Annotating elite partisan sentiment on twitter with GPT: effective for both ordinal and scaled measurement?

Ben Birch

Email: benbirch99@btinternet.com

Github: https://github.com/benbirch99

Abstract — I use the GPT application programming interface (API) to systematically implement GPT-4 (the most recent ChatGPT model released by OpenAI) and text-da-vinci-003 (a legacy instruct model of GPT 3.5) for both ordinal classification and Best-Worst Scaling (BWS) of elite partisan sentiment. Different types of context-embedded prompts are trialled on tweets sent by twitter accounts associated with US partisan elites which contain terms associated with other US partisan elites.

When comparing scores to those of human annotators, GPT-4 out-performs text-davinci-003 for 3 out of the 4 tasks. However, the two models (GPT-4 and text-davinci-003) demonstrate better performance in measuring negative sentiment in both BWS and ordinal classification tasks. Whilst correlations between GPT and human BWS scores are varied, a range of promising avenues available for future research seeking improvements on this front are discussed.

GPT and BWS are leveraged, in part, based on a fundamental desirability for fine-grain, cost-effective, and scalable automated variable measurement tools within social science. Whilst both of these tools are commonly used to make improvements to aspects of variable measurement, this study seeks to galvanize future research looking to creatively combine methods from different domains to create rounded improvement in automated variable measurement.

Index Terms— GPT, Large Language Models, Best-Worst Scaling, Affective Polarization, Partisan Sentiment, Natural Language Processing, Twitter

1 Introduction

The release of GPT-4 and its evidenced efficacy has prompted a growth of GPT-related research and methods within social science. This research is among others seeking to trial GPT in unchartered territory. Whilst GPT performance has been tested for stance (see Gilardi 2023) and sentiment-based classification (see Gilardi 2023; Rathje 2023), this is, to the author's best knowledge, the first implementation of GPT specifically for partisan sentiment classification. Moreover, GPT has not intersected with Best-Worst Scaling methods in general, which is surprising given the current limitations of GPT when ranking and scoring large lists of items (attested by Wu et al., 2023, p4) and the efficiency of BWS. Exploring BWS with GPT is therefore hoped to inform the social science community on the types of design considerations that merit attention when applying GPT for BWS.

Aside from the novelties regarding GPT utilisation, the implementation of BWS (regardless of GPT) for partisan sentiment is also, to my best knowledge, an original contribution. Whilst BWS has more recently been applied in Natural Language Processing (NLP) contexts (see Hollis 2018; Hollis and Westbury 2018; Kiritchenko and Mohammad, 2017), BWS is a rare in social science fields due to methodological and computational concerns pertaining to its use with a large number of items (see section 2.4) despite recent development in BWS research that make it more scalable for large samples (see Hollis 2018).

Due to the aforementioned novelties, traditionally underreported methodological choices are discussed in detail with the aim that others can build upon and greatly improve procedures taken. Relatedly, the various methodological choices deployed place a particular focus on their future applicability on larger data across different contexts to aid researchers in reducing financial and time constraints while automating variable measurement.

Despite delving into novel research areas, it is also hoped that the presented target-based operationalisation of elite partisan sentiment brings conceptual clarity. "Elite partisan sentiment" simply refers to the partisan sentiment expressed by elites on twitter. I operationalise partisan sentiment in two ways pertaining to its measurement as both an ordinal and scaled variable respectively. For ordinal classification tasks, tweets are placed on a three-point scale and classified as either positive, negative, or neutral towards the in-partisan and/or out-partisan group. When scaling with BWS, however, partisan sentiment is operationalised as a concept of in-partisan positivity and out-partisan negativity (see Methods section).

For both operationalisations the two key institutions within US politics, the Democratic and Republican Party, form the two partisan groups conceptualised. Further, each partisan group is conceptualised as encompassing two types of target: the political party as a whole and individual partisan elites residing within the party. Partisan elites comprise all Members of Congress serving within the time period of interest (April – November 2021) alongside the current president and most recent former president of the time period (in this case Biden and Trump). As this is a fundamentally intuitional conception of partisan elites, I refer to these individuals as institutional partisan elites hereafter (abbreviated to IPEs).

2 Related Work/Literature Review

Whilst a range of Machine Learning (ML) classification research has been devoted to creating classifiers for tweet corpuses, there has been an overwhelming attention to broadly two research areas: overall sentiment classification, which is frequently referred to as "valence" classification

(Cliché, 2017; Rosenthal et al., 2017; Jianqiang et al., 2018; Nakov et al., 2019; Naseem et al., 2020) and online hostility; the research on online hostility has encompassed various semantic categorisations such as "hate-speech" (Waseem et al., 2016; Badjatiya et al., 2017; Zhang, 2017; Basile et al., 2019), "toxicity" (see google's Perspective API) , "abuse" (Park and Fung, 2017), and "offense" (Zampieri, et al., 2019; Zampieri, et al.,2020).

The implementation of Neural Networks to both transform texts into vector spaces and generate predictive classes from inputs has helped make inroads into various NLP tasks. Word embeddings (vector representations of words) obtained from neural networks, where surrounding words in a text are predicted based on inputs, have demonstrated amenability to clustering and dimensionality reduction methods.

Clustering these embeddings has proved useful in topic modelling (see for example BERTopic) among other highly context-specific classification tasks. For example, Rheault et al. (2019) use the word2vec architecture with dimensionality reduction to scale the ideology of parliamentary speeches. In a context closely related to this research, Rasmussen (2023) similarly train word2vec embeddings on a corpus of tweets and subsequently combine word embeddings to create focal constructs such as "political hostility" (which is just the average of the trained embeddings for "political" and "hostility"). The cosine similarity of tweets to the focal construct vector is then used to classify tweets according to the construct. Whilst this method is fraught with problems – the most obvious being that a tweet that talks about the political hostility is classified as being politically hostile – the research is illustrative of the potential application of neural networks to automate variable measurement pertaining to the efficacy of neural network architectures to map words into vector spaces.

The growing types of neural network architectures used to train word embeddings has also been leveraged in NLP research. For example, Naseem et al. (2020), use multiple vector space representations for a single text provided by pre-trained embedding transformers to predict the sentiment of tweets. The architecture used in Naseem et al. (2020) is particularly illustrative of the utility that neural networks provide in NLP as the vector representations are subsequently used as inputs into a Long Short-Term Memory Model (a type of recurrent neural network).

Despite the notable improvements in classification performance in NLP tasks attained through the expansion of effective neural network architectures, deep learning for target-based classification has received relatively little attention. "Target-based" classification denotes a task which measures an attitude, feeling, emotion, or sentiment toward a specified entity. For example, "stance" classification, in the context of twitter classification, involves classifying a tweet in accordance with its favourability toward a specified entity (see Mohammed et al., 2017). Whilst online hostility is frequently understood to have a target in the sense that there is an identifiable receiver of vitriol (an individual or group), in only a subset of online hostility research is the identity of the receiver integral to the classification of the hostility (see Waseem, 2016; Gambäck and Sikdar, 2017; Badjatiya et al., 2017; Park and Fung, 2017; Zhang, 2017; Basile et al., 2019 for examples). Nonetheless, in cases where the identity is integral to the measurement of hostility, such as the classification of online sexism or racism, the efficacy of the target-based hostility framework is frequently limited by the stricter benchmark of hostility when compared to sentiment or stance. Despite the conceptual blind spots embedded within the hostility approach with regards to capturing complex phenomena, the vast majority of target-based NLP tasks have been structured within hostility operationalisations.

The concept of sexism is exemplary of the aforementioned limitations within computational linguistics. Sexism can be considered a form of hate speech where hate is directed to an individual or group. However, sexism or misogyny could rather be operationalised as an expressed *unfavorability* (stance) or *negative sentiment* toward women, as a text may express sexism without necessarily directing hate at women. Further still, a text could still be considered to uphold sexist attitudes without expressing hate, unfavorability, or negativity directly towards women; within a stance framework, sexism could instead be conceptualised as an unfavourable opinion towards *gender equality*, which seems to more expansively capture sexism (although this necessitates researchers to set out exactly what it is they understand by the term gender equality). Whilst target-based frameworks have the potential to capture concepts with depth, more simplistic and learner target-based frameworks have gained predominant traction in NLP research.

One plausible explanation for the relative inattention to automated target-based classification outside of online hostility frameworks (see Jiang, 2011; Dong, 2014 for exceptions) is that training models for these tasks is notably more difficult. Training a classifier for target-based classification typically requires a higher specificity of data, which presents barriers to attaining the required data which in turn likely contributes to the smaller sample sizes used in the rare instances of target-orientated datasets designed for ML classification (see example of SemEval below). Indeed, it is conceptually plausible that any text can be classified in relation to its sentiment or hostility. For example, the English Lexicon Project operationalises valence as an underlying latent variable for which all words can be scored. In contrast, classification for stance is more commonly understood to be restricted by topic; under this view, it does not make sense to classify a tweet that doesn't reference abortion as 'neutral' toward abortion.

Additionally, training accurate and conceptually expansive target-based classifiers may require more complex linguistic associations to be established, which in turn, may necessitate larger training samples. These factors – that larger training samples are required yet are simultaneously more difficult to attain– might explain why classic ML models have not performed well on out-of-sample data in the few instances where target-based classifiers have been used (see Dong et al., 2014, Table 1; Jiang, 2011, p157 Table 5).

The SemEval International Workshop on Semantic Evaluation, which hosts annual ML classification tournaments for Natural Language tasks and is a leading contributor of providing publicly available tweet datasets, is illustrative of the disparities across NLP tasks. The SemEval-2016 Task 6 produced a dataset comprised of

4000 tweets annotated for their stance across 5 topics (such as whether a tweet is for or against the right to abortion or neither), with a sample size of n < 1000 for each topic. However, SemEval-2016 Task 4, which involves overall ordinal sentiment classification of tweets with subtasks on 2-point (positive or negative) or three-point (positive, negative, or neutral) level, afforded modelling teams with samples of n >10000 and n > 20000 for the two subtasks respectively.

Moreover, in SemEval the sample sizes available for classic sentiment classification tasks increased in subsequent years; SemEval-2017 Task 4, for example, further provided an increased sample size for 2-point and 3-point sentiment classification (n > 20000 and n > 50000 respectively), with results demonstrating superior classifiers had been built by modelling teams for both subtasks compared to the previous year (see Najov et al., 2019 p11 Table 11 and p10 Table 9 for 2-point and 3-point ordinal classification performance for SemEval-2016 Task 4; see Rosenthal et al., 2017, p509 Table 6 and p510 Table 8 for 2-point and 3-point ordinal classification performance for SemEval-2017 Task 4). Whilst sentiment classification has seen sharp performance increases, stance detection has not yet been reimplemented in a SemEval task despite trained classifiers displaying comparatively poor performance (see Mohammad et al., 2016, p36 Table 2).

2.1 Automated Elite Partisan Sentiment Classification

With regards to the performance of target-based ML classifiers specifically for partisan sentiment classification, the current gap between the effectiveness of classifiers in well-established research areas (such as overall sentiment) stands particularly stark. Despite utilising an extensive array of traditional ML methods with substantial training samples, trained ML classifiers for partisan sentiment classification (Yu et al., 2023; Wojcieszak et al., 2022) have produce models which perform poorly relative to leading classifiers for standard NLP tasks (see Badjatiya et al., 2017, pp.2 Table 1; Cliché, 2017; Park and Fung, 2017, pp.44 Table 2; Jianqiang et al., 2018, Table 2; Naseem et al., 2020, pp.67 Table 4).

Yu et al. (2023), for example, train machine learning models on a sample of N = 10000 by inputting the TF-IDF (Term-Frequency Inverse Document Frequency) matrix representations as inputs to predict whether an elite tweet is positive, negative or neutral toward the US Democratic and Republican Party separately. Specifically, they train Random Forest, Support Vector Machine, K-Nearest-Neighbor, Gradient Boosting Machine, and Decision Tree models for the two parties (i.e 10 models in total) and additionally code for the partisan affiliation of the sender in order to measure in-party and out-party sentiment. The best performing ML method for both parties – the random forest – nonetheless produced classifiers which demonstrated generally unimpressive out-of-sample accuracy (see figure 1; 74.2 for Democrat and 66.4 for Republican).

Wojcieszak et al. (2022) similarly use the TF-IDF to train support vector machine, Decision tree and a K-neighbours models, alongside an ensemble model that takes the majority class predicitons of the aforementioned models, in order to predict whether a "quote" tweet in response to an original elite tweet is positive, negative, or neutral toward the original sender or his/her message. Wojcieszak et al. (2022) also use substantial training data (n=8351) and further use a 4-layer Convolutional Neural Network (CNN) trained on 300-dimension GloVe embeddings as vector representations of tweets as inputs. However, akin to Yu et al. (2023), the best performing model (the CNN) demonstrated relatively poor performance (overall out-of-sample accuracy = 62.2, see Supplementary Material p8 figure S5).

2.2 GPT for Text Classification

GPT offers promise in offering an alternative to traditional ML methods for automating partisan sentiment where the aforementioned shortcomings are effectively obviated. The burgeoning literature trialling GPT for text annotation has established that recent models can out-perform crowd workers on tweet sentiment classification (Gilardi et al. 2023; Rathje et al. 2023) and stance (Gilardi et al. 2023). This holds true even when the cognitive difficulty of tasks is high (see Gilardi et al. 2023, who trial notably complex tasks).

Relatedly, recent GPT models have illustrated that they are able to effectively incorporate contextual information outside of zero-shot prompts to complete specific tasks that are highly knowledge-dependent (see Wu et al. 2023). Moreover, the frequent updating of GPT models based on reinforcement learning from human feedback widens the range of contexts in which researchers can expect high performance. Thus, GPT may prove an effective tool in reducing financial and time constraints of researchers, without sacrificing the quality of annotations researchers could expect from human annotators in targeted partisan sentiment.

Aside from the abundance of results that are demonstrative of GPT's performance, a tangible benefit of Large Language Model (LLM) integration for variable measurement is that specific training datasets are not required of the researcher. This plays an instrumental role in adding to the efficacy of LLMs for partisan sentiment in a target-orientated operationalisation considering the aforementioned technical challenges in obtaining specific, high quality training data.

To train a classifier for target-based elite partisan sentiment in the current research context, one would preferable want hand labelled *tweets,* sent by *MOCs,* that *contain partisan rhetoric.* More specifically, one would ideally want many tweets sent by Democrats of which many mention Democrats and Republicans, as well as many tweets by Republicans of which many mention Republicans and Democrats. Indeed, imbalances in training datasets have been shown to reduce the generalizability of ML classifiers (Wich et al. 2020).

For instance, if 80% of the tweets that reference individual Democrats by Republicans are directed specifically at Joe Biden, an ML classifier trained on the data may perform badly in out-of-sample contexts (for example one that wasn't temporally proximate) by building associations that are too bound to the target imbalance in the training data. This bias might reasonably be termed "reference bias". This is, however, merely one aspect of the imbalance or "bias" difficulty of designing training data sets that are suitable for ML classification; another is "author bias" (Wich et al. 2020), where a small number of authors

are responsible for the majority of tweets in the sample, which may in turn produce other biases such as the previously mentioned reference bias or topic bias; "topic bias" (Skyte, 2021) must similarly be considered if tweets are scraped within specific temporal parameters as elites inherently tweet about trending news topics, which can create associations with topics that aren't generalizable in out-of-sample contexts. Ultimately, a traditional ML classifier will have to contend with these potential imbalances being empirically confounded with partisan sentiment.

The ready availability of LLMs stands in even greater contrast to alternatively training a classifier when considering recent developments in GDPR practices and management in social media contexts. Whilst research measuring partisan sentiment on an ordinal scale with sufficiently large samples of human annotations to train an ML classifier exists (see Russell, 2016, n > 8000; see Yu et al. 2023, N = 10000; see Auter et al. 2016, N > 14000), current data policies of twitter require that collected tweets are made neither publicly available nor available on request. In other words, researchers currently cannot use other researchers' annotated tweets to train classifiers. Moreover, recent changes in the twitter API mean that it is no longer financially accessible to obtain very large samples, which would stand to benefit the most from trained classifiers.

Furthermore, other adjustments in the twitter API mean that it is no longer possible to filter tweets through extensive keyword searches. Specifically, changes to API rate limits make it unfeasible to loop over a large list of elites and perform key word searches for each elite because the key-word search exceeds the word limit available (there's a cap on the number of words that can be included in a key word search); alternatively looping over some subset of key terms for each elite account until all key terms have been included for each elite is also restricted by limits on the number of key searches that can be performed on a single twitter user by an API user within a time range. Therefore, in order to obtain training samples that accommodate extensive key word searchers, the filtering has to be done after tweet collation. These restrictions altogether impose obstacles to obtaining both high quality data with a sufficient sample size for target-based partisan-sentiment classification.

