Does GPT-3 know what the Most Important Issue is? Using Large Language Models to Code Open-Text Social Survey Responses At Scale*

Jonathan Mellon[1], Jack Bailey[2], Ralph Scott[2,3], James Breckwoldt[2], and Marta Miori[2]

[1]West Point
[2]University of Manchester
[3]Cardiff University

This version: January 06, 2023

Abstract

We examine the use of large language models (LLMs) like OpenAI's GPT-3 for coding open-ended survey responses. We compare GPT-3's performance on a classification task using data from the British Election Study Internet Panel (BESIP) with that of a human coder and an SVM (a traditional supervised learning algorithm) fitted to 576,325 manually labelled observations. We find that while GPT-3's 3-shot performance is slightly lower than a second human coder (97% agreement), GPT-3 is able to match the original human coder's collapsed category 95% of the time, and outperforms the SVM in terms of accuracy and bias. The results suggest that LLMs perform acceptably when coding this type of open-ended survey response and may allow for greater use of open-ended questions in future.

*The views expressed herein are those of the authors and do not reflect the position of the United States Military Academy, the Department of the Army, or the Department of Defense. Note: This is a working paper. The results and arguments may change before publication as additional analysis is conducted. Comments are welcome: jonathan.mellon@westpoint.edu

1 Introduction

Open-text survey responses offer certain advantages over their closed counterparts. For instance, they avoid priming respondents to give particular answers (Ferrario and Stantcheva 2022; Esses and Maio 2002). Likewise, they do not prejudge the answers that survey respondents might give (Geer 1991; Schuldt and Roh 2014). Despite these advantages, they have one considerable drawback: analysing these data often means first coding them into closed categories,[1] which can be costly and time-consuming for researchers.

It would be better to have a computer algorithm code these open text responses instead. However, many worry, with good reason, whether algorithms can produce results suitable for analysis (He and Schonlau 2020). The capabilities of artificial intelligence tools have skyrocketed in recent years (Kaplan et al. 2020).[2] Large language models (LLMs) like OpenAI's GPT-3 can now perform novel tasks involving human-readable text (Brown et al. 2020). This raises an interesting question: are these tools now able to code open text well enough to use in public opinion research?

In this paper, we consider the task of coding open-text data from the British Election Study Internet Panel (BESIP). BESIP respondents answered the question "As far as you're concerned, what is the SINGLE MOST important issue facing the country at the present time?". The most important issue (MII) question is a key indicator of issue salience (Dennison 2019; Bevan, Jennings, and Wlezien 2016), although there are ongoing debates about how the responses should be interpreted (Bartle and Laycock 2012; Jennings and Wlezien 2011, 2015). Previously, MII responses in BESIP have been labelled by human coders who assign responses to 50 granular and 13 collapsed

[1] It is also possible to use structural topic models to categorise open-text data (Roberts et al. 2014). But doing so can have some undesirable properties in this case. For instance, it can produce categories that do not reflect the theory of concern and separate substantively similar but linguistically distinct topics. Likewise, it can also create issues for multi-wave surveys where the topics that emerge might differ wave-by-wave. As such, most analyses of these types use pre-defined closed categories instead.

[2] We used chatGPT to copy-edit this article and draft the abstract when given the text of the introduction.

categories, a process that is expensive and time-consuming. Research assistants at BESIP have manually labelled over 657,000 open-text responses to the MII question since 2014.

We compare GPT-3's performance on this classification task against a second human coder and a more traditional supervised learning algorithm, the support vector machine (SVM). GPT-3 was only given a description of the task and categories, but no training examples. Its performance on the classification task was slightly lower than a human's. GPT-3 coded responses into the same collapsed category as the original human coder 95% of the time, compared to 97% for the second human coder. Despite having no dedicated training data, GPT-3 performed equal to or better than an SVM fitted to 576,325 manually labelled observations in terms of accuracy and bias. GPT-3's performance was particularly strong when looking at lower frequency responses. GPT-3 is able to match the original human coder's collapsed category 86% of the time: lower than the human performance of 92% but far exceeding the 74% for the large SVM. This suggests that GPT-3 will be more adaptable than supervised learning methods if the issue agenda changes in the future.

Our goal is to compare the performance a social scientist can expect using supervised learning and current LLMs. Thus, we use these methods in a straightforward manner. In other words, we do not engage in customisation, feature engineering, or fine-tuning. Both supervised learning and LLMs would likely perform better using these tweaks, but this would involve a significant time investment and considerable technical expertise, which reduces the applicability of the results for practitioners. Anyone with intermediate R skills can implement the approach we use in this paper.

Our results suggest that at least some open-ended survey questions can be coded by LLMs with acceptable levels of bias and accuracy, and reasonable confidence that performance will not greatly decrease with distributional changes. If LLMs can perform this task as well as humans, it could save significant time and money and

enable the use of open-ended survey questions to gather information from respondents on topics such as party images, leader images, salient social identities, issue ownership, and media consumption.

2 Current use of machine learning in social science

It is now common for social and political scientists to use machine learning for prediction, estimation, and classification (Grimmer, Roberts, and Stewart 2021). Many existing studies use supervised machine learning techniques (Russell and Norvig 2022, chap. 19). In these techniques, a model is trained on a set of text data and manually-assigned labels. Once trained, the model can then assign labels to any new text that it receives for use in later analysis (Grimmer and Stewart 2013; Goldberg 2017; Gentzkow, Kelly, and Taddy 2019; Kadhim 2019). Automatic text classification using supervised machine learning tends to achieve a high level of accuracy for some tasks and has been used to, for example, code policy areas of news articles (Scharkow 2013; Burscher, Vliegenthart, and De Vreese 2015) and legislative bills (Hillard, Purpura, and Wilkerson 2008).

Our study differs from previous studies in that we use OpenAI's GPT-3. GPT-3 is a large language model (LLM) trained to predict the next most likely word that will occur after a user-input prompt. As an example, an LLM would be likely to predict that the most probable word after the prompt:

```
Roses are red, violets are
```

would be the word "blue" because that is a common continuation within its training data.[3] However, LLMs that use deep learning methods are now able to complete novel tasks by employing multiple levels of abstraction to determine the most pertinent aspects of natural language in context and predict an appropriate response (LeCun, Bengio, and Hinton 2015).

