can be evaluated to assess if a model is learning as expected (e.g., whether the model is overfitting or not). This includes applying the trained model to the test set using various metrics, including precision, recall, F1, and accuracy. Graphical representation is another helpful way of diagnosing model performance, for instance, through learning curves of model performance over the training process, confusion matrix and receiver operating characteristic curve. These methods are the same as those used in the conventional approach described above, hence we will not describe them in detail here.

Prompt Engineering

The final technique we will introduce is prompt engineering, also known as in-context prompting or prompt learning. This is a technique for guiding LLMs to perform a task (e.g., classification) via textual prompts (Liu et al. 2023). Training a prompt model is similar to transfer learning in a way, in that we are exploiting pre-existing knowledge within a model. However, unlike the above approach, we no longer seek to train a model in a classic sense, but rather exploit prompts as guidance to elicit the generative potential already held within LLMs to generate answers to classification questions.

In theory, a nuanced prompt tailored to a task can produce high performance in a classification task even more cheaply than transfer learning. With handcrafted or automatically generated prompts, prompt engineering is particularly beneficial for scenarios/domains where (1) language models show strong performance and (2) labeled examples are scarce or not available (Hu et al. 2022; Liu et al. 2021; Schick and Schütze 2021a). Such prompting has been explored and has shown promising success in several domains, such as text classification (Ali et al. 2022; Raffel et al. 2020), image recognition (Guo et al. 2019; Kolesnikov et al. 2020), and natural language generation (Golovanov et al. 2019; Raffel et al. 2020).

Unlike transfer learning, in prompt engineering a language model is instructed to perform with the help of task description as an auxiliary training signal using "pattern-verbalizer pairs" (Schick and Schütze 2021a, 2021b). A pattern is a natural language prompt or an expression that transforms inputs into cloze-style tasks (i.e. a portion of a passage is removed or closed, and the task is to predict the removed chunk that would best fit the surrounding context). A verbalizer refers to mapping a label/class (e.g., movie) to a list of task-representative words, also called label words, (e.g., action, drama and adventure). For example, given a binary movie sentiment classification task (as we explore below), an *example input in italics*, taken from IMDb Movie Review Sentiment dataset (Maas et al. 2011), with a ***pattern in bold*** is shown as follows:

> *Two Hands restored my faith in Aussie films. It took an old premise and made it fresh. I enjoyed this movie to no end. I recommend it to those people who like Guy Ritchie films. Bryan Brown was fantastic and just about perfect in a role tailor made for him.* ***It was? Negative or Not Negative*** *[MASKED]*

The answer is masked and converted into a class based on a verbalizer, for instance, "Positive": "good," "great;" "Negative": "bad."

Writing a good prompt is crucial for effectively giving instructions and obtaining accurate responses from language models. While interest in prompt design has grown considerably in various domains, such as text classification (Hu et al. 2022; Wang et al. 2019), and knowledge probing (Davison et al. 2019; Petroni et al. 2019), studies on systematic evaluation of the effects of prompts are limited. We provide several heuristics that can help write a good prompt. First, a prompt is generally clear and specific with relevant background information provided. A prompt can be written in the form of a statement or question. Second, when dealing with complicated tasks or multiple sub-questions, breaking down the tasks into sub-tasks

can help models tackle each task one by one (Wei et al. 2022). In Press et al. (2023)'s study, they find that asking language models directly how to write a good prompt for a given task of interest and then writing a prompt following what the language model responded can yield better answers. Third, if the response is required to be in a specific format, delineate the format with examples (e.g., make step-by-step reasoning, and provide a pros and cons comparison). Lastly, a good prompt generally benefits from several iterations and experiments by rephrasing the prompt and providing additional information. Much like tuning the hyperparameters in a conventional machine learning model, there is no one formal way of creating the ideal prompt: A degree of experimentation is always required.

Verbalizer design is also worth considering. A verbalizer maps a label to a list of label words which are recommended to have diverse coverage and little subjective bias for eliciting comprehensive semantic information from language models (Hu et al. 2022). Previous work experimenting with multiple label words for each label shows that the optimal size of label words is generally around three, while varying the number of label words does not lead to significant improvements (Schick et al. 2020; Shin et al. 2020).

Just as with the other techniques above, labeled test data is essential for the evaluation of a model based on prompt engineering. However the amount of training data required is open to question. Up until very recently, prompt engineering models also largely made use of some training data to boost performance through fine-tuning, similar to transfer learning above. But the emergence of openly available LLMs such as the GPT family from OpenAI has challenged this paradigm, creating the potential to prompt models without any pre-training at all, sometimes called "zero-shot" learning (Ratner et al. 2019; Wei et al. 2021). As we will show below, performance can be high even in these zero-shot cases.