The current difficulty of obtaining training data for automating partisan sentiment perhaps explains the practice of using proxies for partisan sentiment in social science. For example, Osmundsen (2021), utilises SentiStrength to measure the overall sentiment of news articles that mention US Republicans and US Democrats as a proxy for the measurement of "negativity and positivity towards republicans and democrats within an article" (p1012). SentiStrength builds on dictionary-based approaches by using linguistic rules to additionally weigh the presence of valence shifters (such as "very", "hardly"), spelling ("amazingggg"), punctuation ("great!"), and other contextual information within texts.

Similarly, Garrido et al. (2021) use VADER to score tweets by their polarity when they contain key words association with immigration. Garrido et al. (2021) then use these scores as proxies for anti-immigration sentiment. VADER is similar to SentiStrength as it is also a rule-based approach to sentiment measurement that expands on traditional dictionary approaches; however, it specifically considers the linguistic idiosyncrasies of social media.

However, Mohammad al. (2016, p.9) show that class co-occurrence between sentiment dictionaries and target-based sentiment is sparse (particularly for positive and neutral classes on a three-point ordinal scale); furthermore, Rathje et al. (2023) has produced results suggesting that the correlation between human "gold standard" classes and dictionary methods are low for general sentiment whilst GPT-4 shows comparatively better performance (pp. 11, Table 5, r = 0.66-0.75 for GPT-4; r = 0.20-0.30 for dictionary-based methods).

The use of GPT therefore shows further promise in averting the pull towards analytical conflation of sentiment and target-based sentiment through the reduction in the cost and time demanded of researchers to obtain effective measurements (see section below).

2.3 Best-Worst Scaling: Background

BWS is a method of choice modelling where annotators make an ordinal decision by assigning one item in a block to "best" and another to "worst." Using standard block design notation, BWS generates b number of blocks (which can be referred to as "tuples") of size k containing t items for annotation (t is interchangeable with N), where items appear on average r times and pairs for all items appear on average λ number of times.

Compared to other ranking-based methods, BWS is particularly efficient given that single judgements over a set generate multiple pair rankings of the k items. Generating complete pair-rankings of all t items, in comparison, requires $\binom{t}{2}$ blocks which scales poorly. These attributes make BWS a promising framework for generating a scaled variable of partisan sentiment, particularly when bearing in mind that GPT has rate-limits and calculates costs based on input and output token counts.

As a point in case, Wu et al. 2023, use the Bradley-Terry model to scale the ideology of Senators on a one-dimension left-right scale. Wu et al. 2023 use GPT to classify 102 Senators on an ordinal scale and subsequently limit pair-wise comparisons to senators within the same or adjacent ordinal categories. While this approach reduces pairwise comparisons drastically compared to the full $\binom{102}{2}$ match ups in a full pairwise design (b = 5151), 2667 matchups are nonetheless generated and require annotation for the 102 senators (see section 3 in Wu et al., 2023). In a hypothetical comparison, BWS using b = $2t$ and k = 4 (i.e 2N 4-tuples) which has been recommended in previous research (see Mohammad 2018; Kiritchenko and Mohammad, 2017) would instead generate 204 blocks to be annotated. Even when parameters are optimised for pair balance – aiming for maximised proximity to λ =1 (approaching the most pair-balanced design given k and t) – we would still only require 859 blocks to be annotated using k = 4 (see following section).

Aside from the potential improvement to scalability garnered by BWS, there are many limitations that can afflict ratings scales that are avoided through the item choice design of BWS. For example, different annotators often have a bias towards parts of the scale, which can be adjusted using statistical methods in the case of many annotators but is hard to mitigate in the case of few annotators. Given the annotators make ordinal *decisions* rather than ordinal

ratings in BWS, this methodological concern does not pertain.

Additionally, in rating scales, different annotators may have an equal "ability", yet use different scores for the same items. For example, on a 10-point scale measuring term valence, it is unclear whether "fine" is a 5 or 6, which can make finding a benchmark with which to assess scores difficult. Additionally, annotators do not have access to all of the scale; they might think that an item is between two ordinal points, yet they cannot express this judgement (see Kiritchenko and Mohammad, 2017 for an overview).

Alongside potential benefits from the vantage of Item Response Theory and scalability, BWS also has desirable properties from a statistical modelling point of view. In particular, BWS has been shown to generate more granular and reliable scores compared to those obtained with ordinal scales (see Kiritchenko and Mohammad, 2017).

Accurately increasing the measurement granularity has the potential to alter inferences. For example, BWS for partisan sentiment could help better capture linear relationships in modelling where researchers are otherwise restricted to ordinal measurement of partisan sentiment and tempted to treat such scales as though they represent continuous data. Conversely, scaling can also be preferable when trying to capture non-linear relationships, as the use of categorical variables can lack sufficient granularity to capture underlying non-linear distributions. This is most pertinent in the case of binary categorical variables in a regression context (i.e when a tweet is classified as either hostile or not) where there is no potential to capture nonlinearity to begin with.

2.4 Best-Worst Scaling: Theory

To maximise the amount of relational information retrieved from ordinal decisions made by annotators, in BWS it is typically required that no item appears in the same block more than once and that no blocks contain the same items. Additionally, for small sample sizes, BWS tuples typically use a balanced incomplete block design (BIBD) where each item appears in exactly r blocks and each possible pair appears exactly λ number of times. Thus, BIBDs for BWS require that r and λ are integers and that lambda satisfies equation 1:

$$\lambda = \frac{r(k-1)}{(t-1)} \quad (1)$$

Satisfying that each item appears in the same number of blocks is typically non-problematic. Given t items and k block sizes, b is typically chosen to fulfil a specified r and vice versa (where r is chosen based on a specified b) which is simply derived from equation 2:

$$r = \frac{k(b)}{t} \quad (2)$$

However, ensuring lambda is an integer becomes problematic when dealing with a large number of items as the number of blocks required greatly increases, demonstrated in equations 3 and 4 below. Equation 3 is used to calculate the average number of times a pair occurs across the blocks, λ, by taking the total number of pairs that appear across the blocks and dividing the number by the number of unique pairs given t. Equation 4 is simply its rearrangement for b:

$$\lambda = \frac{b\binom{k}{2}}{\binom{t}{2}} \quad (3)$$

$$b = \frac{\lambda\binom{t}{2}}{\binom{k}{2}} \quad (4)$$

The increase in b for a given λ and k generates practical problems for maintaining BIBDs when dealing with large samples; it can become financially unfeasible to obtain b annotations. However, from a computational standpoint, algorithmically generating BIBDs also becomes problematic due to the combinatorial explosion of pairs that need to be considered given t.

Whilst earlier work has recommended BIBDs to mitigate potential bias derived from unbalanced pair occurrences and concomitant disparities in competition within sets (Louviere et al., 2013), algorithmic scoring methods have been applied to BWS responses in incomplete block designs (IBDs) where $\lambda < 1$ and demonstrated high correlations with gold standard values along a latent dimension (see Hollis 2018; Hollis and Westbury 2018; Hollis 2019). These scoring algorithms, such as Elo (used for chess rankings), Rescorla-Wagner, and value learning, have the desirable property that the "outcomes" of pairwise matches within blocks are weighted by estimations of the level of competition from previous matches.

These methods have therefore more recently been leveraged to work with much larger sample sizes than those typically used with BWS without sacrificing the quality of value estimates (BWS with BIBDs has typically been used with $t < 50$, but $t > 800$ has been used in Hollis 2018; Hollis and Wesbury 2018; Hollis 2019).

Hollis (2018) and Hollis and Westbury (2018) performed automated simulations of BWS using different scoring methods with item scores known ahead of time. Items with the greatest scores in a block are automatically chosen as "best" and those with the smallest scores are automatically are chosen as "worst." In these performed simulations (Hollis 2018; Hollis and Westbury 2018), the distribution of scores, the level of noise, k and b are systematically varied given t selected items. An Analysis of Variance, à la Hollis (2018), found evidence that across conditions scoring algorithms outperform traditional scoring methods in the case of $\lambda < 1$ and that scoring algorithms are able to achieve notably highly correlations with the underlying true values. Behavioural experiments using human annotations of items where the true values are known, have also corroborated that scoring algorithms produce scores that are more highly correlated with benchmark values (see Hollis 2018, experiment 4).

Using scoring algorithms to obtain estimates of item values therefore has potential to push the boundaries in obtaining high quality scaled measurements that are efficient with respect to time and financial costs.

3 Operationalization Frameworks

I use a target-based operationalisation of partisan sentiment for both ordinal classification and BWS that uses programming logic to isolate words and phrases within tweets (terms) to individual MOCs, whereby prompts are

subsequently algorithmically generated and fed into GPT (see section 4.2 for the logic used in key term matching and appendix section 1 for example prompts and python code).

While both operationalisations consider the partisan relation of the target to the author (in-partisan or out-partisan) and the form the target takes (a specific MOC or the party as a whole) creating 2x2 variables, BWS for partisan sentiment scales in-partisan sentiment on *positivity* and out-partisan sentiment on *negativity*. In contrast, a tweet can be measured as negative towards the in-partisan target and positive toward the out-partisan target when partisan sentiment is ordinally classified.

Despite the poor performance of traditional ML classifiers, partisan sentiment classification operationalized as targeted sentiment toward a partisan entity is, I proffer, the most conceptually sound and applicable way to consequently answer pertinent research questions pertaining to behavioural partisan affective polarization (see section 7.2).

Firstly, with regards to ordinal classification, target-based sentiment is preferable to target-based hostility because the benchmark set by hostility frameworks are comparatively restrictive; whilst elites criticise their partisan opponents, they frequently stop short of using insulting or abusive language.

Secondly, the target-based conceptualisation allows for separately measuring sentiment for different types of partisan targets. With a target-based approach, there is a concomitant ability to distinguishing sentiment directed toward the in-partisan or out-partisan group, in the form of the partisan group as a whole or its comprised individuals; in particular, the targeted approach allows for separate measurements for sentiment towards different partisan targets *within a tweet*, which seems conceptually sound given that a tweet can express sentiment toward the in-partisan group and sentiment toward the out-partisan group, express sentiment toward only one partisan group, or neither of them; for a given partisan target, a tweet can also reference the partisan group as a whole, MOCs, both, or neither.

For BWS, a target-based sentiment conceptualisation is also deemed more appropriate considering the various qualitative information cited in previous research regarding the idiosyncrasies of GPT. Previous research has accounted that GPT is highly sensitive to the wording used in tasks which require comparative decisions to be made about items on a latent variable. Wu et al. 2023 have found that when asked to pick the 'most liberal' of two senators who are Republicans, GPT struggles. Yet, when asking who is the 'most conservative,' GPT is effective. Thus, Wu et al. 2023, who utilise a Bradley-Terry Model, vary the wording between matched pairs such that the 'most liberal' senator among Democrat pairs are chosen and conversely the 'most conservative' among Republican pairs are chosen. Within a BWS design, however, the hostility framework is not amenable to the aforementioned approach because focal concepts, such as 'toxicity', or 'hostility', lack clear semantic counterparts (one could make a case for *complimentary*, *amenable*, *approving*, *supportive* etc). By contrast, 'positive and 'negative' are unambiguously diametrically opposed (akin to 'liberal' and 'conservative.')

Partly based on the aforementioned idiosyncrasy of GPT in its current form, I scale in-party positivity and out-party negativity with BWS on two dimensions. Tweets that are originally annotated as being positive toward the author's in-party are scaled for their positivity and tweets that are originally annotated as being negative toward the author's out-party are scaled for their negativity. This increases the likelihood that the tweets in a set possess some level of the underlying latent variable in question which is congruent with the framing of the prompt. There are other methodological considerations, however, that also render collapsing in-party positivity and out-party negativity for BWS on a single scale potentially problematic.

One particular concern is the skewed (there are far more negative tweets – see appendix section 3 table A11 and A13) and potentially bimodal distribution of the underlying values. Unfortunately, it is generally unclear how bimodality effects the accuracy of BWS scores; while BWS has been trialled on a range of distributions (see Hollis 2018), I am unaware of any previous research that has applied BWS to items sampled from bimodal distributions. Given that a key methodological aim of this study is to aid research in developing scalable variable measurement when dealing with large data, separating partisan sentiment into two variables was considered a more future-proof method across different context.

Aside from a robustness to pitfalls associated with alternative approaches, it is hoped that the granularity attained by the operationalizations allow for greater hypothesis testing. One notable benefit within a regression context is the capability to interact different variables attained from the measurements. For example, the current operationalisations allow for an interaction between the sentiment of a tweet toward a target and the ideology of the target (see Appendix Section 1.2 for GPT prompt examples).

Yet, these granularities still allow for aggregations if desired in specific research contexts. For example, sentiment toward a partisan entity can be aggregated as a single measurement, as opposed to the sentiment toward its individuals and the whole; additionally, in cases when multiple partisan entities are referenced in a single text, the aggregated sentiment of each partisan target can be scored flexibly, although simply taking the mean seems most straightforward.

Measuring partisan sentiment as both an ordinal and scaled variable within a target-based conceptualisation should further bring insights into how well GPT performs within a niche when substantially different frames are deployed; it is possible that GPT performance is or isn't notably slanted toward BWS or ordinal classification within this set-up, which is useful information in itself.

4 METHODOLOGY

4.1 Data Collection

I merge two datasets to obtain institutional partisan elite (IPE) twitter accounts (see section 1 for definition of IPE): the University of California San Diego dataset for the 177th Congress (https://ucsd.libguides.com/congress_twitter#s-lg-guide-main), which contains only official twitter handles associated with IPEs and data used by Mills (2021) which contains a manually coded list of all types of twitter accounts associated with members of the 117th Congress, including their official, personal, campaign, and press

release accounts (https://towardsdatascience.com/take-it-to-twitter-sentiment-analysis-of-congressional-twitter-in-r-ee206a5b05bc).

I use Mills's dataset as it was collected within the timeframe of interest (see below for relevance). I scrape all tweets sent by obtained twitter accounts associated with IPEs between 1st April – 1st November 2021, such that all keyword matching for identifying targets was done *after* tweet collation. I only scrape tweets, as opposed to retweets, as they often do not constitute the retweeter's opinion and opt to focus specifically on IPEs as national-level politics captures the majority of attention and news compared to state and county level politics in the US (Hopkins,2018).

To construct the list of accounts from which tweets were extracted, only those Members of Congress who were serving during the 1st April – 1st November 2021 were considered as IPEs. Whilst I merged Members of Congress according to surname, party, state and chamber (Senate or Representative), the vast majority of unmatched individuals were accounted for by different spelling of respective surnames (although some were unmatched because they had passed which was correctly mitigated by Mills's manual data collection occurring within the timeframe of tweet collation). Unmatched IPEs across the datasets (both ways), were manually checked such that where IPEs were known by multiple surnames, one surname was treated as a middle name which is effectively interpreted by the key term matching algorithm (see following section and Appendix section 1). In line with previous research (Yu e tal., 2023), Senators Angus King and Bernie Sanders are labelled as belonging to the Democratic Party because they have consistently caucused with the Democratic Party.

After IPEs had been placed in the working dataset, differences in full name interpretations between the two original datasets and the new working data set were manually checked and updated. Given that tweets are tokenized and the colloquial format of social media means that various punctuation is often omitted, the first word in compound surnames is removed and appended to an IPE's middle name so that it can be correctly processed by the key term matching algorithm (see following section and Appendix section 1).

Lastly, upon manually checking and updating all unmatched IPEs, only one individual serving in the 117th congress didn't have any type of twitter account who also passed before the time period of interest. Therefore, the same dataset is used for both scraping tweets and creating key terms associated with each IPE. The code used to create the working dataset for tweet collation and the dataset itself is available on github (https://github.com/benbirch99).

4.2 Key Term Matching for Contextual Embeddings

To ensure that all potential references to IPEs were correctly matched, the formatting of IPEs' names were standardised, such that all diacritics were removed. The same was true of the tweets themselves, which were lowercased and tokenized. All URLs and punctuation were also removed, other than hashtags, which remained attached as the first character of their respective tokens (see Appendix for how the key-term algorithm works). After pre-processing, tweets were filtered by using automated language detection so that only tweets considered English were kept, which also removed emoji-only tweets. Lastly, only remaining tweets containing 3 or more tokens were kept, leaving N = 37579.

Whilst previous research has utilised "few shot" prompting, which gives example responses to LLMs as additional information, I utilize algorithmically determined context embeddings which ensures that GPT has the required information to perform the classification task. This is most pertinent in the case of Twitter handles as it would be a lot to ask of GPT to correctly interpret the use of idiosyncratic twitter handles as references to individual IPEs.