[3]chatGPT completes the next word as blue and then adds another 9 lines of rhyming about other flowers.

For instance, GPT-3 (accessed via chatGPT) can complete the prompt:

```
Write a 3 line poem about Margaret Atwood eating a canary
```

With a novel response that never appeared in its training data:[4]

```
Margaret Atwood dines,
With a canary on her plate,
A fleeting, feathered feast.
```

The performance of LLMs on a variety of tasks appears to scale in a power-law relationship with the number of parameters (model complexity), dataset size, and computational time spent in unsupervised training (Kaplan et al. 2020), suggesting that performance will continue to improve as computational capacity increases.

LLMs can generate text on domains or tasks not explicitly covered in their training data. This has allowed some social scientists to use LLMs for classification tasks. Hu et al. (2022) use a domain-specific LLM for conflict coding tasks. Ornstein, Blasingame, and Truscott (2022) use off-the-shelf LLMs—GPT-3 and Google's BERT—to code the sentiment of tweets, tone in political advertisements, ideology coding in manifestos, and virtues mentioned in congressional speeches. While performance varied, they concluded that GPT-3 could sometimes outperform non-expert coders but did not reach expert performance. Argyle et al. (2022) even suggest using LLMs to simulate human survey respondents.

We contribute to this growing literature by benchmarking the performance of off-the-shelf LLMs on a very common use case in public opinion research: coding open-text survey responses into analytic categories. We compare this performance to two relevant alternatives: human coding and supervised learning (using SVMs).

[4]Literary merit aside, the phrase "fleeting, feathered feast" appears to be original to this poem, so GPT-3 appears to be able to reason directly about the structure of poems.

3 Methods

We aim to compare how well LLMs can code responses to a standard most important issue (MII) question embedded in all 23 waves of the British Election Study Internet Panel (Fieldhouse et al. 2022). We explain our evaluation strategy, describe how we code MII responses using GPT-3, and describe the details of the supervised learning models and human coding we used.

3.1 Evaluation strategy

An algorithm can be considered successful in coding open-text responses if it can closely replicate the labels that a human coder would have given. The case we consider is the 81,266 open text responses given to the survey question "As far as you're concerned, what is the SINGLE MOST important issue facing the country at the present time?" across waves 21-23 of the British Election Study Internet Panel (Fieldhouse et al. 2022). These open text responses have already been hand-coded by a research assistant, providing a "correct" answer to compare predicted labels against. We test the coding approaches on two samples of 1,000 open-text responses.

There are two relevant standards to assess the accuracy of algorithms. The first is the percentage of open-text responses that the algorithm codes correctly. This is a good indicator of the accuracy that users of the coded data can expect to deal with.

However, open-text responses are often repeated, so this figure does not necessarily capture how well the algorithm will deal with the full range of possible responses. For example, 10,291 respondents gave the exact answer "Covid" across waves 21-23 of the BESIP. To address this, we also created a dataset of 1,000 randomly sampled unique open-text responses from the BESIP. We expect this dataset to be a harder test because it will overrepresent rare and idiosyncratic responses.

While we expect that human coders will provide the most accurate coding, even humans following clear coding rules will not agree on a label 100% of the time.

The relevant benchmark to evaluate an AI against is the level of accuracy that an independent human coder would achieve on these same responses. We therefore had another human coder (also a co-author on this paper) code both sets of 1,000 open text responses for comparison. That coder (a public opinion specialist) received an hour of training with examples drawn from a separate sample to the test data, and feedback from an experienced coder.

Finally, we code the open-text responses using a supervised learning algorithm: support vector machine (SVM) to see whether LLMs are able to approach or surpass the performance of other machine learning methods. We stem the words and remove numbers but otherwise use the default settings in the RTextTools package (removing stop words, white space, and punctuation, and converting text to lowercase).

We fit one SVM to 1,000 randomly sampled responses from waves 21-23 of BESIP. This represents one common strategy where a researcher labels a moderate sample of training data, then uses a supervised learning algorithm to label a much larger dataset. We also fit a second SVM to 576,325 responses from waves 1-20 of BESIP. This represents the use case of fitting a model to a large collection of existing coding and applying it forwards, an approach that will often not be feasible in real-life usage, but which therefore provides a strong test of the relative merits of LLMs when compared with optimally trained SVMs.

3.2 Constructing the LLM Prompt

OpenAI's GPT-3, or more specifically *text-davinci-003* sometimes referred to as GPT-3.5, which was popularized through the ChatGPT interface released on 30 November 2022, represents the public state-of-the-art LLM (Open AI 2022). This model has been fine-tuned (starting from the original GPT-3) through a three-step process of supervision: first, a human labeler provides an example of the desired outcome for a randomly selected prompt; second, a reward model is trained by asking the model to generate several outputs which are then ranked by a human labeler; finally,

new outputs are evaluated by this reward model through a process of reinforcement learning by proximal policy optimization (Ouyang et al. 2022). In contrast to SVM approaches, these models require only very few in-context examples to achieve a high level of accuracy, with GPT-3 even achieving reasonable performance with three examples being provided (known as "few-shot") (Brown et al. 2020).

To construct an appropriate prompt for GPT-3, we first experimented with chatGPT to understand the capabilities and limits of the engine. At the time of writing, chatGPT is available for free to consumers, so represents a cheap way to refine the prompt without incurring API fees. The final prompt we used is shown in appendix 7.1. The prompt starts by saying:[5]

```
Here are some open-ended responses from the British Election Study to the
    question "what is the most important issue facing the country?". Please
     assign one of the following categories to each open ended text
    response, returning the original response and the most relevant numeric
     code.
```

We then listed the possible categories. In chatGPT, we found that conversationally correcting the AI on its approach worked well, but that we could pre-empt these errors by adding a small amount of additional detail to the category names. For instance:

```
pol-neg: complaints about politics, the system, the media, corruption where
     no politician or party is mentioned
europe: including Brexit
```

We additionally informed the AI about Russia's invasion of Ukraine which took place after GPT-3's training data:

```
For context, Russia invaded Ukraine prior to the fieldwork for this survey,
     so references to Ukraine or Russia are about war.
```

[5]We inadvertently left the reference to a numeric code in the prompt (our earlier testing had included category codes), but this does not appear to have affected the performance. Future iterations of this paper will rerun the prompt with the corrected wording.