All LLMs have a set of hyperparameters that also need to be considered when setting them up for a classification task (Bommarito II and Katz 2022; Rehana et al. 2023). Probably the most crucial to mention here is "temperature" which affects how deterministic the output of the model is. More specifically, given that the input into a LLM is a sequence of text (sometimes called a "context window"), the output of the model is a list of words with associated probabilities of being the next word in the sequence. The model then picks a word from this list based on the temperature, with a higher temperature making it more likely to pick lower probability words. For creative writing tasks, setting a relatively high temperature seems to generate better performance (Caccia et al. 2020; Roemmele and Gordon 2018); for

classification tasks, lower temperatures seem like a better place to start (Rehana et al. 2023). However, the "best" temperature is an open question and hard to define in abstract.

Similar to prompt engineering, zero-shot prompt engineering is particularly beneficial for scenarios where (1) language models show strong performance and (2) labeled examples are scarce or not available (Hu et al. 2022; Liu et al. 2021; Schick and Schütze 2021a). Furthermore, while zero-shot prompt engineering can benefit from narrow prompting aimed at a specific task (Liu et al. 2023), broader prompting aimed at eliciting wider cognitive abilities in LLMs has also seen success (Kojima et al. 2022).

A few final comments on other resources required for prompt engineering are worth mentioning. While developments in the area of model miniaturization are continuing rapidly (Bai et al. 2022), at the time of writing running the latest generation of LLMs on a consumer laptop was not a realistic possibility, hence a server running a GPU was still likely a requirement.[4] LLMs are also accessible through APIs, however as we will describe below these often have a cost attached to them.

Methods

Having introduced the three cheap learning techniques under study in this article, in the remaining part of this paper we want to provide an illustration of them in action, by applying them to two example binary classification tasks related to personal abuse and the sentiment of a movie review. We have chosen these two tasks because we believe they are relatively standard examples of the types of classification task that are appropriate for the social sciences.

In addition to providing an illustration of the techniques in action, our experiments will also help to explore the extent to which these techniques are truly "cheap," by exploring the amount of training data required to achieve different levels of performance, which we refer to as the labeling budget (noting again that for all of the methods we discuss there would still be a requirement to develop a test dataset).

In this methods section, we describe our approach to these experiments: We look first at the datasets we have chosen and how we sample from them, then describe the classification tasks we seek to perform, then discuss the computing infrastructure used for the experiments, and also explore the particular hyperparameters we have made use of in our experiments.

Data

In our experimental section we address two classification tasks: Binary abuse classification and movie review sentiment classification. In order to conduct experiments on these tasks, we make use of existing publicly available labeled datasets. For the abuse classification task we make use of the "Wikipedia Talk: Personal Attacks" dataset (Wulczyn et al. 2017), containing 115,864 comments from English Wikipedia labeled according to whether they contain a personal attack (no personal attack—88.3%, contains personal attack—11.7%). This dataset is a popular one among social scientists, already employed in a variety of studies (Dinan et al. 2021; Sarwar et al. 2022; Tay et al. 2021a, 2021b; Xu et al. 2021).

For the movie sentiment task, we rely on the "IMDb Movie Review Sentiment" dataset (Maas et al. 2011), containing 50,000 movie reviews labeled according to whether they have a positive sentiment or negative sentiment (negative sentiment—50%, positive sentiment—50%). The dataset has also been extensively studied in the literature (Birjali et al. 2021; Priyadarshini and Cotton 2021; Saunshi et al. 2020; Tay et al. 2021b; Yu et al. 2021).

The two chosen datasets have a number of differences, the most obvious being the context in which text entries were written. The Wikipedia Personal Attacks dataset entries are comments on an article, that could be directed at another user or the article's content, whereas the IMDb Movie Sentiment entries are user reviews, all targeted at some piece of media, usually providing some kind of detailed critique. This difference in context could suggest a greater range of types of responses, and possibly differentiation between classes, in the Wikipedia dataset compared to the IMDb dataset.

This context is reflected in the length of the entries across the two datasets (Table 1), with entries from the IMDb dataset being over three times as long as entries from the Wikipedia dataset on average (1297 characters vs. 402 characters). Furthermore, the balance of positive and negative labels in the datasets represents another difference. While the IMDB Movie Review Sentiment dataset has an even 50/50 split between labels, the Wikipedia Personal Attacks dataset is unbalanced, which provides another dimension to examine performance between techniques over. Evaluating the chosen cheap learning techniques across datasets with these differences provides a more generalisable view into the pros and cons of these techniques across different scenarios a social scientist might encounter.

For both datasets, we use the existing train splits (breakdowns visible in Table 2) to create seven training sets of increasing size ([16; 32; 64; 128;

Table 1. Statistics of text length across labels for the two datasets.

		Text length (characters)			
Dataset	Label	Mean	Std Dev	Maximum	Minimum
Wikipedia personal attacks	All	402	734	10000	1
	Positive	341	938	10000	5
	Negative	410	702	10000	1
IMDb movie review Sentiment	All	1297	980	14106	32
	Positive	1303	1014	13593	65
	Negative	1271	928	8698	32

Table 2. Counts of dataset entries across splits and labels.