Given the "blackbox" limitation of LLMs, the format of the prompt was given great attention since this is within the ability of the researcher to control. The prompt designs ensure that all distinct key term occurrences are shown to GPT without duplicating smaller key terms that appear in larger key terms (see Appendix section 1 for an explanation of the indexing logic which ensures this). For example, a tweet that references both "Biden" and "Joe Biden" will return the two terms as key terms, but a tweet that only references "Joe Biden" will only return "Joe Biden" as a key phrase. Similarly, a tweet which uses "#joebiden" and "#biden" will return the two hashtagged terms as key terms, but a tweet that only references "#joebiden" only returns "joebiden" as a key term.

For key term matching with IPEs as the target, each IPE was given a list of potential expressions that could be used to uniquely identify them such that the key terms could not be making reference to another IPE *but may still not be making reference to the IPE at hand*. Whilst a nickname list could have been compiled manually, using programming logic paired with GPT was considered a helpful contribution in reducing research costs and time for others wishing to conduct similar analyses on different sets of elites.

Thus, for each IPE, a key term sub-list is compiled considering the following factors: whether the IPE is a Member of Congress and if so whether they are a representative or senator, how many middle names they have, whether they have a surname unique to other IPEs and, if not, whether their surname is unique to their chamber if they are a Member of Congress. Note that not all Members of Congress have a unique surname. In these instances, a surname alone is not useful for establishing whether a tweet potentially references one specific individual. However, some Members of Congress don't have unique surnames, but their surname *is* unique to their *chamber*. In such instances, prefixes for 'senator'/'representative' paired with their surname can isolate a potential reference to only one IPE.

Specifically, a list is generated containing sub-lists where each sub-list contains tokenized expressions of key terms associated for a particular IPE. For all IPEs, their full name is appended as a tokenized expression to their respective sub-list because there are no IPEs from the time period who share the same full name. Additionally, in instances where an IPE has a middle name, the first name and surname are also appended to the IPE's sub-list as a key term expression because no IPEs shared the same permutation of first name and sur name. Additionally, if an IPE has a middle name, each middle name up until the surname constitutes a tokenized expression. This logic is adept to deal with compound surnames in the informal

setting of twitter given that the first word in a compound surname is treated as though it were the last middle name.

If an IPE's surname is unique, that is also appended as a sub-list. If an IPE is a Member of Congress and has a surname unique to their chamber, the following prefixes are placed before the surname to constitute two new key terms: 'representative' and 'rep' if they serve in the House of Representatives; 'senator' and 'sen' if they serve in the Senate. Joe Biden and Donald Trump are alternatively given the prefix – 'president.'

A hashtag key terms list is then derived from the key terms list by simply joining each string in a key terms sub-list. Hashtags use a separate list because to match hashtags, I perform a substring search within tokens (strings) containing hashtags, rather than matching sub-lists across the tokenized expression of a tweet. I do this to allow for the fact that "#bidenomics" contains "biden" and therefore contains a key term. A twitter handles list is compiled separately based on the data merged from Mills and the University of California San Diego but uses the same indexing as the other lists such that each index in the key terms list, hashtag key terms list, IPE name list and twitter handle list corresponds to a single IPE (see Appendix section 1.1 for a snippet of the IPE key terms list)

In the key term search algorithm (see Appendix section 1.3 for full code), for a tweet, each list is processed for a given IPE before compiling a finalised list of matched key terms, such that only distinct instances of key terms are returned without duplicates. For example, a tweet that uses "#bidenomics" and references "biden", simply returns "biden" (as "biden" is the key term contained in "#bidenomics").

All tweets that contained a surname of an IPE that were not matched based by the key term matching algorithm were manually checked and any matches were added accordingly. This approach was designed with a particular focus on future applications dealing with larger datasets, where researchers want to minimize the number of tweets that need to be manually checked. Where authors were identified as using a key term that only potentially referenced themselves, those matches were removed such that the tweets were not processed by humans or GPT for IPE sentiment classification with the target was the author. Such tweets could still be processed if they also contained other IPEs.

While matching terms associated with IPE and Party targets uses identical logic, key terms for the Parties were simply chosen based on an intuitive understanding of the different terms used to reference the partisan groups in US politics. Then, using the same logic as IPE matching, hashtag terms were derived from the original key term list and the twitter handles list is not derived from either but shares the same outer indexing.

A distinct benefit of utilising a context embedded LLM approach to determine whether tweets reference the partisan target is the potential efficacy of reducing both type1 and type2 error when working with large corpuses. In state-of-the art dictionary approaches for automated target matching, nick names and full names are required to match tokenized tweets to their relevant target (Wojcieszak et al., 2022; Yu et al. 2023). While the logic of these dictionary approaches will effectively minimize the number of false categorizations mentioning a target, the imposed requirements may smuggle bias into samples by not identifying cases where elites are referenced through their respective surnames only. Alternatively, key term 'bootstrapping' techniques, which expand dictionaries by iteratively matching highly co-occurring terms with those already in the dictionary, often require large corpuses to begin with in order to obtain relevant subsequent terms (see King et al., 2017; Waseem and Hovy, 2016; Zhang et al., 2018).

4.3 BWS Tuple Generation

Given the previously outlined methodological motivation of providing BWS methods for large t for future research, I develop a BWS tuple generation algorithm (see Appendix section 2.1 for code and explanations) that takes inputs t, b,

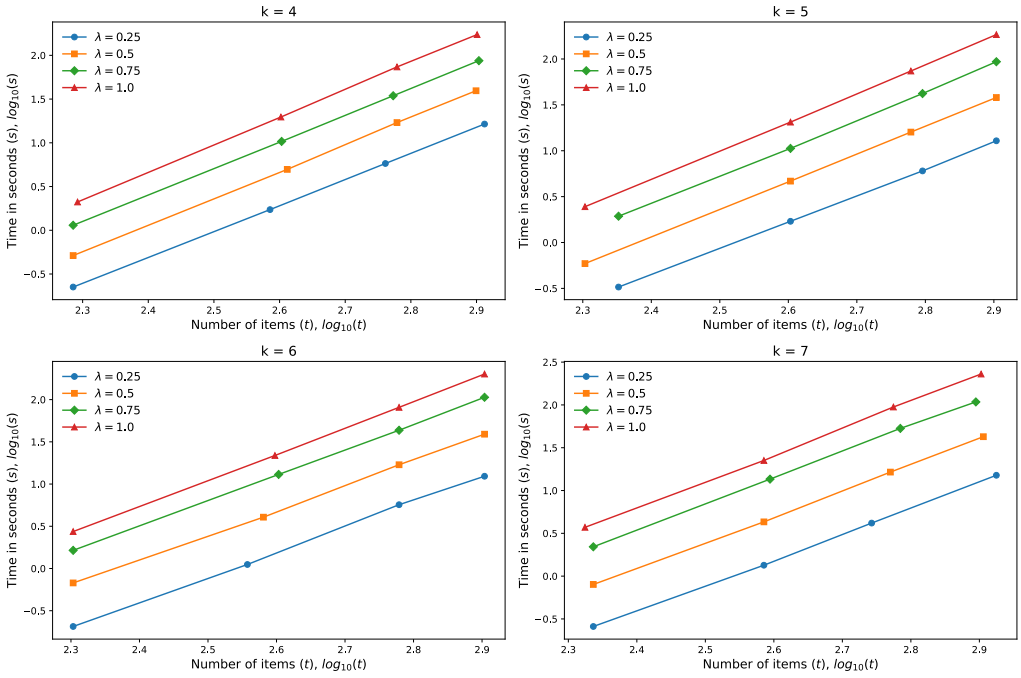

Figure 1. Timed Completions of BWS Algorithm

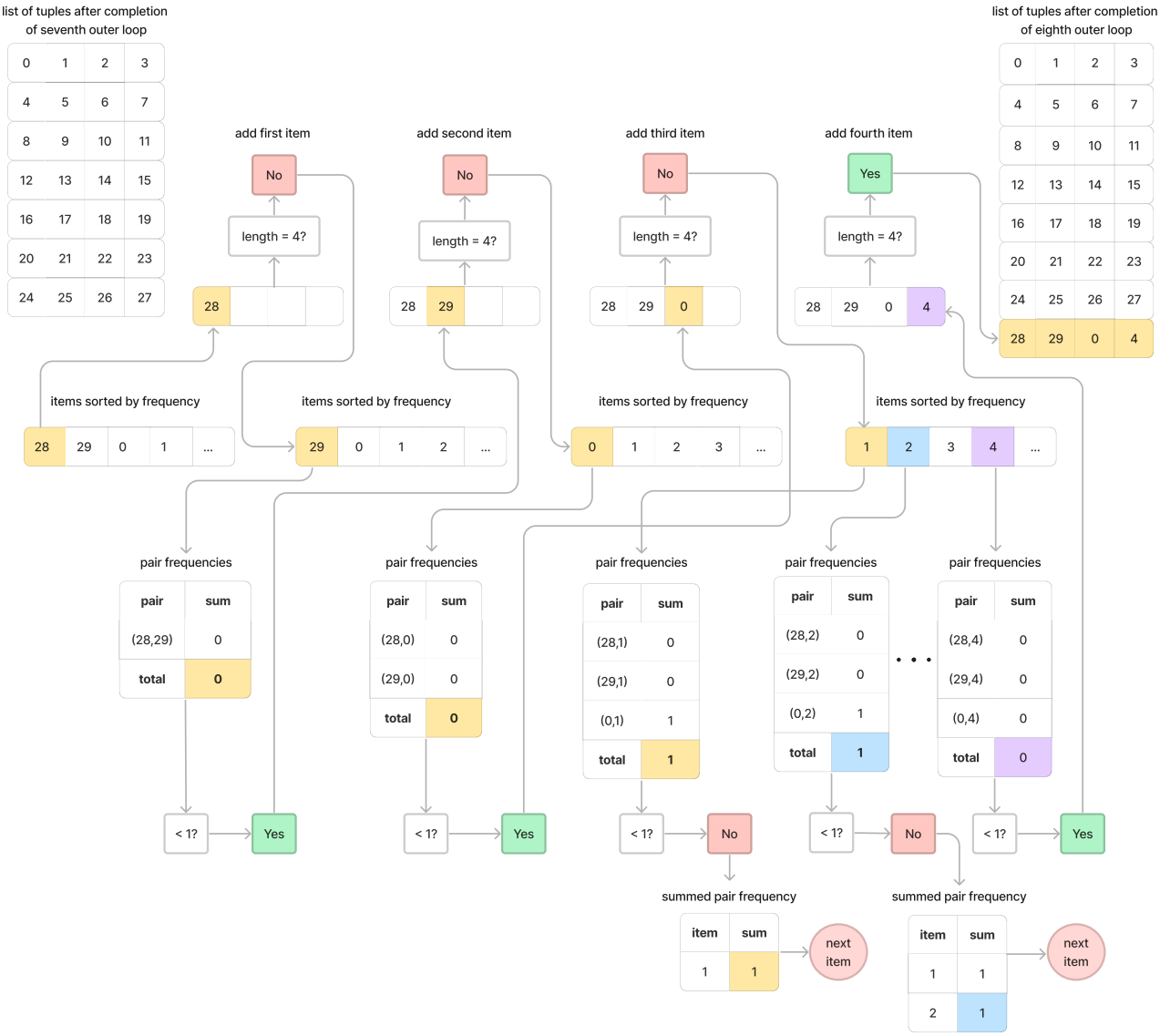

Figure 2. Visualisation of the BWS Algorithm with t = 30, k = 4, b = 60

and k to return tuples ensuring that no item appears more than once in a tuple and that no two tuples contain the same items (there are also additional parameters that can be used to decrease computational time at the expense of λ approximation). The algorithm uses a greedy logic to approximate r and λ by sorting available items and pairs according to the frequencies.

The key strength of the algorithm is that *not all possible pairs given t are considered;* instead the algorithm keeps track of positive pair occurrences and the item with lowest frequency across all tuples (blocks) is automatically added as the first item of a tuple (block) of size k. After the initial item is added, for each new sought item the existing pair frequencies of available items sorted by frequency are considered. If the total sum of pair frequencies of the considered item and items already in the tuple is $< \lambda$, the item is appended to the tuple. Each time an item is added, the availability of the item and its frequency is adjusted. Only after a tuple has reached size k are the pair frequencies adjusted (as items already in the tuple cannot be considered anyway). If no item with summed pair frequencies $< \lambda$ is available, the item with the smallest summed pair frequency is chosen. In the event of ties, the item with the smallest item frequency is chosen.

Whilst previous research has averred that $b = 2t$ (i.e 2N tuples) is generally sufficient (Kiritchenko and Mohammad, 2017; Mohammad 2018), the empirical approach of using simulations by Hollis (2018a, 2018b) underscore that for large t, larger b should be considered (Hollis 2018a, 2018b goes up to 32t). Whilst Hollis (2018a, 2018b, 2019) does not make use of λ, this value most likely underpins why Hollis suspect larger b is required.

Furthermore, studies have produced results evidencing that larger k coupled with scoring algorithms may be more appropriate when dealing with noisy and non-normal data because increasing the tie information generated per decision essentially linearizes the relationship between generated scores the true values (see Hollis 2019). Larger k values, however, should be exercised with caution when using count methods because the pairwise matchups increasingly contain extreme values with the potential to bias distributions toward the extremes. This is because larger k provides more total information about the items in a set per decision but less information about the rankings of

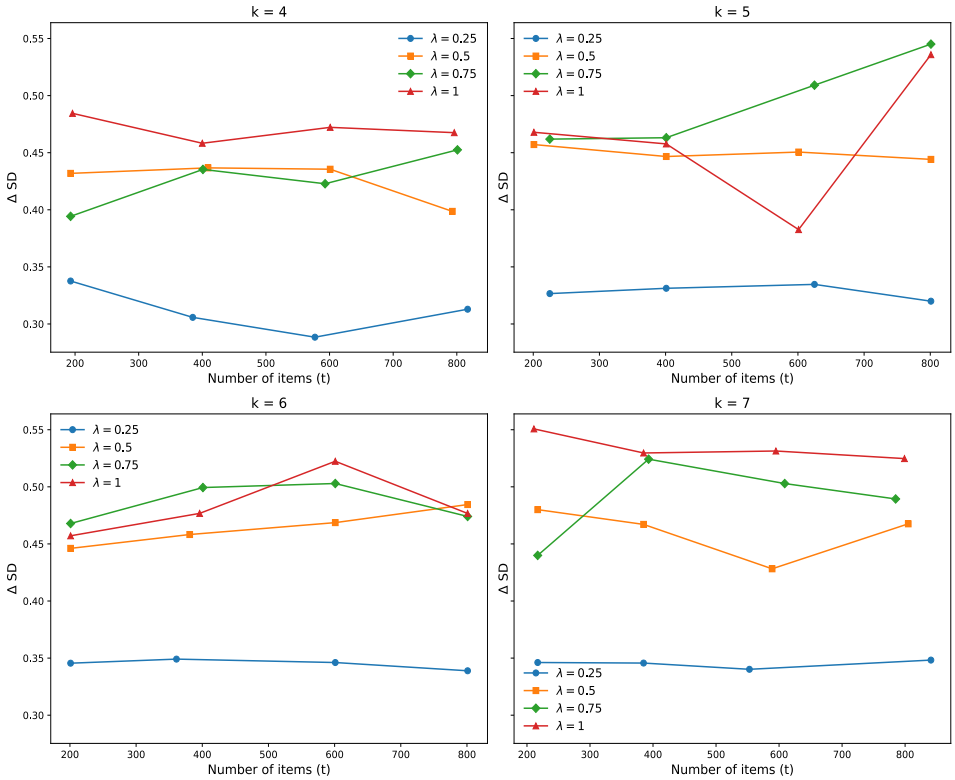

Figure 3. Item Balance of Tuples Generated with BWS Algorithm

Note: Y axis shows the difference in the standard deviations of item frequencies between generated designs and the theoretically maximal item-balanced designs. Given that t was chosen to ensure theoretical existence of perfectly item-balanced designs, the difference in the standard deviations are just the standard deviations of the generated designs.

items in a set as a proportion of the total pairwise combinations.

Due to the aforementioned dynamics – that increases in b given increases in t should be larger than derived increases from t multiplied by a predetermined constant, and that there a legitimate reasons to consider a range of k – I perform timed simulations of the tuple generation algorithm for a range of λ and k and increase t whilst holding λ and k constant in order to derive b. I find all designs for $150 < t < 850$ where an item-balanced [B]IBD exists (calculated by deriving the number of times an item occurs using equation 2). Subsequently, the total list of available [B]IBDs are then filtered so that t values around [200, 400, 600, 800] are chosen. I opt for this approach because it reflects how BWS blocks are most likely considered by researchers; an IBD where an item-balanced design exists would be logical to approximate given that increments in item-balance have evinced substantial gains in the quality of scores obtained (see Hollis 2018, Experiment 4).