And reminded the AI to only use a single label for each response:

```
If multiple issues are mentioned, use the label for the first issue
    mentioned.
```

This is because our testing showed that the AI often assigned multiple labels to each response.

We then showed a short example of the response format and gave another nudge to return only a single code:

```
Code these cases:
a bad economy
immergration

a bad economy|economy-general
immergration|immigration

Code these cases:
climate change and unemployment

climate change and unemployment|environment,unemployment

Please only return one code per response (if multiple match use the first
    issue listed). The correct response should be:

climate change and unemployment|environment
```

We then finished by saying:

```
Code these cases:
```

and listed 50 open-ended responses at a time.

Since we provide only 3 examples, this is a few-shot learning task where the algorithm relies on the task description and its background training rather than

learning from examples. GPT-3 was capable of completing the task at a similar quality level with zero examples, but used inconsistent response formats which made the coding harder to process.

3.3 API

For the actual coding, we used the Open AI's API to make requests to *text-davinci-003*—a closely related model to that underlying chatGPT. The code for the API calls is shown in appendix 7.2. GPT-3 was generally good about using only the categories given, but we had to make a small number of manual edits to its labels. For instance, it used "covid" rather than "coronavirus" as a label in some cases and corrected the capitalization of the "europe" category to "Europe". Despite repeated instructions to return only a single code, GPT-3 still sometimes returned multiple codes. We used only the first code returned in these cases.

3.4 Prompt options that were considered and rejected

Because the open-ended responses vary in length, we chose to conservatively process only 50 responses at a time to avoid hitting token limits in the prompt and response. More efficiency would be possible by varying the batch sizes to maximise the number of responses per batch, but we decided against this since the processing costs and runtime were still low.

We also tried two alternate response formats that reduced the size of the text response (and therefore cost). First, we included numeric codes for each response and asked GPT-3 to return the row number and label:

```
1|Russian war
2|After effects of covid
3|covid apathy
4|The war in ukrain
```

So the response would be:

```
1|war
2|coronavirus
3|coronavirus
4|war
```

We also experimented with an even terser response format where each label was given a numeric code and GPT-3 was asked to respond with the row number and numeric code:

```
1|22
2|48
3|48
4|22
```

GPT-3 was able to perform both of these tasks despite their increased complexity, but there did appear to be a small performance penalty in our initial testing, so we opted for the simpler approach of returning the full text response and label. Further experiments with the prompt included providing worked examples of previously incorrect codings and corrections to these, but this did not appear to reliably improve performance (this matches the experiences of other researchers with GPT-3 (Papay, Waterbury, and Kaplan 2022)). We also experimented with providing the full 3,500 word coding guide to the AI. However, giving this much text seemed to distract GPT-3 from reliably following the main prompt. Researchers should continue to experiment with variations such as these, as their performance may be task dependent and future algorithms may handle them without penalty.

Finally, we also tried using another cheaper and faster model from Open AI *text-curie-001* but this was not able to reliably complete the task as requested. This suggests that easily completing this coding task is a capability only of the latest LLMs.

4 Results

4.1 Coding Accuracy

Figure 1 shows the accuracy levels of GPT-3, the two SVM approaches, and the new human coder compared to the original human coding. The first panel shows agreement in terms of the full 50 MII categories and the second panel shows agreement when the labels are collapsed into 13 analytic categories.

The human coder consistently and significantly outperforms all of the algorithms, but the performance edge for the human varies substantially across conditions. Fitting an SVM to a modest sample of 1,000 random cases always substantially underperformed the other methods in terms of accuracy in all conditions. As a result, we do not recommend this approach unless a considerably lower level of accuracy is acceptable.

The most relevant test for many use cases is the accuracy of the coding for the collapsed 13-category MII variable in a typical sample. On this measure, the human coder achieved 97% agreement with the original coder. However, GPT-3 was able to achieve 95% agreement. The large SVM fitted to 576,000 previous cases achieved similar performance to GPT-3, with 94% agreement.

The results for the 50-category comparison on the random sample are similar, but all coders show slightly lower levels of agreement (reflecting the fact that many miscodings are between labels that appear in the same collapsed category). GPT-3 and the large SVM achieve similar performance, which remains slightly below that of the human coder.

The fact that GPT-3's 3-shot performance can equal or exceed that of a supervised learning model trained on more than half a million cases is impressive. While the British Election Study Internet Panel features a large set of previously labelled data, making this comparison possible, this is not the case for most other use cases, including other open response items on the BESIP survey that have not been manually labelled

to this extent.

The results for the random sample of unique responses show larger performance gaps. When coding into 13 categories, the human coder achieved 91.9% agreement with the original coder, which is substantially higher than GPT-3's performance of 85.9% agreement. However, the two SVM approaches both showed far lower levels of accuracy than GPT-3 on the random sample of unique responses.

This performance difference makes sense because the supervised models rely much more heavily on matching stems to labels. This means that less common responses will be outside of its training data. Because GPT-3 can bring in an understanding of language trained on a large proportion of the entire internet it can still extract meaning from this text.[6]

These results suggest that LLM-based categorization may have an additional benefit: it may be able to handle new phrases in discourse more effectively than supervised approaches. It is likely that the performance of GPT-3 would remain similar for future coding tasks, but the performance of the SVM model may decrease if there is a longer period of time between the labelled data and the new data that needs to be coded.

4.2 Bias

Accuracy is an important metric for assessing a coding technique, but we care not only about the proportion of categories that are correct, but also about whether the coder makes systematic errors. We assess bias by looking at how closely the algorithm reproduces the proportions of responses in each category.

To measure the aggregate bias in category assignment we use the Pedersen index of aggregate volatility (Pedersen 1979). The index is calculated by summing the absolute difference in the proportion of observations assigned to each category by coders A

[6]More specifically, the training dataset includes a filtered version of the CommonCrawl corpus (consisting of nearly a trillion words, collected over 12 years of web crawling), in addition to some higher quality curated datasets including WebText, English language Wikipedia, and two collections of internet-based books (Brown et al. 2020).