Dataset	Split	Label		
		All	*Positive*	*Negative*
Wikipedia personal attacks	All	**115,864**	**13,590 (11.73%)**	**102,274 (88.27%)**
	Train	**69,526**	8,079 (11.62%)	61,447 (88.38%)
	Test	**23,178**	2,756 (11.89%)	20,422 (88.11%)
	Development	**23,160**	2,755 (11.90%)	20,405 (88.10%)
	Exploration (WS)	**100**	12 (12%)	88 (88%)
IMDb movie review sentiment	All	**50,000**	**25,000 (50%)**	**25,000 (50%)**
	Train	**25,000**	12,500 (50%)	12,500 (50%)
	Test	**12,500**	6,250 (50%)	6,250 (50%)
	Development	**12,500**	6,250 (50%)	6,250 (50%)
	Exploration (WS)	**100**	92 (92%)	8 (8%)

The exploration data set is exclusive for the weak supervision technique.

256; 512; 1,024]), in order to assess performance at different labeling budgets. For the Wikipedia Personal Attacks dataset, we create two versions of these seven training sets: One with the original balance of labels (11.7% positive, 88.3% negative), and one with an even balance (50% positive, 50% negative), in order to provide insight into the impact of balanced vs. unbalanced data on technique performance. As the original test data is

large and performing inference on such large data is time-consuming, we create a final test set for each dataset from a 10% stratified random sample (preserving the original class balance) of the original data, resulting in a final test set of 2,316 entries for Wikipedia Personal Attacks, and 1,250 entries for IMDb Movie Review Sentiment.

An extra split is added exclusively for the weak supervision technique. This labeled *exploration* test is intended to test and fine-tune LFs then used in the training process. In our case, we take 100 data points for exploration of each task. Details of these sets are detailed in Table 2 and these additional points should be considered in the labeling budget. The exploration set is drawn from the training set, making sure that the data points in this set never overlap with the 7 training sets mentioned in the previous paragraph.

Model Selection and Training Details

For the conventional baseline models, we develop a Multinomial Naïve Bayes classifier (typically used for text classification—e.g., to represent how many times a word appears in a document) built based on Bayes's theorem (Kibriya et al. 2005).[5] A deeper analysis on why we choose the Multinomial Naïve Bayes and not other options like the presented in Rennie et al. (2003) and Fatourechi et al. (2008) can be read in the Online Appendix A.3 of the online Supplemental Material. For the logistic regression classifier, we construct a logistic regression with an L2 penalty parameter cross-validated to maximize the used F1 macro score.[6] These methods are selected because they can be easily developed and speedily trained without using GPUs. We use TF-IDF to convert data into vectors/features.

For weak supervision, we use the `LabelModel` presented in Ratner et al. (2019) and Ratner et al. (2016) and implemented in the Snorkel package (Ratner et al. 2017). This model takes the label matrix obtained by applying the LFs to the training data set and uses a Bayesian probabilistic model and a matrix completion algorithm to obtain a set of weights. These weights are trained to minimize the error between the golden labels from the training data set and the labels obtained by the weighted sum of LFs over the data set. Once the model is trained, it is applied to the test set.

For both transfer learning and prompt engineering, there is a need to choose a pre-existing model. For both techniques, we chose pre-trained `Distilbert-base-cased` (a distilled version of BERT, Devlin et al. (2019)) as a base model since it retains 97% of BERT performance on several natural language tasks while being 40% smaller and 60% faster

(Sanh et al. 2019). Such characteristics are in line with our goals of developing applications with economic budgets requiring less training time, computation and memory. In addition to the base model, we conducted additional sets of experiments using `DeBERTa-v3` (He et al. 2021) and `GPT-2` (Radford et al. 2019) for transfer learning and prompt engineering, respectively. DeBERTa-v3 is a DeBERTa model with improvements based on replaced token detection and a new weight-sharing method. GPT-2 is an autoregressive transformer-based language model trained on a dataset of 8 million web pages using a next-word prediction objective. While DeBERTa-v3 and GPT-2 have larger model sizes and slightly different model structures and training procedures, both have shown superior performance on downstream tasks (Rodríguez-Sánchez et al. 2022). By including a variety of starting points, we can study the impacts of using different base models on our classification tasks.

For our zero-shot learning experiments, meanwhile, we make use of GPT-3, 3.5 and 4, as provided through the `OpenAI API`. We make use of these because, at the time of writing, they are widely regarded as amongst the state of the art in terms of LLMs and thus likely to offer the best performance for the task at hand. As they are available through an API, they can also be used by anyone with an internet connection, hence size and computational costs are much less of a consideration (we will comment on the actual cost of accessing the API below).

Hyperparameter selection and training details

We fine-tuned all models selected for transfer learning and prompt engineering for three epochs with the default set of hyperparameters: A learning rate of 1×10^{-3}, a maximum text length of 512 tokens and a batch size of eight. To facilitate reproducibility, we conducted three trials of each experimental configuration using seeds [1, 2, 3], with only the seed varied. We use the transformers library (Wolf et al. 2020) for transfer learning and the OpenPrompt library (Ding et al. 2022) for prompt engineering. Our experiments were performed on a single NVIDIA K80 GPU with 56 GiB memory.