Despite using greedy logic, it becomes clear that as t increases the time complexity increases by orders of magnitude, which is unsurprising; keeping λ and k constant for a given t requires that b increases non-linearly (see Appendix section 2.3 for all t, b and completion times for iterations given k and λ values). Nonetheless, the algorithm is able to return IBDs that are exceptionally approximate to the ideal most-possible balanced design (see Appendix section 2.3 for key metrics outlining the performance of all generated BWS designs).

Whilst it is beyond the scope of this research to empirically compare the performance of the algorithm to

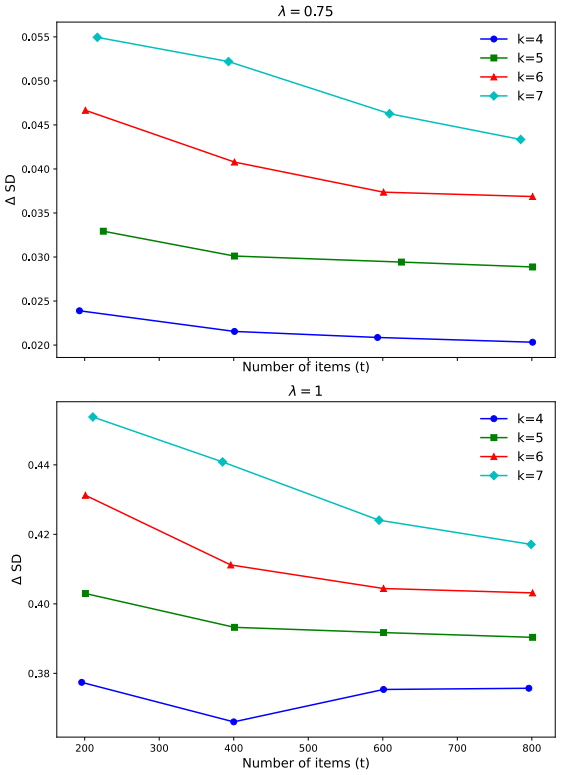

Figure 4. Pair Balance of Tuples Generated with BWS Algorithm

Notes: Y axis shows the differences in the standard deviation of pair frequencies between generated designs and the theoretically maximal pair-balanced designs. Only λ = (0.75, 1) is used because for λ = (0.25, 0.5) for each k, Δ SD = 0 for all t used.

others in the field, previous research would have potentially benefited from approximation toward both item and pair frequencies. For example, Hollis (2018a) only ensures item balance when using $t = 1000$ and $b = 32t$ due to the computational limitation of the randomised selection approach used for tuple generation in the study. In comparison, it is computationally feasible to implement my proposed algorithm without any adjustment of additional parameters that the algorithm takes. These gains become appealing given that in the same study, Hollis (2018a) also show that keeping pairwise combination unique across tuples generated better scores in both simulated and behavioural experiments.

4.4 BWS Scoring

Best-Worst Counting

There are a multitude of methods available for researchers to score BWS responses. The most simple and common is *best-worst counting* where the score of an item is calculated by subtracting the proportion of times it was chosen as worst from the proportion of times it was chosen as best (see equation 5). Whilst best-worst counting does not consider the degree of competition within sets, this shortcoming is mitigated in the classic BIBD design which is reasonable for small *t.* Whilst scores are not expected to effectively scale the true underlying variable given $\lambda < 1$ (see following section) , I use best-worst counting as it is the traditional option used for scoring items with BWS.

$$(5) \qquad BWC_i = \frac{N_i^{best}}{N_i} - \frac{N_i^{worst}}{N_i}$$

Analytic Best Worst

More sophisticated approaches use statistical modelling to score items, such as multinomial logistic regression and hierarchical Bayes. Due to computational efficiency, however, the analytical estimation for the MNL model (shown by Lipovestky and Conklin, 2014), referred to as ABW (for Analytic Best-Worst), is frequently used. Although ABW has the disadvantage of losing confidence intervals, scores have been shown to correlate highly with MNL coefficients (see Marley & Louviere, 2005). Given that a key methodological aim underpinning the methods deployed in this study is their future applicability on large samples, I use ABW (the closed-form solution) as a scoring method.

The ABW calculates utility coefficients (estimations of the MNL) by calculating one plus the best-worst count divided by one minus the best-worst count and then taking the logarithm of that fraction (see equation 6).

$$(6) \qquad ABW_i = \ln\frac{1+BWC_i}{1-BWC_i}$$

Value Learning

The final scoring method used is called *value learning* (Hollis, 2018), which differs from the previously introduced approaches in its algorithmic approach to scoring. Value learning was chosen over other scoring algorithms (such as Elo and Rescorla-Wagner) that have shown to better fit data with specific properties, due to its robust performance across different conditions (Hollis 2018). Specifically, value learning in simulated BWS trials (where items with known true values are sampled) has been the most robust scoring algorithm under conditions of noise for a range of distributions (see Hollis 2018 Fig 2, Table 1). In behavioural studies using BWS with human responses to items with known true values, scoring with value learning has also produced scores that most strongly correlate to the true values (see Hollis 2018, experiment 4; Hollis and Westbury 2018).

For value learning, the scores ("values") for items are iteratively updated weighted by the discrepancy in the observed outcome of a pairwise "match" and the expected outcome. The outcomes of pairwise matches are given by the inferred ranking derived from assigned "best" and "worst" items in the set. The item ranking higher in the pairwise match 'wins' and the other 'loses.' Crucially, unknown rankings within a set are not treated at draws but instead those matches do not occur. For example, for k = 4, given that 5 out of the possible 6 pairwise orderings are known, $5b$ pairwise matches are considered for value learning (where b is the number of sets) across all tuples. Given that each item's value within a pair is updated separately (see equations below), a complete iteration of value learning over all tuples generates $10b$ updates.

When assigning values to outcomes, only the relational difference between values is relevant. However, given only two outcomes are coded in value learning for BWS, the relational differences also are not relevant. For this reason (after Hollis 2018a), I give 'winning' a value of 1, and 'losing' a value of 0.

It should be noted that within the current operationalization it is theoretically possible that an item could be simultaneously chosen as the "most" and the "least" negative within a block which could problematize the pairwise matches scores for value learning. This is because two prompts are shown to GPT for a single block, asking GPT to choose the "most" and "least" negative tweet separately (see Appendix section 1.3.3 for BWS prompt examples). However, no attention was devoted to this feature because no incongruence was observed on this front.

An item's value, V_i, is updated after each match by weighting the observed discrepancy in the expected outcome of the match (which is simply the item's current value, V_i) and the actual outcome for i, γ_i, which takes values 1 or 0) by the multiplication of the salience α and learning rate β (see equation 9). The current value of an item V_i, considered as the probability that it wins the matchup, is converted to an odds value O_i (equation 7). Salience is calculated as 1 minus the odds of the winning item winning O_w, divided by the summed odds of for both the winning and losing items winning (see equation 8). The following equations show value learning in chronological order.

$$(7) \qquad O_i = \frac{V_i}{1-V_i}$$

$$(8) \qquad \alpha = \frac{1-O_w}{O_w + O_l}$$

$$(9) \qquad V_i = V_i + \alpha\beta(\gamma_i - V_i)$$

Items start with a value of 0.5 and in cases where neither item in a match has played before, salience is set to 0.5 (otherwise there is no update for the items given that α

would be 0). 25 iterations were run for all applications of value learning, with randomized order each time (given that order matters in value learning); An initial learning rate of β =0.025 was used. However, after Hollis (2018a), the learning rate, β, was divided by the iteration number for each subsequent iteration (see Raschka, 2015 for evidence supporting a reduction in β over epochs).

Log-odds Transformations

For both value learning and best-worst counting, Hollis (2018a) has shown that the log-odds transformation of the scores provide higher correlations with the underling variable than the raw scores using those methods. Given that ABW already uses a log-ratio transformation to score items, I also (like Hollis 2018a) maintain the untransformed scores for ABW. The log-odds transformation takes the natural logarithm of the odds ratio of an item score, which can also be termed the logit transformation: $\text{logit}(S_i) = \ln(S_i/1 - S_i)$.

4.5 Tasks

GPT models are trialled across 4 tasks. Despite the abundance of research demonstrating the efficacy of GPT-4, there is evidence it has been out-performed by other GPT models on sentiment-related tasks (see Rathje et al. 2023 Fig 1). Thus, I also use text-davinci-003, a legacy *instruct* model of GPT 3.5, which was created specifically for better instruction compliance (it is generally less "chatty"). For every task, the applied GPT models use a temperature of 0, which ensures that the highest probability tokens are chosen as a response for each prompt. In different semantics, if the prompts were reapplied to the same tweets, the answers would be the same (unless the model was updated between the implementations by OpenAI). Choosing a temperature of 0 is standard practice for GPT classification particularly when GPT models are not intended to be reiterated over data.

Tasks 1 and 2 are use GPT for mixed ordinal and nominal classification of partisan sentiment toward both the in-partisan and out-partisan group; Task 1 assessed sentiment classification with IPEs as the target; Task 2 assessed classification with the overall party as the target. In both tasks, GPT models and human annotators are asked to classify a tweet as mentioning a specified target, and if so, to classify the tweet as positive, negative, or neutral toward the specified in-partisan/out-partisan target. Task 1 and 2 were completed by GPT-4 and text-davinci-003, however two types of prompts were trialled (see Appendix section 1.3.1) in Task 1. As the performance for Prompt Type 2 was significantly better for Task 1, only Prompt Type 2 was utilised thereafter for all subsequent tasks. In Prompt Version 1, contextual information such as the 'potentially referenced' IPE's twitter handle was given to GPT-4 and text-davinci-003 below the tweet whereas in Prompt Version 2 contextual information was embedded in the main text of the prompt (see Appendix section 1.3.1).

For each of the two classification tasks, respondents and GPT were presented the same set of 200 tweets. 3 annotators labelled the tweets to generate a "gold standard" for each task, which were tweets where at least 2 out of the 3 annotators agreed, which yielded a sample size of n=189 for Task 1 and n=199 for Task 2. Tweets shown to GPT and annotators for Task 1 and 2 were those that were algorithmically detected as containing a key term associated with the out-partisan or in-partisan group (as described in section 4.2 and Appendix section 1.1). Within the tasks, tweets which mention more than one IPE or both parties were allowed to appear twice. Additionally, tweets could appear across tasks if they mentioned both an IPE and political party.

Task 3 and 4 utilised Best-Worst Scaling with GPT to measure in-partisan positivity and out-party negativity respectively. The tweets for Task 3 and 4 were selected by implementing GPT-4 with Prompt Version 2 from Task 2 over the whole dataset and randomly sampling 50 tweets classified as positive (from a potential N = 1085) for Task 3 and 50 tweets classified as negative (from a potential N=2453) for Task 4. Only the party as a target was considered due to financial limitations, however combining the logic used from Task 1 and Task 3/4 to apply BWS for measuring in-partisan positivity/negativity toward IPEs would be completely feasible.

For the BWS tasks (3 and 4), tweets from both Republicans and Democrats can be show in a single item. For scaling in-party positivity (Task 3), the "most positive" and the "least positive" tweet towards the authors own party are selected; for scaling out-party negativity (Task 4), the "most negative" and the "least negative" tweet towards the authors own party are selected.

For Task 3, tweets were shown to text-davinci-003, GPT-4, and 2 annotators in the form of 2N 4-tuples (i.e $b = 2t$, $k = 4$), which is a standard BWS design. Text-davinci-003 and GPT-4 were assessed against the mean of the two annotators' scores. Task 4 repeats the design of Task 3, however, additionally uses GPT-4 with 3N 4-tuples (i.e $b = 3t$, $k = 4$). Given that $t = 50$ items were used for each BWS task, k=4 with $b = 2t$ gives b=100 and $\lambda = 0.49$ (3 s.f); k=4 with $b = 3t$ gives b=150 and $\lambda = 0.73$ (3 s.f). Best-Worst Counting (BWC), Analytic Best-Worst (ABW), and value Learning are used to score all tweets for tasks 3 and 4 with additional logit transformations of BWC and value scores.

Whilst I use IBDs with $\lambda < 1$ for a relatively small t where BIBDs were feasible, the methodological objectives were to establish methods to measure partisan sentiment when dealing with large samples, which therefore made it preferable to use λ values that were more likely to be paired with algorithmic scoring in such cases.

Using $\lambda < 1$ was also not expected to compromise the quality of scores given that previous research has been able to achieve accurate BWS scores with approximate and lower λ values (see Hollis 2018a where k=4, $t = 1000$ and simulations using $b = 32t$ give the maximum λ of 0.38 3.sf and scores achieving $R^2 > 0.95$; see Hollis 2019 which also shows reasonable performance with $k = 8$, $t = 840$, $b = 8t$ giving $\lambda = 0.53$, 3.sf). Furthermore, $b = 2t$ has also been shown to be sufficient for a large t ($t > 1000$). For example, Kiritchenko and Mohammad (2017) show the split half reliability for different b values (number of tuples) on 3207 items (t= *3207)* using 4-tuples (k=4) and show that performance increases start to notably diminish at around $b = 2t$.

I implement the developed algorithm for BWS tuple generation which approximates item and pair balance to r and λ using a greedy logic (see section 3.3 for explanation of algorithm and Appendix section 2.1 for code). Whilst alternative algorithms could have ensured optimal item and

pair balance toward r and λ, this study was considered a good test case for generating BWS scores with suboptimal item and pair balance. Firstly, suboptimal item and pair frequencies become more realistic when using BWS with larger t; secondly, the developed algorithm shows evidence that deviation from r and λ is in fact slightly higher with *smaller* t but nonetheless stabilizes (see figure 3 and 4). Despite higher deviations with smaller t, item and pair frequencies in generated tuples for human and GPT annotations are still very proximate to optimal values (see Appendix section 2.2 for item and pair frequencies)

I use 4-tuples ($k = 4$), which are typical in BWS, as they are particularly efficient in giving pair rankings and, as previously noted, appropriate for both counting and algorithmic scoring methods. In a given set of items A, B, C, D, given that A is chosen as the "best" and D the "worst", we are able to establish 5 out of the possible 6 pair rankings: $A > B$, $A > C$, $A > D$, $B > D$, $C > D$.

5 RESULTS

For Tasks 1 and 2, annotators displayed good agreement (see Table 1). I use three metrics, all of which capture inter-annotator reliability in slightly different ways. I use Fleiss's Kappa treating the 4 categories ("positive", "negative", "neutral" and "doesn't mention") as nominal variables. I additionally compute Fleiss's Kappa just for the classification of whether a tweet references the partisan target in question (a binary classification). Lastly, I compute Krippendorff's alpha with equal relative distances between 'negative', 'neutral' and 'positive' (numerically converted to 1,2,3 respectively) by substituting categorizations of "doesn't mention" with missing values, which Krippendorff's alpha is apt to handle. Krippendorff's alpha takes into account the closeness of ratings, which is a useful property when working with ordinal data. Unsurprisingly, the inter-rater reliability metrics point to greater agreement about whether a tweet actually mentioned the target of interest which provides a good benchmark with which to test GPT binary classification for target occurrence within tweets (see Table 1).

TABLE 1
Classification Inter-Rater Reliability (Task 1 and 2)

Method	Party	IPE
Fleiss's Kappa all 4 categories	0.72	0.73
Fleiss's Kappa binary (mention/doesn't mention)	0.92	0.91
Krippendorff's alpha (negative, neutral, positive)	0.81	0.83

Notes: All numbers rounded to 2 significant figures..
Number of annotators: 3

While annotators generally displayed solid agreement for tasks 1 and 2 (see Table 1), the agreement between annotators for BWS, measured as the Pearson correlation coefficient of their scores, was generally poor and notably conditioned on the task. In particular, the correlations for BWS on Task 3 – measuring in-party positivity are notably low (see Table 2). The spearman rank underscores the discrepancy between tasks. Spearman rank is a useful indicator of inter-rater agreement for BWS within this context because it is insensitive to transformations of the raw scores. As value learning can change the rank ordering of items, only two spearman rank tests are required for each task. For in-party positivity, p values were > 0.01 for non-algorithmic scoring methods but ≤ 0.01 for value learning, whilst for out-party negativity all scoring methods had p-values ≤ 0.01 (See Table 2).