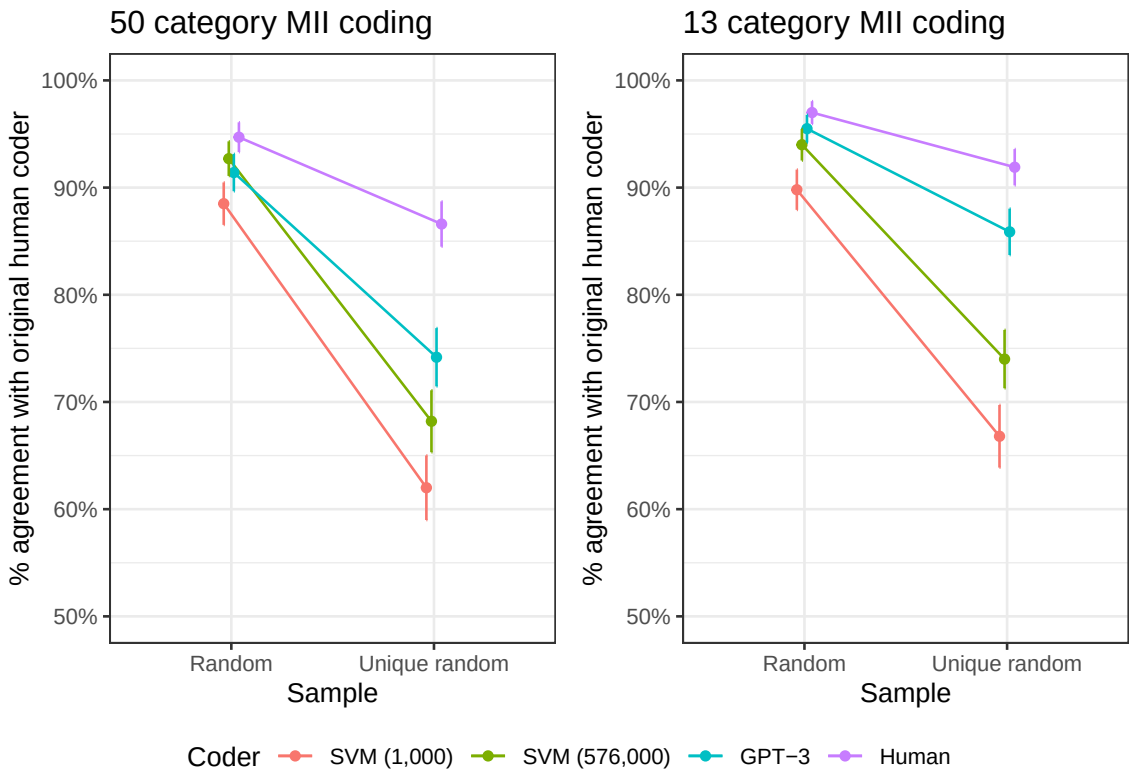

Figure 1: Percentage of responses that agree with original human coder for four coders across the 1,000 random responses and 1,000 random unique responses.

and B and then dividing the total by 2. If A and B assign categories identically the index would be zero and if they assign categories entirely differently (e.g. there is no overlap in the categories they use), the index would be 1.

As an example, suppose that coder A assigns 20% of observations to the category "Dogs" and 80% to the category "Cats" while coder B assigns 50% of observations to each category. We would calculate the aggregate difference between coder A and coder B as:

$$\frac{|0.2 - 0.5| + |0.8 - 0.5|}{2} = \frac{0.6}{2} = 0.3$$

In each case, we calculate the Pedersen index for the categories assigned by a coder compared to the categories assigned by the original human coder. It is important to note that this measure only focuses on the proportion of observations in each category and does not look at whether the same observations are assigned to each category.

Figure 2 shows the error in category proportions using the Pedersen index of aggregate volatility.

The human coder shows the lowest level of bias across all conditions. A chi-squared test cannot reject the null hypothesis that the human coder is unbiased relative to the original coding in any of the four conditions.

GPT-3 shows low levels of bias in the random sample (not statistically distinguishable from unbiased when using the chi-squared test in this sample) when assessed using either 13 or 50 categories. However, the bias is higher in the random sample of unique responses. For the 50 category coding of the unique responses, GPT-3's Pedersen index is 0.14, compared to 0.07 for the human coder. This indicates that GPT-3 makes some systematic errors in its categorizations when disaggregated. However, the Pedersen index for GPT-3 decreases substantially when collapsing the unique-random-sample labels into 13 categories, with a Pedersen index of 0.07, with the human Pedersen index falling to 0.04.

The SVM trained on a random sample of 1,000 responses underperforms greatly in terms of bias compared to the other coders. The SVM trained on all previous MII responses, performs comparably to GPT-3 on the random sample, but greatly underperforms on the random sample of unique responses. This suggests that SVMs will perform reasonably well on new data that has a similar distribution to their training, but may make substantial errors if the underlying distributions change (a problem that GPT-3 avoids because it does not use tailored training data).

4.3 What types of errors did GPT-3 make?

The most common error was for GPT-3 to code a response as "covid-economy" when the original human coder coded it as "coronavirus". Table 1 shows the open-text responses where this combination was observed. In most cases, either interpretation can be reasonably justified depending on how much economic subtext a coder is willing to infer from the text.