Specific to prompt engineering, the training data is structured as [*text*] *Prompt* [*class*]. As discussed above, finding the best prompt is not trivial and requires trial and error (though previous work shows that prompt choice is not a dominant factor in model performance compared with hyper-parameters such as random seeds, Le Scao and Rush (2021)). To try and mitigate the possibility that our results are based on the specifics of an arbitrary prompt template, we test three prompt variations for each task.

The prompts are varied in pattern and language style while being valid descriptions without changing the semantics, representing real-world user scenarios. For abuse classification, the prompts are, "Is this text abusive?", "Does this text contain abuse?", and "It was? Abusive or Not Abusive." For movie sentiment classification task, the prompts are, "Is this text negative?," "Does this text contain negative sentiment?", and "It was? Negative or Not Negative." Having a variety of prompts allows us to show how performance might vary between different reasonable prompts.

OpenAI's API also has a number of hyperparameters that may affect the output provided by the model (OpenAI 2020). For example, previous work has evaluated generated text on varying levels of temperature, top p, and max tokens (Bommarito II and Katz 2022; Rehana et al. 2023). Similar to Rehana et al. (2023), we set temperature, which affects how deterministic the output text is, to 0.1. While temperature is above zero the results will not be purely deterministic, however performance has been shown to improve empirically. While each of our prompts attempts to elicit a single-word response, we set the maximum number of returned tokens to 20 in order to examine why non-response occurs within the model. We additionally provide a number of stop tokens, such as a period and a comma, to limit as much as possible the model's tendency to provide more generated text beyond the single-word response we ask of it.

Results

In this section, we present the results of our experiments, to illustrate these techniques in action. Although our experiments made use of multiple models and configurations, in order to simplify presentation we display just a sample of these. In particular, for transfer learning we show results generated using `DistilBERT`, and for prompt engineering we also look at `DistilBERT` and look at only one fixed prompt ("Does this text contain abuse?" for abuse classification and "Does this text contain negative sentiment?" for movie sentiment classification). For Naïve Bayes, logistic regression and weak supervision, we present the full results. Other base models and prompts provided similar results.

We begin by investigating our binary abuse classification task, shown in Figure 1. This figure has several panels, reflecting whether the underlying data is balanced or not. We can see that in the simple case of underlying balanced data (Figure 1, Panel (a)), which will hence generate both balanced training datasets and test datasets), there is actually little difference between any of the techniques in question. The "conventional"

machine learning approaches achieve a high performance (Macro F1 of around 0.8) with only around 150 examples. While it is eventually outperformed by prompt engineering, transfer learning and weak supervision, this is only after the labeling budget crosses the 250 mark,[7] and even after 1,000 examples have been used the difference is not enormous. Zero-shot learning with GPT-3.5 and GPT-4 offers an impressive Macro F1 (around 0.9) without any training data.

However, we know that the prevalence of abusive text is generally lower than the presence of non-abusive text. Hence a more "realistic" scenario is that both training and test data are unbalanced (Figure 1, Panel (b)). Here we can see that conventional machine learning performs worse than its "cheaper" counterparts, with a Macro F1 score of less than 0.5 for all levels of the training budget investigated for the Naïve Bayes classifier, and with high standard deviation and a lower Macro F1 score than the rest for the case of the logistic regression. In the unbalanced case, where abuse is around 10% of the total dataset, then even with a relatively large labeling budget only a small amount of cases from the positive class will be seen. The difference between the other techniques in particular is notable, with transfer learning able to achieve the highest overall performance (though the Macro F1 of 0.8 from GPT-4 is again notable, considering that no training data is required to achieve it).

Of course, in the case of unbalanced data, one common workaround is to artificially balance the training dataset (though of course preserving the real underlying distributions of data in the test data). This can improve model performance on a limited training budget (though achieving training balanced

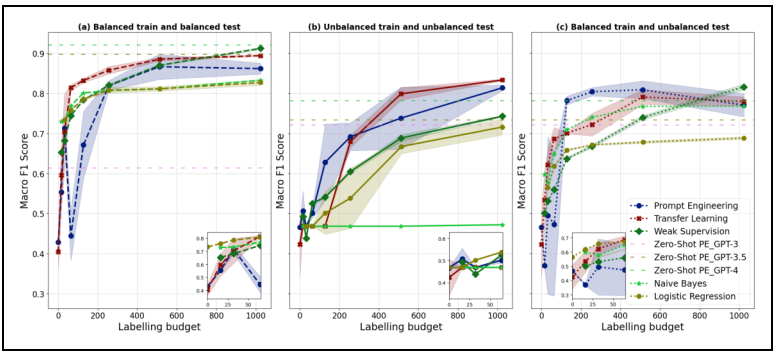

Figure 1. Learning curves of binary abuse classification task.

data still requires a higher labeling effort, or trying to artificially inflate the likelihood of labeling positive data through the use of keywords). Results from this approach are presented in Figure 1, Panel (c). We can see that model performance is improved in all cases, reaching the range of 0.7-0.8 after a few hundred examples (the zero-shot approaches all fall within this range as well), except for the logistic regression results. Weak supervision is actually the best-performing technique in this scenario (as the test data is the same, the results for zero-shot prompt engineering are the same in Panels (b) and (c)).