TABLE 2
Correlation (r) between annotator BWS scores

Score	In-party positivity	Out-party negativity
BWC	0.34	0.50
ABW	0.19	0.41
Logit (BWC)	0.20	0.42
Value	0.37	0.48
Logit (Value)	0.38	0.49

Notes: All numbers rounded to 2 significant figures.
Number of annotators: 2
Spearman rank: for in-party positivity all p values > 0.01 other than value scores; for out-party negativity all p-values < 0.01

The aforementioned metrics for inter-annotators agreement on BWS tasks are first and foremost indicative of a high level of *difficulty*. Interpretation of GPT performance for Tasks 3 and 4 should therefore be seen in light of the fact that human annotators themselves disagreed considerably. Although ordinal classification forces annotators to place items on a limited scale, BWS forces annotators to rank items when it is perhaps difficult to draw meaningful distinctions. It is plausible that in this instance *ordinally ranking* items within the used operationalization was intrinsically difficult, introducing high levels of noise to judgements. Noise is present within annotator judgements when the observed rank ordering of items in blocks stands in contradiction to their latent value, which is a realistic feature of BWS annotations.

Explaining low annotator agreement in terms of task difficulty may explain why correlations between GPT and annotators' means for out-party negativity are perhaps much higher than one would expect when taking the inter-annotator agreement at face value (see Table 5). Firstly, noise in annotator judgements does not necessarily compromise the estimation of the latent values; in fact, noise has conversely been shown to improve estimations when scoring items algorithmically (see Hollis 2018 experiment 2). Secondly, taking the mean for human annotations likely improves score accuracy in a similar way to increasing b in the presence of noise. In simulations by Hollis (2018), value estimation for t items is shown to become more conditioned on larger b in the presence of noise. Whilst splitting blocks across annotators, such that annotators see different tuples, likely better estimates the underlying latent value, there were overriding concerns about whether correlating scores for annotators who had seen different tuples could be considered a fair test for gauging inter-rater agreement.

TABLE 3
IPE-targeted Sentiment Classification Performance (Task 1)

	Text-davinci-003						GPT-4		
	Prompt Version 1			Prompt Version 2			Prompt Version 2		
Class	precision	recall	f1	precision	recall	f1	precision	recall	f1
Negative	0.92	0.67	0.78	**0.94**	0.88	0.91	0.84	**1.00**	**0.91**
Neutral	0.17	0.67	0.27	0.27	**0.71**	**0.39**	**0.75**	0.12	0.21
Positive	**0.87**	0.41	0.55	0.86	0.59	0.70	0.83	**0.94**	**0.88**
Doesn't mention	0.77	0.30	0.44	**0.97**	0.54	0.69	0.91	**0.95**	**0.93**
macro avg	0.68	0.51	0.51	0.76	0.68	0.67	**0.83**	**0.75**	**0.73**
weighted avg	0.77	0.51	0.57	0.85	0.70	0.74	**0.85**	**0.86**	**0.82**

Notes: All numbers rounded to 2 significant figures.
N= 186. The highest score for each metric for each class is put in bold.
Macro-Averaged Mean Absolute Error:Text-davinci-003 prompt version 1: 0.37; prompt version 2: 0.27; GPT-4: 0.29

Turning to GPT performance, results from Task 1 (see Table 3) which utilised two prompt designs showed overwhelming support for further use of Prompt Version 2 (see Appendix section 1.3.1 for example prompts). When comparing each precision, recall, or f1 for a particular class (negative, neutral, positive), text-davinci-003 with Prompt Version 2 outperformed GPT with Prompt Version 1 in every case except for precision of the positive class where the difference is marginal (0.87 over 0.86). Due to the comprehensiveness of Prompt Version 2's superiority over Prompt Version 1, Task 2 only used Prompt Version 2. Unfortunately, the structure of Prompt Version 2 could not be replicated within a BWS framework given that multiple tweets are shown in a single prompt. Thus, akin to Prompt Version 1, contextual information is presented below the tweet for BWS tasks (see Appendix section 1.3.3 for a relevant prompt example).

GPT-4 overwhelmingly achieves the best performance for tasks 1 and 2 (see tables 3 and 4). The difference in the macro averages and weighted averages for precision, recall, and f1 highlight its general superiority to text-da-vinci-003. Additionally, when focussing on precision, recall and f1 of specific classes, GPT-4 demonstrates better classification performance with the notable exception of the neutral class; both GPT models demonstrate a poor ability to correctly classify tweets as neutral, however GPT-4 neutral classification is comparatively worse than that of text-davinci-003 and markedly *conservative*. GPT-4 demonstrates good (sometimes perfect) precision, but very poor recall of the neutral class; in other words, GPT-4 hardly ever classifies tweets as neutral when it should, but when it does it is usually right (see Appendix section 3 for relevant confusion matrices).

This slightly unusual result is congruent with the results of previous studies. For example, Rathje et al. (2023), find that text-davinci-003 out-performs GPT-4 for overall sentiment classification due to the same issue of over-conservative neutral classification. GPT-4's poor neutral recall notably drags down the macro average recall, which is a useful statistic due to its robustness to class imbalance and invariance to order (in contrast to F1 which changes when categories are switched –Macro Average Recall is just the sum of each class recall divided by the number of classes).

In a similar vein to using Krippendorff's alpha for inter-coder reliability among the annotators for tasks 1 and 2, I additionally use the macro-averaged mean absolute error (MA-MAE) computing missing values for "doesn't mention" classifications. MA-MAE similarly computes positive and negative disagreements as further apart than positive (or negative) and neutral and is robust to class imbalance. For Task 1 (with the target being an IPE) the MA-MAE for text-davinci-003 prompt version 1 was 0.37, which was the highest. The second highest was GPT-4, 0.29, which again is partly explained by its low neutral recall. Text-davinci-003 obtained the lowest with a score of 0.27. Similarly, for Task 2 with the overall party as the target of partisan sentiment, text-davinci-003 has the lower score (0.3) compared to GPT-4 (0.32).

These results also serve to highlight the notable difference between GPT-4 and text-davinci-003 in classifying tweets as mentioning the target in question. GPT-4's superior macro-average and weighted-average classifications for tasks 1 and 2 is greatly influenced by its stark superiority in this regard (see Table 3 and 4). In a hypothetical operationalization where we definitely knew if

TABLE 4
Party-targeted Sentiment Classification Performance (Task 2)

	text-davinci-003			GPT-4		
Class	precision	recall	f1	precision	recall	f1
Negative	0.85	0.95	0.90	**0.85**	**1.00**	**0.92**
Neutral	0.45	**0.44**	**0.44**	**1.00**	0.05	0.10
Positive	**0.78**	0.75	0.77	0.73	**1.00**	**0.84**
Doesn't mention	**1.00**	0.12	0.22	**1.00**	**0.89**	**0.94**
Macro avg	0.77	0.57	0.58	**0.90**	**0.73**	**0.70**
Weighted avg	0.75	0.75	0.73	**0.85**	**0.80**	**0.73**

Notes: All numbers rounded to 2 significant figures.
N= 199. The highest score for each metric for each class is put in bold.
Macro-Averaged Mean Absolute Error:
Text-Davinci-003: 0.30; GPT-4: 0.3

a tweet mentioned an IPE for all tweets in a sample, it is arguable that text-davinci-003 provides a better choice of classifier. Nonetheless, GPT-4's proficiency in correctly classifying tweets as mentioning/not mentioning targets could prove very useful in future research utilising GPT for target-based classification tasks.

As touched upon previously (see section 4.2), GPT as an alternative to keyword-based approaches is a welcome addition to the social scientist's toolbox. Firstly, without access to large relevant corpuses, keyword methods can become hindered. Secondly, the potential for Type 1 and Type 2 error can make it incumbent upon researchers to manually check a large portion of collated samples, which stands in contrast to GPT's potential to expansively automate matches.

Turning to GPT performance on BWS tasks, multiple patterns can similarly be drawn from the results (see Table 5). GPT-4's much-speculated improvement on legacy versions is illustrated when looking specifically at out-party negativity (Task 4). For each scoring method, GPT-4 produced higher correlations with annotator means. However, bizarrely, the opposite is true for in-party positivity, where GPT-4 produced lower correlations with annotator means for each scoring method. Nonetheless, it should be stressed that there are marked differences in the overall performance for out-party negativity and in-party positivity.

Given that we not have access to reliable estimates of the underlying latent variables in question, we cannot locate the causes of the poor correlations for in-party positivity with certainty (see section 7.1 for a discussion on difficulties relating to latent variables). However, one could reasonably speculate that if ordinal decisions over items for in-party positivity are more "difficult", the judgements would become noisier, potentially necessitating a comparatively larger *b* and *k* for valid estimates of the latent values. These parameters have been shown to affect the accuracy of estimates of underlying values in the presence of noise within judgements (see Hollis 2018; Hollis and Westbury 2018). This explanation would be congruent with the previously noted finding that inter-rater correlations for in-party positivity are much lower than for out-party negativity.

Another nuanced question on the choice of parameters for BWS also emerges from the finding that greater *b* for GPT consistently improved correlations for different scoring methods when scaling out-party negativity (see Table 5). Future research on BWS (which I discuss further in section 7.1) should attempt to uncover the relationship between parameter values used to obtain accurate score estimates from human judgements and those obtained using GPT. Such research would likely be intrinsically context-dependent and could therefore attempt to decipher these relationships within the current research context.

Whilst the raw BWS scores provided comparable inter-annotator agreement with value learning, I use the logit transformation of the value scores in final exploratory findings as the logit transformation of value scores for out-party negativity objectively provided higher correlations (see Table 5). Given the working assumption that we have given annotators and GPT a rather difficult task, it is not unsurprising that there is supporting evidence for value learning. Value learning has evinced efficacy when annotator judgements are noisy compared to traditional scoring methods (see Hollis 2018; Hollis and Westbury 2018). The robustness of value learning in this regard is unsurprising given that it is able to take into consideration the competition between items within tuples. Generally, GPT and the inter-annotator results underscore the importance of scoring methodology within BWS given the wide range of correlations displayed conditioned on the scoring method and transformations.

Although it is beyond the remit of this study to uncover the explanations behind these results, certain findings are

TABLE 5
Correlation (*r*) Between Humans and GPT for Party Sentiment Classification (tasks 3 and 4)

	In-party positivity		Out-party negativity		
Score	Text-davinci-003 ($b = 2t$)	GPT-4 ($b = 2t$)	Text-davinci-003 ($b = 2t$)	GPT-4 ($b = 2t$)	GPT-4 ($b = 3t$)
BWC	0.42***	0.41***	0.42***	0.67***	0.68***
ABW	0.21	0.18	0.49***	0.5***	0.60***
Logit (BWC)	0.23	0.20	0.49***	0.51***	0.60***
Value	0.46***	0.44***	0.42***	0.67***	0.69***
Logit (Value)	**0.46***	0.44***	0.42***	0.68***	**0.70***

Notes: All numbers rounded to 2 significant figures.
The combination of model and scoring method with the highest correlation for in-party positivity and out-party negativity is put in bold.
*Asterisks indicate statistical significance for Pearson correlation: p≤0.001: ***, p≤0.01: **, p≤0.05:**

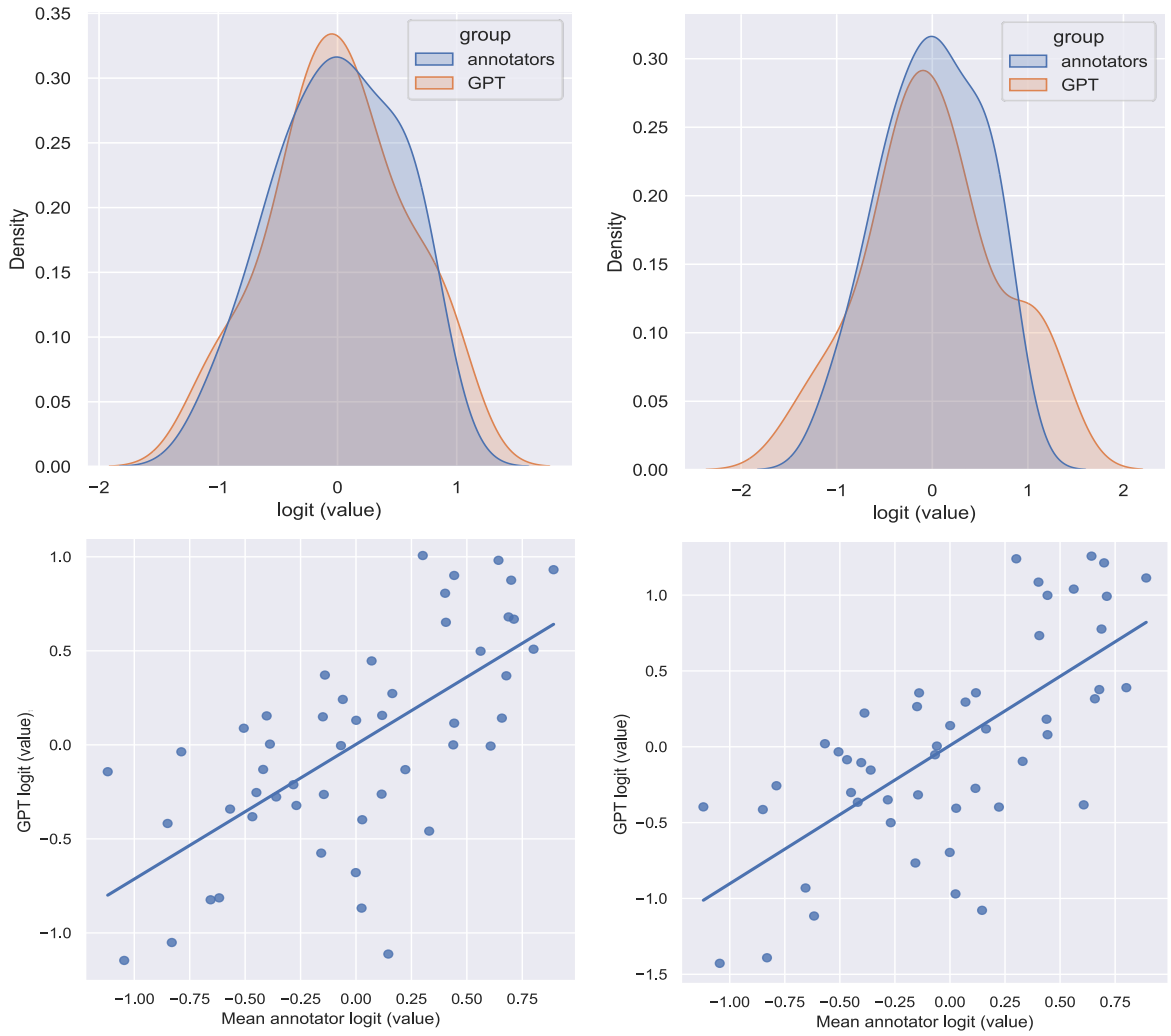

Figure 5. GPT-4 and mean annotator out-party negativity score densities (above) and correlations (below) for $b = 2t$ (left) and $b = 3t$ (right)

TABLE 6
Out-party Negativity Ordinal Classification Performance

Class	precision	recall	f1
Slightly negative	0.74	0.80	0.77
Very negative	0.78	0.72	0.75
Macro avg	0.76	0.76	0.76
Weighted avg	0.76	0.76	0.76

Notes: All numbers rounded to 2 significant figures.

amenable to initial exploratory hypothesising. With regards to the marginal improvement of GPT-4 for out-party negativity with greater *b*, one could proffer, based on figures 5 and 6, that this is a consequence of the greater number of updates (matches between pairs) helping to extremify the scores of items that were initially distant from the mean, making the other items comparatively closer to the mean (normalization of the scores would have directly demonstrated this effect but it can still be observed in Figure 6). Indeed, Figure 5 shows that value learning with greater

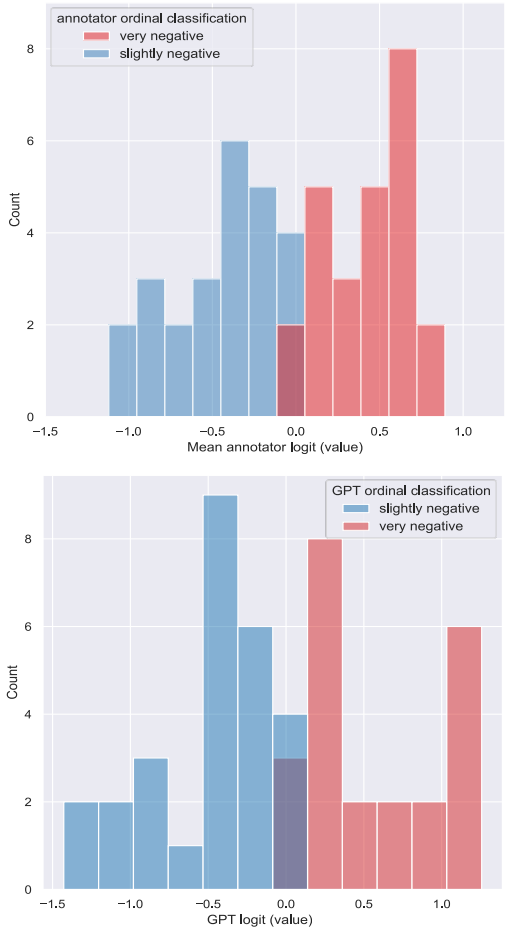

Figure 6. Histogram of mean annotator and GPT-4 out-party negativity scores grouped by median-split binary classification

b essentially "stretches" the distribution (increased the standard deviation) whilst Figure 6 shows that GPT scores become more correlated with annotators' around the annotator mean when using $b = 3t$ (i.e 3N tuples).