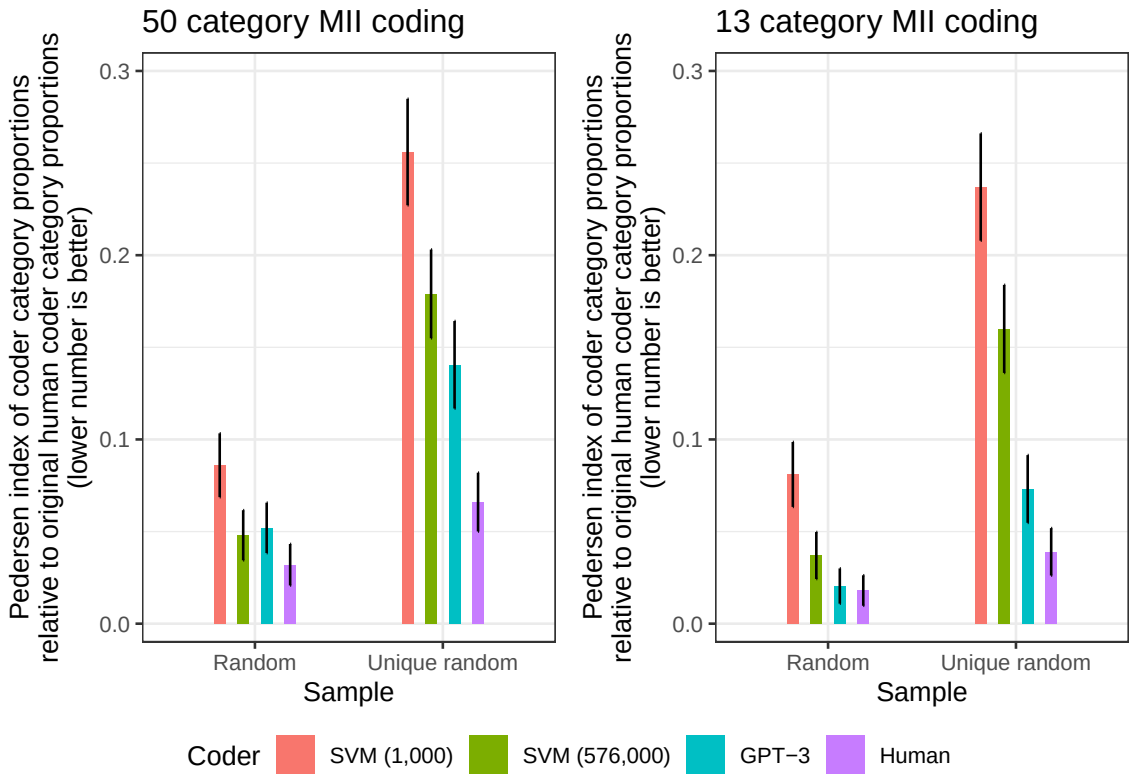

Figure 2: Pedersen index of category assignments for the four coders compared to the original human labelling. Shown in terms of the random sample of 1,000 responses and random sample of 1,000 unique responses, and 13 or 50 categories. 95% confidence intervals calculated using normal bootstrap (1,000 resamples).

Table 1: Open text responses where GPT-3 labelled it covid-economy and original human coder labelled as coronavirus

Text
lockdowns
The control of covid
CLOSING THE BORDERS SO AS NOT TO LET THE VIRUS ESCALATE
Organising recovery after Covid
Covid overreach by the government
government covid restrictions
Post pandemic recovery
Getting the country back to normal.
Recovery from Covid
Getting rid of all covid restrictions
coming out of the pandemic, getting the country moving again
Covid-19 Recovery
End of Covid regulations and start of economic recovery
The after effects of the pandemic
The relaxing of the covid restrictions
Opening up the services sector
End lockdown
Post covid recovery
Reopening after Covid
Living with covid
Covid restrictions being lifted too fast
Post Covid adaptation
Covid-19 restrictions
To get back to normality
Opening up post covid
Rebounding from Covid
Everything opening back up

Table 2: Open text responses where GPT-3 labelled it war and original human coder labelled as foreign affairs

Text
Ensuryng that ?Ukraine does not lose to Adolf Putin The crisis in ukraine Ukraine Crisis Putin's madness Russian hostility
Ukrainian crisis and threat of escalation from Russia russian threat Threat to world peace by Russia Russian aggression. Russia's behaviour
The security situation in eastern Europe (Russia/Ukraine)

Another common category of error was where GPT-3 labelled a text response as "war" while the human coder labelled it as "foreign affairs". Table 2 shows the open text responses with this pattern. The human coder has interpreted the war category relatively narrowly whereas GPT-3 has included cases that do not explicitly mention invasions or war but imply it.[7]

The second most common category of coding differences was where GPT-3 gave the uncoded designation (because the response was too ambiguous to determine) and the human coded the text as "coronavirus". Table 3 shows these responses. In this case, the human is more willing to make a reasonable inference that COVID is being referred to, while GPT-3 conservatively applies the uncoded designation. This may reflect the wording used to describe the uncoded category in the prompt:

```
uncoded: for responses that cannot be reliably assigned to a category
```

In another case, GPT-3 fails to code Omicron as "coronavirus" presumably because the Omicron variant emerged after its training data was collected.

In all three cases, the differences reflect subtle interpretations of the bounds of

[7]After further discussion, the BES team is considering switching the coding of the war category to include responses that imply war without making it explicit. However, we stick to the original schema in this paper.

Table 3: Open text responses where GPT-3 labelled it uncoded and human coder labelled as coronavirus

Text
dumb people who think covid is a pandemic and do no research
Getting back to a form of normality
Pandamia
Covis
Lifting all restrictions
Omicron restrictions
Return of vivid as before
Movie 19
Getting back to normal after the opening up of the country following lockdown
Stopping any idea of a new normal.
Easing restrictions following pandemic
Foreign travel

categories rather than clear errors. It may be possible to further improve performance on these edge cases with more detailed instructions about categories. However, our testing suggested that fixing one problem often created a different problem elsewhere, so it may be difficult for current LLMs to reliably avoid this level of error in few-shot open text coding tasks.

5 Conclusions

Our results suggest that current state of the art LLMs such as GPT-3 are capable of performing open-text survey labelling tasks at close to human-level performance. While there may be some research questions that need the additional accuracy and reduced bias of a human coder (for instance, a research question focused on the nature of COVID discourse might turn on the exact categorization of coronavirus and covid-economy responses), most use cases are likely well served by GPT-3's performance.

How generalizable is the success of GPT-3 in coding open-text survey responses? The task of MII coding in recent waves of the BESIP likely represents a best-

case scenario for open-text coding tasks for social scientists, as the responses are definitionally about issues that are widely discussed in the English language. It is likely that these issues were covered extensively in GPT-3's training data. It is likely that text coding tasks that require more specialised knowledge will show lower performance. For instance, identifying which provisions in a piece of legislation benefit a particular industry might see a larger gap between human and LLM performance.

However, public opinion surveys generally encourage responses that do not reflect deep expertise, precisely because the surveys are designed to be completed by the general public. Public opinion survey coding may be a fruitful application of LLMs because the domains they cover are also likely to be well represented in LLM training data.