We will now move on to address our binary movie sentiment classification task, shown in Figure 2. The format of this figure is the same, with panels addressing classifier performance over a variety of labeling budgets for balanced data, unbalanced data, and a balanced training set with unbalanced test data. Again, these three setups were chosen as representative of common real-world research scenarios. The overall results from this task are similar to the abuse classification task. For balanced data, we can see that all of the methods perform in a similar fashion. Our conventional Naïve Bayes approach achieves a Macro F1 below 0.7 with around 100 labeled examples. Prompt engineering, transfer learning and weak supervision are able to outperform it, especially as the labeling budget increases, eventually arriving at scores of around 0.9. However, in this case the logistic regression achieves as good results as transfer learning. Again, zero-shot learning performs well, with performance above 0.9 for GPT-4. In the unbalanced case, the performance of all models is lower, though transfer learning and prompt engineering can achieve a Macro F1 of 0.8 with around 1,000 examples, again common

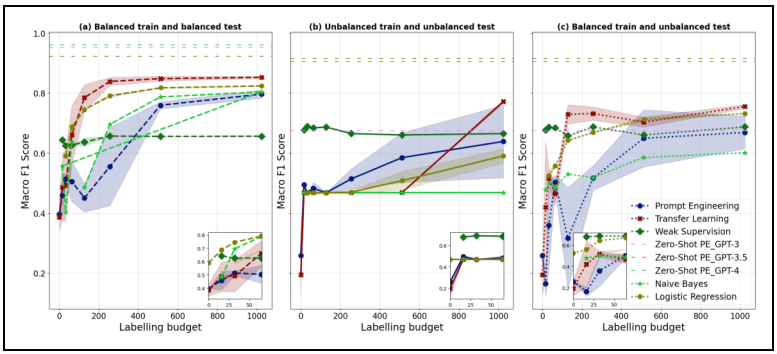

Figure 2. Learning curves of binary movie sentiment classification task.

with GPT-4. Our Naïve Bayes model performs very poorly in this context. Finally, when training data is balanced but test data remains unbalanced, more of the approaches can achieve a Macro F1 of around 0.8, though we would recall again that the cost of generating a balanced training dataset from unbalanced data can be quite high.

The IMDB movie sentiment database (Maas et al. 2011) has been, as mentioned in the Methods Section "Data", extensively studied in the literature. As such, a natural worry arises that the database has been used in the training datasets of GPT-3.5 and GPT-4, thus contaminating the zero-shot testing described above. Given that we are unable to check if indeed the database was used to train the GPT models, we collected an analogous validation of movie reviews from another review aggregator, TMDB.[8] The included reviews were publicly published from October 2021, past the training date cut-off for the GPT-3.5. As in Maas et al. (2011), we label as positive those reviews with a score higher than 6 out of 10, and we label negative those reviews leaving a score lower than 5, omitting neutral reviews and including no more than 30 reviews from a single movie. Our final database comprises 855 reviews, with a distribution of 73.3% of positive reviews and 26.7% of negative reviews. When performing a zero-shot test on the TMDB dataset, we obtain Macro F1 scores between 0.87 and 0.94 for both GPT-3.5 and GPT-4, using the three prompts as the other tasks. GPT-3 has, on the other hand, different performances depending on the prompt, with Macro F1 scores of 0.67, 0.85 and 0.11.[9] The results are in accordance with the results presented in Figure 2, thus validating them for our use of the three versions of GPT. In other words, we are confident the high performance of GPT in these classification tasks is not solely a result of GPT's training data already including the IMDB dataset.

The bias analysis performed for both tasks and all the techniques here is inspired by the work of Ashwin et al. (2023), where the authors examine the biases that open and closed LLMs have when annotating refugees' interviews. We compute the over-prediction of each class in our studied tasks. For the non-zero-shot prompt engineering techniques, we compute the over-prediction at the highest labeling budget ($n = 1,024$). Given the good results of the zero-shot prompt engineering shown in Figures 1 and 2, we compare the biases of the different techniques at their best performances to give the reader an extra dimension with which to compare the studied techniques other than their performance scores. Results are shown in Figures 3 and 4.