For more exploratory purposes, I lastly convert the logit transformation of value learning scores to a binary classification of out-partisan negativity. I split the scale at the annotator median to produce a binary classification of "slightly negative" and "very negative." The results (see Table 6 and Appendix section 3 Table A14) show promise in splitting BWS scales given the solid classification performance obtained for the two classes.

It should be emphasized, however, that it is not entirely clear how one should interpret the classes obtained from the splits to the BWS scale despite the wording of the class labels used. Future research may want to explore how splits to scaled variables compare to ordinal categories known ahead of time. Nonetheless, even if one were to have both ordinal and BWS measurements for the same items such that the scores could be compared, it would still not be clear how we should assess the performance of one method against the other given that both methods are using fundamentally different measurement techniques (see section 7.1 for a related discussion on the difficulty of variable latency).

6 Discussion

On a range of the selection tasks, GPT-4 most notably proved to be effective in measuring partisan sentiment. Nonetheless, these results, although garnered from random samples with low counts of missing values, cannot be treated with the same level of confidence as large, systematic designs. These results are merely suggestive of which types of GPT models, scoring methods, and prompts are more promising.

There are many limitations of the study that pertain to practical issues that future studies with funding would sensibly rectify. Firstly, the sample size should be greatly increased. Previous research that has applied human annotations for ML classifiers of partisan sentiment on twitter have gained annotations as large as N=10000 (see Yu et al., 2023; see also Wojcieszak et al., 2022, who use N = 8351). Secondly, the quality of the annotations could be likely be improved by implementing standard social science approaches to obtaining high quality annotations. A more rigorous process of obtaining annotations through a survey platform could utilize built-in software tools to notify annotators when make "incorrect" annotations. For example, Mohamed et al. (2018) interspersed "gold standard" tweets in a classification exercise and have immediate feedback to crowd workers when they got question wrong. Additionally, workers whose accuracy fell below 70% were removed from further participation. Hollis and Westbury (2018), who obtain annotations with BWS, use a similar approach by randomly placing 'test trials' in annotators' tasks. If workers' accuracy on the gold-standard sets falls below 70%, they are flagged as noncompliant and, akin to Mohammed (2018), prevented from further participating.

In addition to these standard procedures, future research may want to go beyond convention by devoting more resources in explaining the nature of the task to respondents. Annotating tweets for partisan sentiment frequently require knowledge of the US political context. The use of example

tweets with annotations and explanations, should therefore likely be more extensive than when using BWS for term valence (see Hollis and Westbury 2018) as a point of comparison.

Additionally, it was beyond the scope of this study to systematically trial the provided BWS tuple generation algorithm with others in the field. However future research may want to focus on this area alone. Aside from comparatively testing the algorithm's performance, others may want to alter the logic or incorporate/combine logics of different algorithms. For example, one could incorporate a random sampling approach for adding *k* items to a tuple at once, and then the greedy algorithm provided could be used only if the initial tuple is invalid (which will become more likely as more tuples are generated). This "initial shot in the dark" approach could also be logically bounded by relevant parameters, such that more random guesses are used, for example, as less items are available or as the proportion of tuples required decreases (or vice versa – more shots in the dark are used when it is more likely to work).

In a similar vein, the thoroughness of this study could be greatly improved if the asymmetry of the tasks were rectified. Specifically, two potential BWS tasks were omitted due to financial limitation. However, future studies could easily implement BWS for partisan sentiment with IPEs as the target, based on the operationalisation of Task 1 and Task 3. With regards to the prompts themselves, prompts should similarly be more systematically trialled in future work (again this was asymmetrically trailed in Tasks 1 and 2 owed to financial limitations). Relatedly, trialling a greater number of prompt designs would also offer valuable insights particularly when considering that only one design was used for BWS and that GPT has thus far not intersected with BWS.

In particular, future research implementing GPT for BWS should attempt to implement the embedded context approach used in Prompt Version 2 in tasks 1 and 2. With this approach, the main text of prompts would explicitly say who the target is for each tweet. Whilst it is more intuitive to put such contextual information beside individual tweets when humans make annotations, GPT may produce better responses with different prompt designs.

Alternatively, future work could instead opt for few-shot prompting or go even further with fine-tuning when using GPT for variable measurement. Whilst both approaches present promising directions for improving performance, they both decrease scalability in various ways. Few-shot prompting increases the token count of prompts, which will ultimately make each prompt more costly. Additionally, fine-tuning requires an annotated sample to begin with which can be time-consuming and/or expensive to obtain.

Therefore, there is also potential to simplify prompts and test whether contextual information is necessary at all. This seems a more feasible option for party-based sentiment classification as there are a relatively small number of terms that can be used to address the parties. Nonetheless, recent updates in GPT capabilities now make providing less contextual information more feasible. Specifically, the recent upgrade in GPT internet connectivity may drastically reduce the need for context-heavy prompts such as those used in this study. Relatedly, future research seeking to carry out similar work should consider keeping URLs in tweets when using GPT with browsing capabilities. Also, improvements in GPT capabilities should encourage future implementations of GPT for BWS to trial prompts that extract both "the most" and "the least" for a block of tweets simultaneously (which is how human annotators typically annotate BWS blocks).

Despite there being an array of improvements, which stem from various shortcomings in the thoroughness of the research, there are also potential conceptual deficiencies that warrant attention. One potential flaw in the operationalisation of partisan sentiment is that the keyword-based approach does not allow for capturing partisan sentiment expressed in policy agreements/disagreements because only tweets which contain partisan language are considered as potentially expressing partisan sentiment; however, this begs the question as to whether openly supporting or critiquing a policy constitutes an expression of sentiment toward the partisan entities responsible for the policy.

Conceptually, it does seem reasonable within a sentiment framework to say that an expressed opinion toward a policy without any partisan reference is not actually an expression of sentiment toward the responsible partisan entity. One could *assume* the author's partisan sentiment from such as expression, however it is not in fact evident. This conceptual detail is related to a fundamental difference between stance and sentiment. Mohammed et al. (2017) underscore that a tweet can express favourability/unfavorability (stance) toward something without directly referencing it. For example, they aver that openly supporting one candidate for an election expresses a stance on other candidates. Therefore, conceptual inadequacies within the deployed operationalisation seem more likely to stem from opting for a sentiment-based framework over a stance-based framework rather than there being incoherencies within the sentiment framework used.

Future research could easily adapt the methods used here for a stance conceptualisation of partisan expression where tweets are classified as 'favourable', 'unfavourable', or 'neutral.' BWS could then be used to scale out-partisan unfavorability and in-party favourability. However, with a stance framework, keyword methods would likely prove problematic and all tweets would require annotation. Additionally, formatting would likely require changing to ensure that GPT is provided information pertaining to how the target entity does not have to be referenced in order for an unfavourable stance to be expressed, which may require few-shot prompting or fine-tuning.

With regards to conceptual issues specifically relating to BWS, the operationalisation of in-party positivity and out-party negativity for partisan sentiment raises the important question of how appropriate the use of separate scales is considering that elites can be negative toward their in-partisans and positive toward their out-partisans (which is hereafter termed *partisan sentiment incongruence*). These conceptual concerns are, however, partly mitigated by the fact that partisan sentiment incongruence is empirically rare; no such instances of partisan incongruence were found by either humans or GPT in this study.

Additionally, whilst tweets classified as 'neutral' make up a considerable portion of both in-partisan and out-partisan tweets (although there are more neutral classifications for in-partisan tweets), concerns around using BWS for items originally classified inaccurately are

also partly mitigated by the ability to extract outliers with BWS; as noted earlier in the results section, outliers on the "least" ends of the distribution for both tasks were found to correspond to tweets that are actually neutral toward the partisan target. Whilst this is demonstrable that BWS can find initially incorrect classifications, it does however undermine the accuracy of rest of the data and raises concerns that BWS should be re-implemented with the exclusion of the outliers.

Unfortunately, as with many other methodological concerns that have been raised thus far, it is not entirely clear how the inclusion of items that do not possess the latent variable being scaled impacts the scores of BWS. Part of the inability to answer these uncertainties within the current study stems from the fact that we do not have access to the underlying latent values. Relatedly, high correlations between GPT and humans are not evidence of high correlations between GPT and the underlying latent value, although (as will be discussed in the following section), I have used this heuristic as a proxy for practical purposes throughout. There is, therefore, ground to say that methodologically the cart has been put before the horse – and that without access to these underlying latent values obtained through many high-quality annotations, we simply cannot answer many important questions.

Consequently, in the proceeding section I consider how future research, where item values are known ahead of time can offer valuable insights into answering a host of uncertainties with regards to BWS (many of which I have already alluded to).

7 Future Directions

7.1 Best-Worst Scaling

Future research exploring GPT for BWS specifically for partisan sentiment could try and approximate the underlying latent values by obtaining many high-quality annotations. Whilst this would require significant time and cost, we would be able to directly explore multiple research questions using such an approach.

Firstly, future research trying to approximate the underlying latent values, could implement multiple operationalizations of partisan sentiment. One alternative operationalization would simply split in-party positivity and out-party negativity by the original nominal categorization of whether the tweet mentioned the party or IPE in question rather than additionally requiring that the tweet is positive or negative about the entity (depending on the author's partisan relation to the target). This would most likely make the task fundamentally easier, which is a promising change given the results found in this study; despite not achieving notably high correlations with human annotators on BWS tasks, GPT performance should be interpreted with the caveat that we have most likely given GPT a difficult task. Indeed, for the preferred method of scoring (logit transformation of the value learning score), the correlation between GPT and mean human annotations was actually higher than the correlation between the annotators themselves. These results suggest merit in testing BWS methods that reduce competition within sets.

Secondly, attaining many human annotations on many items, could also enable better informed parameter choices. To continue with the 'difficulty' problem, the k parameter affects the competition with sets and also the cognitive load, which can be separated from the inherent difficulty of item ranking. However, without having access to 'true values,' we cannot interpret how changing k specifically changes the competition within tuples. Yet, to approximate the distribution of values, BWS needs to blindly settle on an initial k, which is problematic given the evidence suggesting the quality of estimates attained from k values is sensitive to the level of noise and the latent value distribution (see Hollis 2019). In general, future research should explore methods that try to approximate optimal parameter values based on multiple BWS rounds as a means of approaching the latency problem.

Notably, these methodological concerns are particularly difficult to mitigate and are endemic to the property of latent variables themselves, as latent variables are characterised by the fact that one cannot directly observe the values – only other variables that are suggestive of the values. In a modelling context where one is overwhelmingly concerned with the accuracy of predictions as opposed to making causal inferences, these issues may be less problematic and one could use BWS on subsamples using different parameters, use the BWS scores in a model and measure out of sample model predictive performance.

Obtaining BWS values that can reasonably be treated as though they approximate the true values would allow for trialling GPT on subsamples to measure GPT performance with different parameters (λ, k and b). Specifically, this approach would allow for some estimation of how GPT parameters should be scaled in accordance to t, noise and the distribution type, which could be experimentally trialled. Additionally, this type of research design could also incorporate GPT trials with items that are falsely classified as possessing the latent value of interest to empirically test how and to what extent false inclusions harm the scores of valid items and whether false inclusions are detected as outliers. If research were to operationalize BWS as is the case in this study as well as the aforementioned alternative (of splitting the scales solely by out-partisan and in-partisan references), it can then also be empirically observed how normalized item scores correlate between the operationalizations and further contextualize how outlier values should be interpreted across different operationalizations.

Lastly, obtaining values for items that can be treated as 'gold standard' would allow for comparing how different measurement methods compares with BWS (with GPT). For example, research could compare BWS with many-annotator ordinal measurements where the scale is converted to numerical values and normalized annotators means are used. These comparisons would allow for a more informed discussion about under what circumstances BWS is more/less efficient at gaining fine-grained measurement. Alternatively, with GPT implementation, research could leverage the probability of tokens from GPT responses on prompts asking for ordinally scaled measurement.

While the aforementioned avenues would generate useful insights about the properties of BWS when working with annotated items, the utility of BWS is likely context dependent, which has been partly evinced by the plethora of unanswered research questions I have unearthed despite BWS being a well-established measurement method. It would be a costly and time-consuming task for researchers

to have to obtain large amounts of annotations prior to BWS automation for each unique usage and potentially negate its advantages in being a particular efficient scaling method for variable measurement.

More widely applicable insights can be gained in future research simulating BWS methods for hypothetical item values without having the items themselves (in simulations the "best" and "worst" items are chosen automatically based on the item value. Despite previous research adopting this approach (see Hollis 2018), there are many directions in which future research could expand knowledge.

Firstly, future research could seek to better understand whether subsample distributions can be accurately obtained when using BWS with scoring algorithms over the whole sample. This would require comparing scores when BWS is used over the subsample separately. Further, subsample balance could be experimentally altered to test how imbalance effects the correlation between the BWS scores. Within this research context, for example, the sample of out-partisan negative tweets is an unbalanced, with more republican tweets that are negative toward the Democratic party and vice-versa. Scaling where the measurements are being used as covariates in statistical models, should consider potential methodological issues concerning subsample distributions. Using BWS scores interacted with partisan identity, may create spurious relationships if subsample imbalance effects the accuracy of BWS scores.

Relatedly, simulation approaches should trial BWS on (skewed) bimodal distributions, which hitherto have not been considered. Particularly, in relation to this study, it would be informative to understand how BWS with items split by hypothetical categorical variables compares with BWS over the whole distribution given bimodally distributed true values. In relation to this study, such research would pertain to scaling out-partisan and in-partisan sentiment separately.

Lastly, as I have already expressed (in section 4.2), it is my intuition that λ is more relevant to BWS performance than previous simulation research has acknowledged. Previous research has trialled different b expressed as a coefficient of t given a fixed t. However, b expressed as a coefficient of t most likely does not suffice as t gets larger owed to the combinatorial explosion of potential pair matches, and the increasing pair imbalance provided by the recommended b. Future research should compare BWS scores for different t, given different b obtained by holding λ constant and alternatively using a constant coefficient of t.

7.2 GPT for Partisan Sentiment Measurement

Given the multitude of methodological improvements that can be made with consideration to particular details, GPT for partisan sentiment classification is a promising avenue for automated measurement that can realistically be applied to answer a host of questions that are normatively pressing in social science. Due to the relative paucity and ineffectiveness of traditional ML tools available to researchers seeking to measure elite partisan sentiment, many of these pertinent questions have also been effectively negated.

In particular, automated measurement of elite partisan sentiment would make inroads into providing researchers with a means to measure elite affective polarization as either a dependent or independent behavioural variable across social media research.

Studies have generated results suggesting that elite cross-party negativity increases public partisan identity salience (Levendusky , 2013) and dislike of the out-partisan group (Levendusky , 2013; Levendusky 2016; Suhay et al., 2018; Bank et al. 2021); additionally, there is evidence that both affectively and ideologically polarizing content from elites receives greater viewership (Hong & Kim, 2016) and overall spread (Xudong et al. 2021).

These findings are indicative of what Bor et al. (2022) describe as the "connectivity hypothesis" behind the "hostility gap." The hostility gap refers to the fact that 'online discussions are felt as significantly more hostile than offline discussions' (pp. 1) despite the fact that individuals do not report being more hostile on social media. According to the "connectivity hypothesis" (Bor et al., 2022), if users are psychologically more drawn to polarizing content, affectively polarizing content by elites may receive more user engagement, which may increase its visibility, which in turn makes the online sphere feel more politically polarizing despite there being a congruence with offline behaviour. In other words, the connectivity hypothesis submits that the algorithmic amplification of certain content deceptively makes it *seem* as though the community is more polarized online.