We expect our results to generalise to future LLMs (such as GPT-4, which is due to be released in 2023), as the performance on few-shot tasks has rapidly increased as these models develop. We show that, as GPT-3 understands human language prompts, it is possible to teach it to code open-text data in much the same way one might teach a human RA. However, unlike an RA, LLMs like GPT-3 produce results much more quickly and at a fraction of the cost.

Being able to code open-text survey responses into closed categories without investing in manual labelling opens up new research possibilities. Researchers who would prefer to use a different schema for a particular open-ended question can now cheaply run their own classification on an LLM. This will also help researchers make comparative coding of issue agendas, even if the original data collectors did not use the same schema.

Finally, cheap text coding makes the wider use of open-ended survey questions more viable. Scholars have speculated that open-ended questions may be preferable for certain types of questions, but this approach has largely been confined to narrow uses in practice because of cost. When we allow respondents to give their views without the constraints of closed categories, we may find answers we could not have

anticipated.

6 References

Argyle, Lisa P., Ethan C. Busby, Nancy Fulda, Joshua Gubler, Christopher Rytting, and David Wingate. 2022. "Out of One, Many: Using Language Models to Simulate Human Samples." In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 819–62. https://doi.org/10.18653/v1/2022.acl-long.60.

Bartle, John, and Samantha Laycock. 2012. "Telling More Than They Can Know? Does the Most Important Issue Really Reveal What Is Most Important to Voters?" *Electoral Studies* 31 (4): 679–88. https://doi.org/10.1016/j.electstud.2012.07.005.

Bevan, Shaun, Will Jennings, and Christopher Wlezien. 2016. "An Analysis of the Public's Personal, National and EU Issue Priorities." *Journal of European Public Policy* 23 (6): 871–87. https://doi.org/10.1080/13501763.2015.1070191.

Brown, Tom B., Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, et al. 2020. "Language Models Are Few-Shot Learners." arXiv. https://arxiv.org/abs/2005.14165.

Burscher, Bjorn, Rens Vliegenthart, and Claes H. De Vreese. 2015. "Using Supervised Machine Learning to Code Policy Issues: Can Classifiers Generalize Across Contexts?" *The ANNALS of the American Academy of Political and Social Science* 659 (1): 122–31. https://doi.org/10.1177/0002716215569441.

Dennison, James. 2019. "A Review of Public Issue Salience: Concepts, Determinants and Effects on Voting." *Political Studies Review* 17 (4): 436–46. https://doi.org/10.1177/1478929918819264.

Esses, Victoria M., and Gregory R. Maio. 2002. "Expanding the Assessment of Attitude Components and Structure: The Benefits of Open-ended Measures." *European Review of Social Psychology* 12 (1): 71–101. https://doi.org/10.1080/14792772143000021.

Ferrario, Beatrice, and Stefanie Stantcheva. 2022. "Eliciting People's First-Order Concerns: Text Analysis of Open-Ended Survey Questions." *AEA Papers and Proceedings* 112 (May): 163–69. https://doi.org/10.1257/pandp.20221071.

Fieldhouse, Edward, Jane Green, Geoffrey Evans, Jonathan Mellon, Christopher Prosser, Jack Bailey, Cees van der Eijk, Hermann Schmitt, and Rose de Geus. 2022. "British Election Study Internet Panel Waves 1-23." https://doi.org/10.5255/UKDA-SN-8810-1.

Geer, John G. 1991. "Do Open-Ended Questions Measure "Salient" Issues?" *Public Opinion Quarterly* 55 (3): 360. https://doi.org/10.1086/269268.

Gentzkow, Matthew, Bryan Kelly, and Matt Taddy. 2019. "Text as Data." *Journal of Economic Literature* 57 (3): 535–74. https://doi.org/10.1257/jel.20181020.

Goldberg, Yoav. 2017. *Neural Network Methods for Natural Language Processing*. Synthesis Lectures on Human Language Technologies. Cham: Springer International Publishing. https://doi.org/10.1007/978-3-031-02165-7.

Grimmer, Justin, Margaret E. Roberts, and Brandon M. Stewart. 2021. "Machine Learning for Social Science: An Agnostic Approach." *Annual Review of Political Science* 24 (1): 395–419. https://doi.org/10.1146/annurev-polisci-053119-015921.

Grimmer, Justin, and Brandon M. Stewart. 2013. "Text as Data: The Promise and Pitfalls of Automatic Content Analysis Methods for Political Texts." *Political Analysis* 21 (3): 267–97. https://doi.org/10.1093/pan/mps028.

He, Zhoushanyue, and Matthias Schonlau. 2020. "Coding Text Answers to Open-ended Questions: Human Coders and Statistical Learning Algorithms Make Similar Mistakes." *Methods* data (November): 17 Pages. https://doi.org/10.12758/MDA.2020.10.

Hillard, Dustin, Stephen Purpura, and John Wilkerson. 2008. "Computer-Assisted Topic Classification for Mixed-Methods Social Science Research." *Journal of Information Technology & Politics* 4 (4): 31–46. https://doi.org/10.1080/19331680801975367.

Hu, Yibo, MohammadSaleh Hosseini, Erick Skorupa Parolin, Javier Osorio, Latifur Khan, Patrick Brandt, and Vito D'Orazio. 2022. "ConfliBERT: A Pre-trained Language Model for Political Conflict and Violence." In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 5469–82. Seattle, United States: Association for Computational Linguistics. https://doi.org/10.18653/v1/2022.naacl-main.400.

Jennings, Will, and Christopher Wlezien. 2011. "Distinguishing Between Most Important Problems and Issues?" *Public Opinion Quarterly* 75 (3): 545–55. https://doi.org/10.1093/poq/nfr025.

———. 2015. "Preferences, Problems and Representation." *Political Science Research and Methods* 3 (3): 659–81. https://doi.org/10.1017/psrm.2015.3.

Kadhim, Ammar Ismael. 2019. "Survey on Supervised Machine Learning Techniques for Automatic Text Classification." *Artificial Intelligence Review* 52 (1): 273–92. https://doi.org/10.1007/s10462-018-09677-1.

Kaplan, Jared, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. "Scaling Laws for Neural Language Models." arXiv. https://arxiv.org/abs/2001.08361.

LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton. 2015. "Deep Learning." *Nature* 521 (7553): 436–44. https://doi.org/10.1038/nature14539.

Open AI. 2022. "Model Index for Researchers." https://beta.openai.com/docs/model-index-for-researchers.

Ornstein, Joseph T, Elise N Blasingame, and Jake S Truscott. 2022. "How to Train Your Stochastic Parrot: Large Language Models for Political Texts." github.io.

Ouyang, Long, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, et al. 2022. "Training Language Models to Follow Instructions with Human Feedback." arXiv. https://arxiv.org/abs/2203.02155.

Papay, Spencer, Sam Waterbury, and Russell Kaplan. 2022. "How Much Better Is OpenAI's Newest GPT-3 Model? - Scale." *ScaleAI*. https://scale.com/blog/gpt-3-davinci-003-comparison.

Pedersen, Mogens. 1979. "The Dynamics of European Party Systems: Changing Patterns of Electoral Volatility." *European Journal of Political Research* 7 (1): 1–26. https://doi.org/10.1111/j.1475-6765.1979.tb01267.x.

Roberts, Margaret E., Brandon M. Stewart, Dustin Tingley, Christopher Lucas, Jetson Leder-Luis, Shana Kushner Gadarian, Bethany Albertson, and David G. Rand. 2014. "Structural Topic Models for Open-Ended Survey Responses." *American Journal of Political Science* 58 (4): 1064–82. https://doi.org/10.1111/ajps.12103.

Russell, Stuart J., and Peter Norvig. 2022. *Artificial Intelligence: A Modern Approach*. Fourth edition, global edition. Pearson Series in Artificial Intelligence. Harlow: Pearson.

Scharkow, Michael. 2013. "Thematic Content Analysis Using Supervised Machine Learning: An Empirical Evaluation Using German Online News." *Quality & Quantity* 47 (2): 761–73. https://doi.org/10.1007/s11135-011-9545-7.

Schuldt, Jonathon P., and Sungjong Roh. 2014. "Media Frames and Cognitive Accessibility: What Do 'Global Warming' and 'Climate Change' Evoke in Partisan Minds?" *Environmental Communication* 8 (4): 529–48. https://doi.org/10.1080/17524032.2014.909510.

7 Appendix

7.1 GPT-3 prompt

```
Here are some open-ended responses from the British Election Study to the
    question "what is the most important issue facing the country?". Please
     assign one of the following categories to each open ended text
    response, returning the original response and the most relevant numeric
     code.
health: include NHS
education
election outcome
pol-neg: complaints about politics, the system, the media, corruption where
     no politician or party is mentioned
partisan-neg: use this code for negative statements about a party or
    politician
societal divides
morals: including abortion, complaints about religiosity and society's
    character
national identity, goals-loss: ethnonationalism
concerns about racism and discrimination
welfare
terrorism
immigration: include overpopulation. do not include refugees
asylum: asylum seekers/refugees
crime
europe: including Brexit
constitutional: UK constitutional issues except brexit, devolution and
    scottish independence. Include electoral system here.
international trade
```

devolution

scot-independence

foreign affairs: not including war

war: including Ukraine, Russia and Syria include desire for peace

defence

foreign emergency: short-term not including war

domestic emergency: short-term e.g. Flooding, storms and Grenfell Tower but not COVID

economy-general

personal finances

unemployment

taxation

debt/deficit: national/government debt not personal debt

inflation: price rises

living costs: including energy crisis

poverty

austerity: cuts and lack of spending on services

inequality

housing: including homelessness

social care: including care for elderly, adults, disabled and children

pensions/ageing

transport/infrastructure: including HS2

environment

pol values-authoritarian: pro-authoritarian responses or anti-socially liberal responses that don't fit better anywhere else: e.g. complaints about political correctness, LGBT people, wokeness etc

pol values-liberal: pro- socially liberal responses or anti-authoritarian responses that don't fit better anywhere else e.g. responses that are pro women's rights, LGBT people, free speech and other minorities or responses that are concerned about authoritarian groups

pol values-right: pro-economic right wing or anti-left wing responses that don't fit better anywhere else

pol values-left: pro-economic left wing or anti-right wing responses that don't fit better anywhere else

Referendum unspecified: where you can't tell whether response is about Brexit or indyref

Coronavirus: covid but not its economic impacts

Covid-economy

Black Lives Matter and backlash to it

other: for responses that do not fit into any other category

uncoded: for responses that cannot be reliably assigned to a category

For context, Russia invaded Ukraine prior to the fieldwork for this survey, so references to Ukraine or Russia are about war.

Responses about Brexit should be coded as Europe.

If multiple issues are mentioned, use the label for the first issue mentioned.

Code these cases:

a bad economy

immergration

a bad economy|economy-general

immergration|immigration

Code these cases:

climate change and unemployment

```
climate change and unemployment|environment,unemployment

Please only return one code per response (if multiple match use the first
    issue listed). The correct response should be:

climate change and unemployment|environment


Code these cases:
```

7.2 GPT-3 API call code

The following function makes an API call to open AI using the prompt (see previous appendix), a vector of open-ended survey responses, and a category lookup (to reflect the longer category titles used in the prompt compared with the original coding). An example coding three cases is shown below, but we coded 50 at a time for this paper. The categories assigned were inspected and cleaned to match the correct formatting (this was also done for the human coders).