While the over-prediction percentages vary between each task—particularly for the zero-shot PE with GPT-3.0—results remain qualitatively similar

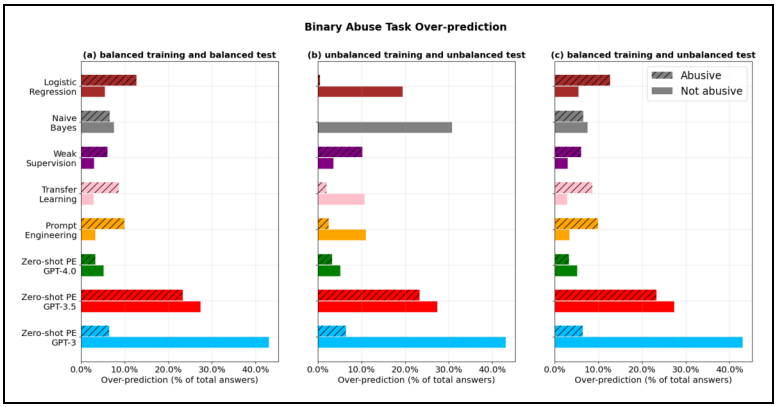

Figure 3. Over-prediction bar plots for every technique used when performing the binary abuse task. For non-zero shot techniques, we use the highest labeling budget $n = 1,024$.

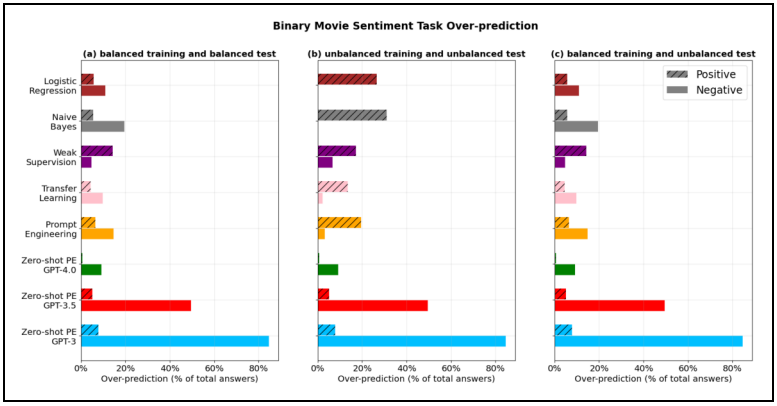

Figure 4. Over-prediction bar plots for every technique used when performing the binary movie sentiment task. For non-zero shot techniques, we use the highest labeling budget $n = 1,024$.

allowing us to draw general conclusions. The GPT-3 technique has the highest over-prediction for both tasks, not detecting abusive language (binary abuse task) or positive language (binary movie sentiment task),

while having a relatively low percentage (<10%) of over-prediction for the opposite classes. The highest over-prediction for both classes is obtained by GPT-3.5 (ChatGPT) in the case of the binary abuse task, while presenting a similar behaviour as GPT-3 for the binary movie sentiment. Finally, GPT-4.0 presents the lowest over-predictions for both tasks. This is interesting because, while the F1-macro scores between GPT-3.5 and GPT-4.0 are relatively similar in Figure 1 and 2, the over-prediction analysis gives completely different behaviours for both models.

The classical machine learning techniques (Naïve Bayes and logistic regression) over-predict one class when there is an unbalanced training dataset, making them sensitive to class unbalance. Finally, the Weak Supervision, Transfer Learning and Prompt Engineering techniques show similar behaviours, with over-predictions lower than 20% for both cases and an inversion of the most over-predicted class for balanced and unbalanced training datasets.

It is also worth evaluating the amount of training time required for each of the techniques, something which has knock-on consequences for the financial costs involved if making use of a cloud platform. Overall, the training time is short for all three techniques and two datasets (see Figures 5 and 6). As the labeling budget grows, the training time also grows linearly except for weak supervision. For weak supervision, increasing labeling examples does not seem to affect training time when trained on Wikipedia personal attacks dataset. The training time is generally around

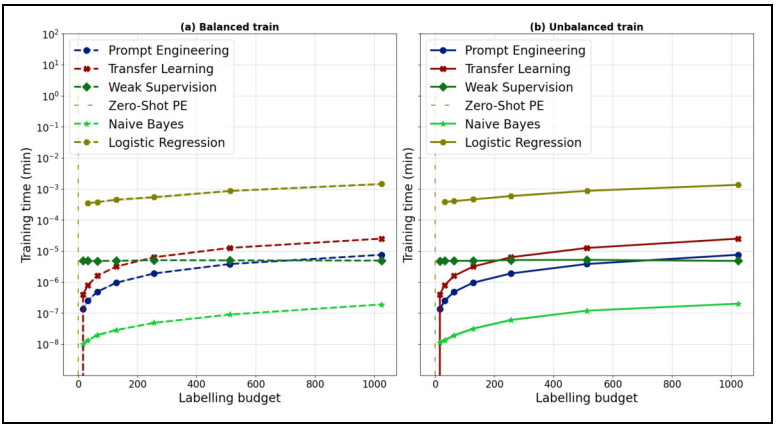

Figure 5. Training time of binary abuse classification task.