However, beyond the connectivity hypothesis, the algorithmic distortion in the visibility of affectively polarizing content sent by elites may further create a socially undesirable feedback loop by increasing behavioural and/or attitudinal affective polarization in users who encounter the content. Indeed, Skytte (2021) utilises a survey experiment and finds results which suggest that *perceived* elite affective polarization can increase public affective polarization (Skytte 2021). An effective classifier of elite affective polarization would allow researchers to test hypotheses related to the connectivity hypothesis which have hitherto been underexplored.

For example, longitudinal survey designs measuring attitudes at multiple time frames on a sample (a panel design) linked with social media data could measure the influence of partisan-sentiment by elites on user partisan affection. Whilst researchers have used longitudinal survey data linked social media data to answer a range of research questions (Eady et al. 2020; Lee et al. 2023; Munger et al. 2022), this framework has not thus far been utilised to study effects of elite affective polarization on user affective polarization.

Moreover, à la Yu et al. (2023) and Wojcieszak et al. (2022), observational studies could seek to investigate the amplification of polarizing content by taking a sample of elite tweets and measuring the effect of partisan sentiment on engagement metrics. Additionally, the mechanisms behind the amplification of polarizing content could be explored; previous research has incorporated random samples of users and retweeters of elite tweets while ascertaining followers to use Bayesian spatial modelling to map user ideology (see Barbera, 2015) to further examine the mechanisms behind the amplification of affectively polarizing content sent by elites (Yu et al., 2023 and Wojcieszak et al., 2022).

8 CONCLUSION

The availability of LLMs has afforded researchers the ability to conduct automated classification without requiring context-specific datasets. Indeed, some of the results pertaining to GPT implementation for BWS further point to the increased scalability that LLMs can bring to research methods writ large.

Despite the practical benefits pertaining to LLMs , their "blackbox" nature can leave researchers somewhat short on ideas about how to improve classification performance. Overriding this LLM-related opacity is also the issue of latency concerning partisan sentiment, which presents further obstacles in making inferences about the appropriateness of BWS parameters and prompt designs for specific research contexts. BWS research specifically addressing the latency issue is therefore paramount in laying the groundwork for future implementations of GPT for BWS; without benchmark values, there is no standard with which to judge GPT confidently.

Relatedly, there is a great space for others to expand upon and refine some of the methods used within this study. More generally, there is tremendous potential within social science to creatively combine existing tools and methods from different research areas to make original contributions toward scalable variable measurement. Despite its numerous shortcomings, it is ultimately hoped that this research is indicative of this potential within modern social science.

REFERENCES

Auter, J. Zachary. Fine, A. Jeffrey. 2016. "Negative Campaigning in the Social Media Age: Attack Advertising on Facebook." *Polit Behav* 38, pp. 999–1020. https://doi.org/10.1007/s11109-016-9346-8

Badjatiya, Pinkesh. Gupta, Shashank. Gupta, Manish. Varma, Vasudeva. 2017. "Deep learning for hate speech detection in tweets." In *Proceedings of the 26th international conference on World Wide Web companion*, pp. 759-760

Bail, A. Christopher. Argyle, P. Lisa. Brown, W. Taylor. Bumpus, P. John. Chen, M. B. Haohan. Hunzaker, Fallin. Lee, Jaemin. Mann, Marcus. Merhout, Friedolin. Volfovsky, Alexander. 2018. "Exposure to opposing views on social media can increase political polarization." *PNAS* 115 (37), pp. 9216-9221. https://doi.org/10.1073/pnas.1804840115

Barberá, Pablo. 2015. "Birds of the Same Feather Tweet Together: Bayesian Ideal Point Estimation Using Twitter Data." *Political Analysis* 23, pp. 76–91. doi:10.1093/pan/mpu011

Basile, Valerio. Bosco, Cristina. Fersini, Elisabetta. Nozza, Debora. Patti, Viviana. Pardo, Francisco. Rosso, Paolo. Sanguinetti, Manuela. 2019. "SemEval-2019 Task 5: Multilingual Detection of Hate Speech Against Immigrants and Women in Twitter." In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pp. 54–63, Minneapolis, Minnesota, USA. *Association for Computational Linguistics*

Bor, Alexander. Petersen, Michael. 2022. "The Psychology of Online Political Hostility: A comprehensive, Cross-National Test of the Mismatch Hypothesis." *American Political Science Review* 116 (1)

Cliché, Mathieu. 2017. "BB twtr at SemEval-2017 Task 4: Twitter Sentiment Analysis with CNNs and LSTMs." In *Proceedings at the 11th International Workshop on Semantic Evaluations (SemEval-2017)*, pp. 573-580, Vancouver, Canada, August 3-4. *Association for Computational Linguistics*

Davidson, Thomas. Warmsley, Dana. Macy, Michael. Weber, Ingmar. 2017. "Automated Hate Speech Detection and the Problem of Offensive Language." In *Proceedings Of The 11th International AAAI Conference On Web And Social Media*. *Association for the Advancement of Artificial Intelligence* (AAAI)

Dong, Li. Wei, Furu. Tan, Chuanqi. Tang, Duyu. Zhou, Ming. Xu, Ke. 2014. "Adaptive Recursive Neural Network for Target-dependent Twitter Sentiment Classification." In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 49–54, Baltimore, Maryland. *Association for Computational Linguistics*

Druckman, N. James. Peterson, Erik. Slothuus, Rune. 2013. "How elite partisan polarization affects public opinion formation." *American Political Science Review* 107 (1), pp.57-79. doi:10.1017/S0003055412000500

Eady, Gregory. Bonneau, Richard. Tucker, A. Joshua. Nagler, Jonathan. 2020. "News Sharing on Social Media: Mapping the Ideology of News Media Content, Citizens, and Politicians." doi:10.31219/osf.io/ch8gj

Gambäck, Björn. Sikdar, Utpal. 2017. "Using Convolutional Neural Networks to Classify Hate-Speech." In *Proceedings of the First Workshop on Abusive Language Online*, Vancouver, Canada, 85–90. *Association for Computational Linguistics*

Gilardi, Fabrizio. Alizadeh, Meysam. Kubli, Maël. 2023. "Chatgpt outperforms crowd-workers for text-annotation tasks." *arXiv preprint arXiv:2303.15056*

Hollis, Geoff. 2018. "Scoring best-worst data in unbalanced many-item designs, with applications to crowdsourcing semantic judgments." *Behavior Research Methods, 50* (2), pp. 711-729. doi: 10.3758/s13428-017-0898-2

Hollis, Geoff. Westbury, Chris. 2018. "When is best-worst best? A comparison of best-worst scaling, numeric estimation, and rating scales for collection of semantic norms." *Behavior Research Methods, 50* (1), pp. 115-133. doi: 10.3758/s13428-017-1009-0

Hollis, Geoff. 2019. "The role of number of items per trial in best-worst scaling experiments." *Behaviour Research Methods*. doi: 10.3758/s13428-019-01270-w

Hong, Sounman. Kim, H. Sun. 2016. "Political polarization on twitter: Implications for the use of social media in digital governments." *Government Information Quarterly* 33 (4), pp. 777-782

Jiang, Long. Yu, Mo. Zhou, Ming. Liu, Xiaohua. Zhao, Tiejun. 2011. "Target-dependent Twitter Sentiment Classification." In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pp. 151–160, Portland, Oregon, USA. *Association for Computational Linguistics*

Jianqiang, Zhao. Xiaolin, Gui. Xuejun, Zhang. 2018. "Deep Convolution Neural Networks for Twitter Sentiment Analysis." In *IEEE Access* 6, pp. 23253-23260. doi: 10.1109/ACCESS.2017.2776930

Kim, Yonghwan. Kim, Youngju. 2019. "Incivility on Facebook and political polarization: The mediating role of seeking further comments and negative emotion." *Computers in Human Behavior* 99, pp. 219-227

Kim, W. Jin. Guess, Andrew. Nyhan, Brendan. Reifler, Jason. 2021. "The Distorting Prism of Social Media: How Self-Selection and Exposure to Incivility Fuel Online Comment Toxicity." *Journal of Communication* 71 pp. 922–946, doi:10.1093/joc/jqab034

King, Gary. Lam, Patrick. Roberts, Margaret. E. 2017. "Computer-Assisted Keyword and Document Set Discovery from Unstructured Text." *American Journal of Political Science* 61, pp. 971-988. https://doi.org/10.1111/ajps.12291

Kiritchenko, Svetlana. Mohammad, Saif. 2017. "Best-Worst Scaling More Reliable than Rating Scales: A Case Study on Sentiment Intensity Annotation." In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics* 2, pp. 465–470, Vancouver, Canada. *Association for Computational Linguistics*

Levendusky, Matthew. 2013. *How Partisan Media Polarize America*. University of Chicago Press

Levendusky, Matthew. Malhotra, Neil. 2016. "Does Media Coverage of Partisan Polarization Affect Political Attitudes?" *Political Communication*, 33 (2), pp.283-301, doi: 10.1080/10584609.2015.1038455

Lipovetsky, Stan. Conklin, Michael. 2014. "Best-Worst Scaling in analytical closed-form solution." *Journal of Choice Modelling* 10, pp. 60-69

Louviere, Jordan. Lings, Ian. Islam, Towhidul. Gudergan, Siegfried. Flynn, Terry. 2013. "An introduction to the application of (case 1) best–worst scaling in marketing research." *International Journal of Research in Marketing* 30 (3), pp. 293-303

Marley, A. Anthony. Louviere, J. Jordan. 2005. "Some probabilistic models of best, worst, and best–worst choices." *Journal of Mathematical Psychology* 49 (6), pp. 464-480

Mohammad, Saif. Kiritchenko, Svetlana. Sobhani, Parinaz. Zhu, Xiaodan. Cherry, Colin. 2016. "SemEval-2016 Task 6: Detecting Stance in Tweets." In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pp. 31–41, San Diego, California. *Association for Computational Linguistics*

Mohammad, Saif. Sobhani, Parinaz. Kiritchenko, Svetlana. 2017. "Stance and sentiment in tweets." *ACM Transactions on Internet Technology* 17 (3), pp. 1-23. doi:0000001.0000001

Mohammad, Saif. Bravo-Marquez, Felipe. Salameh, Mohammad. Kiritchenko, Svetlana. 2018. "SemEval-2018 Task 1: Affect in Tweets."

In *Proceedings of the 12th International Workshop on Semantic Evaluation*, pp. 1–17. *Association for Computational Linguistics*

Munger, Kevin. Egan, Patrick. Nagler, Jonathan. Ronen, Jonathan. Tucker, Joshua. 2022. "Political Knowledge and Misinformation in the Era of Social Media: Evidence From the 2015 UK Election." *British Journal of Political Science* 52 (4)

Nakov, Preslav. Ritter, Alan. Rosenthal, Sara. Sebastiani, Fabrizio. Stoyanov, Veselin. 2019. "SemEval-2016 task 4: Sentiment analysis in Twitter." *arXiv preprint arXiv:1912.01973*.

Naseem, Usman. Razzak, Imran. Musial, Katarzyna. Imran, Muhammad. 2020. "Transformer based Deep Intelligent Contextual Embedding for Twitter sentiment analysis." *Future Generation Computer Systems* 113 pp. 58–69

Osmundsen, Mathias. Bor, Alexander. Vahlstrup, Peter. Peterson, Michael. 2021. "Partisan Polarization is the Primary Psychological Motivation behind Political Fake News Sharing on Twitter." *American Political Science Review 115 (3),* pp. 999-1015

Park, Ji. Fung, Pascale. 2017. "One-step and Two-step Classification for Abusive Language Detection on Twitter." In *ALW1: 1st Workshop on Abusive Language Online*, Vancouver, Canada. *Association for Computational Linguistics*

Rathje, Steve. Mirea, Dan-Mircea. Sucholutsky, Ilia. Marjieh, Raja. Robertson, Claire. Van Bavel, J. Jay. 2023. "GPT Is an Effective Tool for Multilingual Psychological Text Analysis." *PsyArXiv*, doi:10.31234/osf.io/sekf5

Raschka, Sebastian. 2015. *Python machine learning: Unlock deeper insights into machine learning with this vital guide to cutting-edge predictive analytics*. Birmingham: Packt publishing ltd.

Rasmussen, Stig. Bor, Alexander. Osmundsen, Mathias. Petersen, Michael. 2023. "'Super-Unsupervised' Classification for Labelling Text: Online Political Hostility as an Illustration." *British Journal of Political Science*, pp. 1–22. doi:10.1017/S0007123423000042

Rheault, Ludovic. Cochrane, Christopher. 2019. "Word Embeddings for the Analysis of Ideological Placement in Parliamentary Corpora." *Political Analysis* 28, pp. 1-22. doi:10.1017/pan.2019.26

Rosenthal, Sara. Farra, Noura. Nakov, Preslav. 2017. "SemEval-2017 Task 4: Sentiment Analysis in Twitter." In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pp. 502–518. *Association for Computational Linguistics*

Rowe, Francisco. Mahony, Michael. Graells-Garrido, Eduardo, Rango, Marzia. Siever, Niklas. 2021. "Using Twitter to track immigration sentiment during early stages of the COVID-19 pandemic." *Data & Policy* 3. doi:10.1017/dap.2021.38

Russell, Annelise. 2018. "U.S. Senators on Twitter: Asymmetric Party Rhetoric in 140 Characters." *American Politics Research*, 46 (4), pp. 695–723. https://doi.org/10.1177/1532673X17715619

Suhay, Elizabeth. Bello-Pardo, Emily. Maurer, Brianna. 2018. "The polarizing effects of online partisan criticism: Evidence from two experiments." *The International Journal of Press/Politics* 23 (1), pp. 95-115

Skytte, Rasmus. 2021. "Dimensions of Elite Partisan Polarization: Disentangling the Effects of Incivility and Issue Polarization." *British Journal of Political Science* 51, pp. 1457–1475, doi:10.1017/S0007123419000760

Waseem, Zeerak. 2016. "Are You a Racist or Am I Seeing Things? Annotator Inuence on Hate Speech Detection on Twitter." *In Proceedings of the First Workshop on NLP and Computational Social Science*, Austin, Texas, pp. 138–142. *Association for Computational Linguistics*. http://aclweb.org/anthology/W16-5618

Waseem, Zeerak. Hovy, Dirk. 2016. "Hateful Symbols or Hateful People? Predictive Features for Hate Speech Detection on Twitter." In *Proceedings of the NAACL Student Research Workshop*, San Diego, California, pp. 88–93. *Association for Computational Linguistics*. http://www.aclweb.org/anthology/N16-2013

White, H. Mark. 2021. "bwsTools: An R package for case 1 best-worst scaling." *Journal of Choice Modelling 39.*

Wich, Maximilian. Bauer, Jan. Grog, Georg. 2020. "Impact of Politically Biased Data on Hate Speech Classification." In *Proceedings of the Fourth Workshop on Online Abuse and Harms*, pp. 54–64. *Association for Computational Linguistics*

Wojcieszak, Magdalena. Casas, Andreu. Yu, Xudong. Nagler, Jonathan. Tucker, A. Joshua. 2022. "Most users do not follow political elites on Twitter; those who do show overwhelming preferences for ideological congruity." *Sci. Adv* 8 (39), doi:10.1126/sciadv.abn9418

Wu, Y. Patrick. Tucker, A. Joshua. Nagler, Jonathan. Messing, Solomon. 2023. "Large language models can be used to estimate the ideologies of politicians in a zero-shot learning setting." *arXiv preprint arXiv:2303.12057*

Yu, Xudong. Wojcieszak, E. Magdalena. Casas, Andreu. 2023. "Affective polarization on social media: In-party love among American politicians, greater engagement with out-party hate among ordinary users." *Polit Behav*. https://doi.org/10.1007/s11109-022-09850-x

Zampieri, Marcos. Malmasi, Shervin. Nakov, Preslav. Rosenthal, Sara. Farra, Noura. Kumar, Ritesh. 2019. "SemEval-2019 Task 6: Identifying and Categorizing Offensive Language in Social Media (OffensEval)." In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pp. 75–86, Minneapolis, Minnesota, USA. *Association for Computational Linguistics*

Zampieri, Marcos. Nakov, Preslav. Rosenthal, Sara. Atanasova, Pepa. Karadzhov, Georgi. Mubarak, Hamdy. Derczynski, Leon. Pitenis, Zeses. Çöltekin, Çağrı. 2020. "SemEval-2020 Task 12: Multilingual Offensive Language Identification in Social Media (OffensEval 2020)." In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pp. 1425–1447, Barcelona (online). *International Committee for Computational Linguistics*

Zhang, Ziqi. 2017. "Hate Speech Detection Using a Convolution-LSTM Based Deep Neural Network."