The R code for the initial API call was written by chatGPT.

```
codeMIIGPTb <- function(prompt, cases, api_key,
model = "text-davinci-003", cats) {
  new.prompt <- paste(c(prompt, cases), collapse = "\n")
  body <- list(
    prompt = new.prompt,
    model = model,
    temperature = 0,
    max_tokens = 2000
  )
```

```
  api_endpoint <- "https://api.openai.com/v1/completions"

  response <- POST(api_endpoint,
                   add_headers("Content-Type" = "application/json",
                               "Authorization" = paste0("Bearer ", api_key)),
                   body = toJSON(body, auto_unbox = TRUE))

  response_json <- fromJSON(content(response, "text", encoding = "UTF-8"))
  completion <- response_json$choices[1, ]$text

  responses <- data.frame(do.call(rbind,
 strsplit(strsplit(completion, "\n")[[1]], "|", fixed = T)))
  colnames(responses) <- c("text", "long_label")
  responses$long_label <- sapply(strsplit(responses$long_label, ",", fixed
      = T), function(x) x[1])
  responses$label <- cats$label[match(tolower(responses$long_label),
      tolower(cats$long_label))]

  return(responses)
}

codeMIIGPTb(cases = paste("the economy post covid",
                          "LACK OF CONTROL OF IMMIGRATION",
                          "Possibility of Scottish Independence splitting the
                              Union"),
                      prompt = prompt,
                      api_key = "YOUR-API-KEY",
                      model = "text-davinci-003", cats)
```

7.3 R SVM Code

This appendix shows the R code used to fit the SVM to 1,000 random cases. The approach for fitting the SVM to all responses from waves 1-20 followed the same logic.

```
# SVM: 1,000 random cases

# Housekeeping -------------------------------------------------------

# Load libraries

library(tidyverse)
library(RTextTools)
library(jbmisc) # remotes::install_github("jackobailey/jbmisc")
library(haven)
library(here)


# Set random seed

set.seed(666)


# Read in BES data

str_data <-
  read_dta(
    here(
      "_data",
      "raw",
```

```
    "BES2019_W23Strings_v23.1.dta"
  ),
  col_select =
    c(
      id,
      num_range("MII_textW", 21:23),
      num_range("mii_catW", 21:23)
    ),
  encoding = "latin1"
) |>
drop_na(id)



# Transform text data ----------------------------------------------------

# Convert strings data to long format

str_data_long <-
  str_data |>
  pivot_longer(
    cols = -id,
    names_to = c(".value", "wave"),
    names_sep = "W"
  ) |>
  rename(
    miitext = MII_text,
    miicat = mii_cat
  )
```

```
# Mark empty strings and "__NA__" as missing

str_data_long <-
  str_data_long |>
  mutate(
    across(
      c(miitext, miicat),
      function(x) mark_na(x, c("", "__NA__"))
    )
  )



# Drop missing data list wise

str_data_long <-
  str_data_long |>
  drop_na()



# Sample 1,000 strings at random

doc_data <-
  str_data_long |>
  sample_n(
    size = 1000,
    replace = FALSE
  )
```

```
# Remove extraneous data

doc_data <-
  doc_data |>
  select(
    miitext,
    miicat
  )


# Create document matrix

doc_matrix <-
  doc_data |>
  select(-miicat) |>
  create_matrix(
    language = "english",
    removeNumbers = TRUE,
    stemWords = TRUE,
    removeSparseTerms = 0
  )


# Create a container

container <-
  doc_matrix |>
  create_container(
    labels = doc_data$miicat,
    trainSize = 1:nrow(doc_data),
```

```
    virgin = FALSE
  )



# Fit SVM Model ------------------------------------------------------------

# Train svm model

svm <-
  train_model(
    container,
    "SVM"
    )


# Save trained model

saveRDS(
  svm,
  here(
    "_data",
    "proc",
    "svm_sample.rds"
  )
)



# Fit to new overall data -------------------------------------------------
```

```
# Load human-coded data

load(
  here(
    "_data",
    "proc",
    "overall_sample.rda"
  )
)



# Transform data to match training data

overall.sample <-
  overall.sample |>
  select(miitext = MII_textW)



# Create testing matrix

overall_matrix <-
  overall.sample |>
  create_matrix(
    originalMatrix = doc_matrix,
    language = "english",
    removeNumbers = TRUE,
    stemWords = TRUE,
    removeSparseTerms = 0
  )
```

```
# Create container

overall_container <-
  overall_matrix |>
  create_container(
    labels = rep(0, nrow(overall.sample)),
    testSize = 1:nrow(overall.sample),
    virgin = FALSE
  )


# Classify data using trained model

overall_classify <-
  classify_model(
    container = overall_container,
    model = svm
  )


# Load the overall data again

load(
  here(
    "_data",
    "proc",
    "overall_sample.rda"
  )
```

```
)


# Merge in SVM-predicted category

overall_sample_svm_1000 <-
  overall.sample |>
  select(
    id,
    wave,
    MII_textW,
    code,
    label
  ) |>
  mutate(
    svm_subsample_code = as.numeric(overall_classify$SVM_LABEL)
  )



# Save to disk

save(
  overall_sample_svm_1000,
  file =
    here(
      "_data",
      "proc",
      "overall_sample_svm_1000.rda"
    )
)
```

```
# Glimpse accuracy

sum(overall_sample_svm_1000$code ==
    overall_sample_svm_1000$svm_subsample_code, na.rm = T)/nrow(
    overall_sample_svm_1000)



# Fit to new unique data ------------------------------------------------

# Load human-coded data

load(
  here(
    "_data",
    "proc",
    "unique.sample.rda"
  )
)



# Transform data to match training data

unique.sample <-
  unique.sample |>
  select(miitext = MII_textW)



# Create testing matrix
```

```
unique_matrix <-
  unique.sample |>
  create_matrix(
    originalMatrix = doc_matrix,
    language = "english",
    removeNumbers = TRUE,
    stemWords = TRUE,
    removeSparseTerms = 0
  )


# Create container

unique_container <-
  unique_matrix |>
  create_container(
    labels = rep(0, nrow(unique.sample)),
    testSize = 1:nrow(unique.sample),
    virgin = FALSE
  )


# Classify data using trained model

unique_classify <-
  classify_model(
    container = unique_container,
    model = svm
  )
```

```
# Load the overall data again

load(
  here(
    "_data",
    "proc",
    "unique.sample.rda"
  )
)


# Merge in SVM-predicted category

unique_sample_svm_1000 <-
  unique.sample |>
  select(
    id,
    wave,
    MII_textW,
    code,
    label
  ) |>
  mutate(
    svm_subsample_code = as.numeric(unique_classify$SVM_LABEL)
  )


# Save to disk
```

```
save(
  unique_sample_svm_1000,
  file =
    here(
      "_data",
      "proc",
      "unique_sample_svm_1000.rda"
    )
)

# Glimpse accuracy

sum(unique_sample_svm_1000$code ==
    unique_sample_svm_1000$svm_subsample_code, na.rm = T)/nrow(
    unique_sample_svm_1000)
```