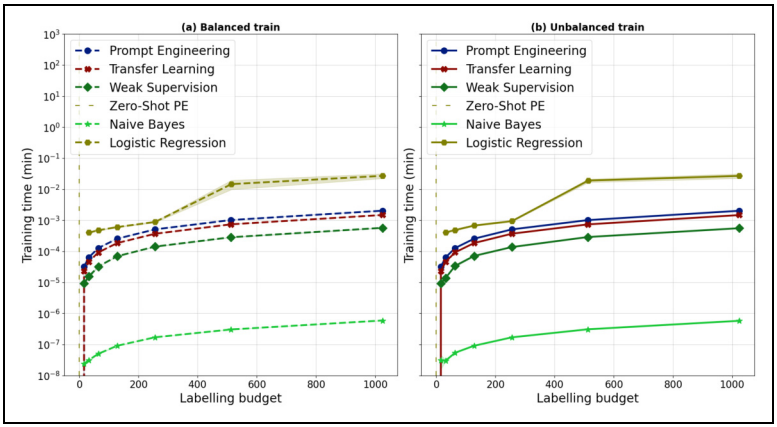

Figure 6. Training time of binary movie sentiment classification task.

10^{-5} minute regardless of labeling budgets and slightly decreases when using the maximum labeling budget in our experiments (n=1,024). Training times in this case are a sum of two elements: The time that the set of LFs takes to label data points and the time that the probabilistic `LabelModel` (Ratner et al. 2019) takes to compute the weights and probabilities associated to the LFs. The different behaviours seen in Figure 5 and Figure 6 depend on the nature of the LFs used. In the case of the binary movie sentiment analysis, we add two `regex` LFs, thus observing the linear increase of time as they require more time to look for matches as the number of data points increases. In the case of the binary abuse experiment, the LFs used are dictionary-based or transforming annotators' knowledge into a LF. Given that directory-based functions (looking for words in a text) are not as computationally expensive as regular expression functions, given that the maximum length of the text is fixed, we obtain a stable training time of 1 minute.

It is also important to note how the conventional machine learning methods bound the training times of all the others. While Naïve Bayes training time is the lowest of the bunch, the logistic regression is the highest, with a difference between the two of almost 5 orders of magnitude for both tasks. However, the training times for all techniques are still relatively low, below 0.1 minutes.

While Zero-shot Prompt Engineering does not require any training time, it is worth briefly commenting on the costs that could be incurred through the

Table 3. Results of zero-shot testing the GPT-3, 3.5 and 4 models on the collected TMDB dataset to label positive and negative movie reviews.

Model	Number of points	Prompt	Macro F1 score
GPT-3	855	Using one word, does the movie review contain negative sentiment, Yes or No?	0.67
GPT-3	855	Using one word, classify the sentiment of the movie review using 'Positive' or 'Negative'.	0.85
GPT-3	854	You are a researcher who needs to classify movie reviews as containing negative sentiment or not containing negative sentiment. Using one word, does the movie review contain negative sentiment, Yes or No?	0.11
GPT-3.5	852	Using one word, does the movie review contain negative sentiment, Yes or No?	0.92
GPT-3.5	843	Using one word, classify the sentiment of the movie review using 'Positive' or 'Negative'.	0.90
GPT-3.5	851	You are a researcher who needs to classify movie reviews as containing negative sentiment or not containing negative sentiment. Using one word, does the movie review contain negative sentiment, Yes or No?	0.90
GPT-4	855	Using one word, does the movie review contain negative sentiment, Yes or No?	0.93
GPT-4	855	Using one word, classify the sentiment of the movie review using 'Positive' or 'Negative'.	0.94
GPT-4	855	You are a researcher who needs to classify movie reviews as containing negative sentiment or not containing negative sentiment. Using one word, does the movie review contain negative sentiment, Yes or No?	0.88

API to perform classification tasks (these are correct at the time of writing). Costs through such an API are per "token": A small piece of text that might be equivalent to a word, a part of a word or a piece of punctuation (as a rule of thumb, a token is often around four characters long). To perform a classification using an LLM there will be costs for both input tokens (the chunk of text to be classified) and output tokens (the response indicating the classification). Of course, the response is usually just one or two tokens long, meaning the length of the input text is the main factor

determining API costs. At the time of writing, GPT-3 had a cost of 0.0006 per 1,000 tokens, GPT-3.5 a cost of 0.0013 per 1,000 tokens and GPT-4 a cost of 0.0025 per 1,000 tokens. The price for a project will, obviously, vary according to the model chosen and the type of input text but, to give an example, the movie reviews were on average around 325 tokens each, meaning that 10,000 movie reviews could be classified on GPT-4 for around 8. Hence, API costs are only likely to be a significant concern for relatively large pieces of text where there are millions of observations to be classified. In the case in which we would like to train the newer OpenAI models (GPT-3.5, GPT-4.0) just as we did with GPT-2.0 for both datasets, we would also need to consider the API costs of the training data sets into account. However, given the good results that the zero-shot learning experiments give, we did not consider these scenarios which would represent an extra cost and thus would go against the spirit of this study.

Conclusions

In this paper, we have sought to give an overview of three methods for automatic content analysis that are "cheaper" than the conventional approach to machine learning, in the sense that they seek to limit the amount of labeled data that is required as a basis for training a model. In this final section, we will draw some conclusions about the significance of our experimental results, and future directions for automatic content analysis in the field.

First, it was striking how all of the techniques we reviewed achieved strong performance in most of the tasks we reviewed, on small labeling budgets. In most cases, a few hundred data points were enough to achieve a Macro F1 score of above 0.7. It should be noted however that the conventional baseline we addressed also performed well for many tasks on similar labeling budgets, provided the data were balanced (though we should emphasise that unbalanced data scenarios are very common in automatic content analysis).

Second, perhaps most of all, it was striking how zero-shot prompt engineering performed well across all of the tasks we reviewed, comfortably outperforming any of the other methods on the movie review task and offering equivalent performance on the abuse classification task. However, even if their performances are high, over-predictions and thus systematic biases can also be present. An analysis of these elements are needed no matter the technique used to avoid arriving at incorrect conclusions. Considering these techniques require no labeled training data (though still require test

data) and are very easy to implement, they appear to offer a very promising and flexible solution to automatic content analysis in the future. Their performance and biases on a wider variety of social science concepts need to be tested; and it should also be borne in mind that there is a financial cost to using them (though not one that will be significant unless millions of lengthy text documents are being labeled). Furthermore, larger models than the current generation of LLMs are already in development, and it is reasonable to expect both performance gains and price decreases as they emerge (in all of our experiments, the increase in size from GPT-3 to GPT-4 corresponded with an increase in performance and a decrease in over-prediction). As this next generation emerges, it may be that other methods of automatic content analysis will start to become redundant, though this proposition would need to be rigorously tested. This proposition would be of particular interest as open- and closed-source LLMs can present serious biases that could skew their uses (Ashwin et al. 2023; Spirling 2023).

Lastly, despite the impressive performance of zero-shot prompt engineering using LLMs, it is important to assess LLMs across various types of tasks to understand their capacity and limitations. For instance, our results show that model performance and over-predictions can vary across seeds or the prompts tested. This instability further highlights the importance of robustness and reproducibility in LLMs. Furthermore, LLMs can be biased towards certain demographics or identities when, for example, asked to perform curriculum screening (Dastin 2022; Veldanda et al. 2023), or annotating refugees' interviews (Ashwin et al. 2023). In this paper, we solely consider binary classification tasks as our intention is to keep results as relatable as possible so that the reader can understand the concepts and bring this knowledge to their classification tasks, as complex as they could be. While an open question remains on how LLMs perform on more complicated tasks that involve language understanding such as logical reasoning and inference, we contribute by giving social scientists different strategies to tackle challenges where data scarcity and bias might be present like analysing interview data from under-represented groups (Ashwin et al. 2023; Atari et al. 2023; Bonikowski and Nelson 2022; Deterding and Waters 2021; Small and Calarco 2022) or text analysis and labeling of historical archives (Jaillant 2022; Marciano et al. 2018; Rahal et al. 2024).

Declaration of Conflicting Interests

The authors declared no potential conflicts of interest with respect to the research, authorship, and publication of this article.

Funding

The authors received the following financial support for the research, authorship, and/or publication of this article: This work was supported by the Public Policy Programme of The Alan Turing Institute under the EPSRC grant EP/N510129/1. This work was supported by the Ecosystem Leadership Award under the EPSRC Grant EP/X03870X/1 and The Alan Turing Institute. Funded by the UK Government.

ORCID iDs

Leonardo Castro-González https://orcid.org/0000-0001-5631-0137
Yi-Ling Chung https://orcid.org/0000-0003-2076-0720
John Francis https://orcid.org/0000-0003-1405-4408
Pica Johansson https://orcid.org/0009-0008-9850-5686

Data Availability Statement

Our code and experimental results are available here at https://github.com/Turing-Online-Safety-Codebase/cheap_learning, allowing users to understand, verify and replicate findings.

Supplemental Material

Supplemental materials and Appendices for this article are available online.

Notes

1. https://github.com/Turing-Online-Safety-Codebase/cheap_learning
2. Precision refers to the number of true positives divided by the total amount of the positive elements retrieved. In other words, precision measures the reliability of the results. On the other hand, recall refers to the ratio of the true positives divided by the total amount of relevant items (true positives + false negatives).
3. See for instance: https://huggingface.co/docs/transformers/model_doc/t5
4. Statements on the size of models that a typical laptop can run will go out of date quickly, however at the time of writing Llama 2-70b, arguably the best open source LLM, required about 320 GB of RAM to run (NVIDIA 2022), which is beyond the realistic range of a typical laptop. Running quantized versions of Llama is undoubtedly possible, though quantized versions may offer lower performance as well as still being quite slow to run.
5. We use the package developed by scikit-learn: https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html