APPENDIX

1 Prompts for GPT

1.1 Key Term Matching Algorithm

To give the reader an indication of the matching process for IPEs, below is a snippet of the IPE names list with corresponding snippets of the key terms list and the hashtag terms list.

```
IPE_name_list = [‘Nicole Malliotakis’, ‘Carolyn Maloney’, ‘Sean Patrick Maloney’]

IPE_key_kerms_list = [[['nicole','malliotakis'], ['rep', 'malliotakis'], ['representative',
'malliotakis'], ['malliotakis']], [['carolyn', 'maloney']], [['sean', 'maloney'], ['patrick',
'maloney'], ['sean', 'patrick', 'maloney']]]

IPE_hashtag_list = [['repmalliotakis','representativemalliotakis','nicolemalliotakis','malliotakis'],
['carolynmaloney'],['seanmaloney', 'patrickmaloney', 'seanpatrickmaloney']]
```

One should be able to see how the indexing logic behind key terms and hashtag terms is identical when considering parties as targets (see below). Whilst the IPE key terms list was derived using logical rules, the list of key terms for the parties was constructed based on knowledge of US politics.

```
party_names_expanded = [[['republican'], ['republicans'], ['gop'], ['grand', 'old', 'party'],
['house', 'republican'], ['house', 'republicans'], ['republican', 'party']],[['democrat'],
['democrats'], ['democratic', 'party'],['democrat', 'party'],['house democrat'],
['house', 'democrats'], ['democratic'], ['dem'], ['dems']]]

party_hashtags = [['republican', 'republicans', 'gop', ‘grandoldparty',
'houserepublican', 'houserepublicans', 'republicanparty'],['democrat',
'democrats', 'democraticparty', 'democratparty','housedemocrat',
‘housedemocrats', 'democratic', 'dem', dems']]
party_handles = [['@HouseGOP','@GOP','@SenateGOP'],['@DNC', '@TheDemocrats', '@HouseDemocrats',
'@AppropsDems','@HouseAgDems']]

Party_names = ['Republican Party','Democratic Party']
```

Take as an example a made-up tweet with its corresponding tokenization:

```
Tweet = “This tweet references the Republican Party and speaks of Republicans – in
particular, House Republicans. In addition, it uses a hashtag: #republicanparty. However, it
also mentions the Democratic Party #democratparty.”

Tokenized_expression_of_the_tweet =  [‘this’, ‘tweet’, ‘references’, ‘the’, ‘republican’,
’party ’, ’and’,  ‘speaks’,  ‘of’, ‘republicans’, ‘in’, ‘particular’, ‘house’,‘republicans’,
‘in’ ,‘addition’, ‘it’, ‘uses’, ‘a’, ‘hashtag’, ‘#republicanparty’, ‘however’, ‘it’, ‘also’,
‘mentions’, ‘the’, ‘democratic’, ‘party’, ‘#democratparty’]
```

Leaving aside hashtags (see below), my algorithm generates a list that maps key term occurrences of a party to the index position in the tokenized expression of a tweet. For the example above, the Democratic Party key words matches would take the form:

```
[([[23, 24]], ['democratic', 'party']),([[23]],['democratic'])]
```

The Republican Party key words match would take the form:

```
[([[4, 5]], ['republican', 'party']), ([[4],[9]],['republican']),([[12,13]], ['hous
e', 'republicans'])]
```

For each Party, a while loop is used to retrieve a term for the prompt only if the total number of intersections between its indexes and the indexes of larger key terms (expressed as a list for each occurrence for each term) that are non-empty is less than the number of total occurrences of the smaller term.

For the Republican Party, a while loop will only make one iteration because the maximum length of a key term match is two. The algorithm establishes that the intersection between [4,5] and [4] is non-empty but that the intersection between [9] and [4,5] is empty. Because the number of non-empty intersections is less than the length of [[4], [9]], (which is 2) 'republican' is recognized as a distinct term.

For hashtags, the algorithm first establishes whether a tweet in a string format contains a hashtag or not. If it does, all tokens with a hashtag are appended to the list 'hashtags' in lowercase format. I similarly store the index position in the tokenized expression of a tweet for each hashtag term for a given Party. However, unlike previously, for each hashtag containing a key word, the algorithm appends only the largest keyword (sub)string contained within the hashtag string and adds it to the list generated previously for non-hashtag terms. Only unique items are then finally appended.

The final result takes the form of two lists: one is a key terms list containing the unique terms used in a tweet for each party, called 'party_term_occurrence' (see section below); the other, 'party_name_occurence', is the name of the party that has been referenced with indexing that corresponds to the key terms list (see section below). In this example, party_term_occurence looks like this:

```
[[['democratic', 'party'],['democratparty']],[['republican'],['republican', 'party'], ['house', 'republicans']]]
```

This logic is key to standardizing the order of key terms matched to their relevant Party. Whilst it is not necessary when dealing with Parties because there are only 2 items considered, this approach becomes instrumental when dealing with IPEs given the size of the group and the concomitant impracticality of having different columns in a data frame corresponding to different IPEs.

1.2 Code for Key Word Matching (Party Example)

Given the created lists storing key terms for the parties (see previous section for "party_names_expanded", "party_hashtags", "Party_names") a pandas data frame, 'tweet_df', a tweet in the column 'tweet' and its tokenized expression in the column 'tweet_tokenized' we can get unique terms corresponding to each party with:

```
party_term_occurence = []
party_name_occurence = []

for s in range(len(tweets_df)):
    party_terms = []
    party_names = []
    tweet = tweets df['tweet'][s]
    tweet_tokenized = tweets_df['tweet_tokenized'][s]
    hashtags = []
    if '#' in tweet:
        for n in range(len(tweet_tokenized)):
            if '#' in tweet_tokenized[n]:
                hashtags.append(tweet_tokenized[n])
    for r in range(len(party_names_expanded)):
        party_terms_total = []
        party_names_total = []
        party_name_expanded = party_names_expanded[r]
        results_larger = []
        for t in range(len(party_name_expanded)):
            results_index = []
            results_name = []
            for ind in (i for i,e in enumerate(tweet_tokenized) if\
            e==party_name_expanded[t][0]):
                if tweet_tokenized[ind:ind+len(party_name_expanded[t])]\
                ==party_name_expanded[t]:
                    results_name.append(party_name_expanded[t])
                    results_index.append(list(range(ind,ind+len(party_name_expanded[t]))))
```

```
        results_name = [item for row in results_name for item in row]
        results_name = [ele for ind, ele in enumerate(results_name)\
        if ele not in results_name[:ind]]
        results = results_index, results_name
        if results!= ([], []):
            results_larger.append([results])
    results_larger = [item for row in results_larger for item in row]
    results_larger = [ele for ind, ele in enumerate(results_larger)\
    if ele not in results_larger[:ind]]
    if results_larger!= []:
        results_matched_unique = []
        result_lengths = []
        for c in range(len(results_larger)):
            result_lengths.append(len(results_larger[c][1]))
        longest_term_matched = max(result_lengths)

        for f in range(len(results_larger)):
            if len(results_larger[f][1]) == longest_term_matched:
                results_matched_unique.append(results_larger[f][1])

        i = 1
        while 0 < i < (longest_term_matched):
            for y in range(len(results_larger)):
                match_length = len(results_larger[y][1])
                if match_length ==i:
                    term_index = results_larger[y][0]
                    all_others_larger = []
                    for q in range(len(results_larger)):
                        if len(results_larger[q][1]) > i:
                            all_others_larger.append(results_larger[q][0])
                    establish_intersections = []
                    for items in all_others_larger:
                        for item in items:
                            for term in term_index:
                                if len(set(term).intersection(item))>0:
                                    establish intersections.append(1)

                    if sum(establish_intersections) < len(term_index[0]):
                        results_matched_unique.append(results_larger[y][1])

            i = i+1
        if results_matched_unique != []:
            party_terms_total.append(results_matched_unique)
            party_names_total.append(Party_names[r])

    if hashtags != []:
        results_larger_hashtag = []
        party_hashtag_expanded = party_hashtags[r]
        for f in range(len(hashtags)):
            for t in range(len(party_hashtag_expanded)):
                if party_hashtag_expanded[t] in hashtags[f]:
                    hashtag_and_index = [f,party_hashtag_expanded[t]]
                    results_larger_hashtag.append(hashtag_and_index)

        if results_larger_hashtag!= []:
            results_matched_unique_hashtag = []
            result_lengths_hashtag = []
            for c in range(len(results_larger_hashtag)):
                result_lengths_hashtag.append(len(results_larger_hashtag[c][1]))
```

```
                longest_term_matched_hashtag = max(result_lengths_hashtag)

                for f in range(len(results_larger_hashtag)):
                    if len(results_larger_hashtag[f][1]) == \
                    longest_term_matched_hashtag:
                        results_matched_unique_hashtag.append([results_larger_hashtag[f][1]])
                i = 1
                while 0 < i < (longest_term_matched_hashtag):
                    for y in range(len(results_larger_hashtag)):
                        match_length_hashtag = len(results_larger_hashtag[y][1])
                        if match_length_hashtag ==i:
                            term_index_hashtag = [results_larger_hashtag[y][0]]
                            all_others_larger_hashtag = []
                            for q in range(len(results_larger_hashtag)):
                                if len(results_larger_hashtag[q][1]) > i:
                                    all_others_larger_hashtag.append(results_larger_hashtag[q][0])

                            if len(list(set(term_index_hashtag) - \
                            set(all_others_larger_hashtag)))> 0 and \
                            [results_larger_hashtag[y][1]] not in results_matched_unique_hashtag:
                                 results_matched_unique_hashtag.append\
                               ([results_larger_hashtag[y][1]])
                    i = i+1
                if results_matched_unique_hashtag!= []:
                    party_names_total.append(Party_names[r])
                    for m in range(len(results_matched_unique_hashtag)):
                        party_terms_total.append([results_matched_unique_hashtag[m]])

        party_terms_unnested = [item for row in party_terms_total for item in row]
        party_terms_unnested_unique = [ele for ind, ele in enumerate(party_terms_unnested) \
        if ele not in party_terms_unnested[:ind]]
        party_names_unique = [ele for ind, ele in enumerate(party_names_total) if ele not in \
        party_names_total[:ind]]
        if party_terms_unnested_unique != []:
            party_terms.append(party_terms_unnested_unique)
            party_names.append(party_names_unique)
    party name occurence.append(party names)
    party_term_occurence.append(party_terms)

tweets_df['party term occurence'] = party_term_occurence
tweets_df['Party name occurence'] = party_name_occurence
```

1.3 Prompt Examples

In the following prompts, highlighted terms are embedded contextual terms that change in accordance to each tweet that is parsed by GPT (other than the tweet itself).

1.3.1 Prompts for Task 1

Prompts for nominal classification for target matching and ordinal sentiment classification toward the target (IPEs):

Version 1:

Below is a tweet by a US Democratic politician that potentially references another US Democratic politician. The potentially referenced US Democratic politician that I would like you to focus on is presented under the tweet and referred to as the ‘target’. Any terms contained in the tweet that are associated with the target are presented under the tweet. If the target’s twitter handle is contained in the tweet, it is also presented below the tweet. Decide if the tweet is 1) positive, 2) negative, or 3) neutral about the target. Provide the answer as ‘1’, ‘2’ or ‘3’. If the tweet doesn’t reference the target, provide the answer as ‘4’.

Tweet: 50% of US adults will have at least one dose of a COVID-19 vaccine by the end of the week — that' s reason to celebrate. @HouseDemocrats and @POTUS Biden are getting shots in arms, kids back to school, and people back to work.
Target: President Joe Biden
Used terms associated with President Joe Biden: 'Biden'
President Joe Biden's twitter handle: POTUS

Version 2:

> Below is a tweet by a US Democratic politician that potentially references President Joe Biden. The tweet contains President Joe Biden's Twitter handle, which is '@POTUS.' The additional terms contained in the tweet that are associated with President Joe Biden are: 'Biden'. Decide if the tweet is 1) positive, 2) negative, or 3) neutral about the President Joe Biden. Provide the answer as '1', '2' or '3'. If the tweet doesn't reference President Joe Biden, provide the answer as '4'.
>
> Tweet: 50% of US adults will have at least one dose of a COVID-19 vaccine by the end of the week — that's reason to celebrate. @HouseDemocrats and @POTUS Biden are getting shots in arms, kids back to school, and people back to work.

1.3.2 Prompts for Task 2

Prompt for nominal classification for target matching and ordinal sentiment classification toward the target (parties):

> Below is a tweet authored by a US Republican politician that potentially references the US Democratic Party. The terms contained in the tweet that are associated with the US Democratic Party are: 'Democrat Party', 'Democrats.' Decide if the tweet is 1) positive, 2) negative, or 3) neutral about the US Democratic Party. Provide the answer as '1', '2' or '3'. If the tweet doesn't reference the US Democratic Party, provide the answer as '4'.
>
> Tweet: Democrats are pushing an ideology of control and socialism as opposed to solving America's problems. Americans don't want the IRS spying on their bank accounts. It's unconstitutional, and we can't stand for it. The direction of the Democrat Party right now is stunning.

1.3.3 Prompts for Task 3

Prompt for Best-Worst Scaling of in-party positivity toward respective targets (parties):

> Below is a list of 4 tweets. Each tweet is authored by either a US Democratic or US Republican politician. Each author references the party he/she belongs to in the tweet by either mentioning a term or twitter handle associated with the party. Therefore, if a tweet is authored by a US Democratic politician, it references the US Democratic Party and if a tweet is authored by a US Republican politician, it references the US Republican Party. The name of the author's party and any terms contained in the tweet that are associated with the party are presented below the tweet. If a twitter handle associated with the author's party is contained in the tweet, it is also presented below the tweet. Decide which tweet is the most positive about the party to which its author belongs. Provide the answer as a single number. Answer 1 for 'Tweet1', 2 for 'Tweet2', 3 for 'Tweet3' or 4 for 'Tweet4.'
>
> Tweet1: Good luck to @SenatorTimScott as he offers a powerful and optimistic response that lays out w here Republicans want to take our great nation
> Author's party: US Republican Party
> Used terms associated with the US Republican Party: 'Republicans'
>
> *Etc…*

Note that GPT for BWS required each 4-tuple to be parsed twice, with the only difference in the prompt being that 'most' is replaced with 'least.'

1.3.4 Prompts for Task 4

Prompt for Best-Worst Scaling of out-party negativity toward respective targets (parties):

> Below is a list of 4 tweets. Each tweet is authored by either a US Democratic or US Republican Member of Congress. Each author references his/her rival party in the tweet by either mentioning a term or twitter handle associated with the party. Therefore, if a tweet is authored by a Democratic Member of Congress, it references the US Republican Party and if a tweet is authored by a Republican Member of Congress, it references the US Democratic Party. The name of the author's rival party and any terms contained in the tweet that are associated with the rival party are presented under the tweet. If a twitter handle associated with the author's rival party is contained in the tweet, it is also presented below the tweet. Decide which tweet is the most negative about the rival party of its author. Provide the answer as a single number. Answer 1 for 'Tweet1', 2 for 'Tweet2', 3 for 'Tweet3' or 4 for 'Tweet4.'
>
> > Tweet1: Republicans need to stop spreading Donald Trump's baseless and dangerous lie that the election was stolen. It caused an insurrection and got people killed.
> > Author's rival party: US Republican Party
> > Used terms associated with the US Republican Party: 'Republicans'
>
> *Etc…*

2 BWS Trials

2.1 Code

Below is the snippet of code that can be used to generate BWS trials. The comments try to explain each step.

```
#import relevent libraries
import math
import random
import itertools
from collections import defaultdict,Counter
from statistics import mean
import numpy as np
import heapq

def generate_balanced_tuples(t: int, b=lambda t: t * 2, k=4, max_item=lambda t: t):
    #input value key:
    #t = number of items, integer
    #b = the number of tuples desired
    #k = tuple size, integer
    #max_item = the number of items sorted by frequency looped over to generate possible
    #pairs choosing max_item lower than t can speed up tuple generation, likely at the
    #expense of pair frequency balance
    b = b(t) if callable(b) else b
    max_item = max_item(t) if callable(max_item) else max_item

    #if b is not provided, we assume b = 2t
    #Similarly, if max_item is not provided, we assume t
    #if k is not provided, we assume k=4
    #calculate lambda:

    #total number of pairs per tuple (for 4-tuples, pairs_per_tuple will always be 6):
    pairs_per_tuple = math.factorial(k) / (math.factorial(2) * math.factorial(k-2))
    #how many pairs will appear in our tuples:
    total_pairs = pairs_per_tuple * b
    #how many posible pairs are there:
```

```
    unique_pairs = math.factorial(t)/(math.factorial(2)* math.factorial(t-2))
    #get a lambda value to dictate a maximum pair frequency
    lambda_value = total_pairs/unique_pairs

    #use counters to keep track of item and pair frequencies
    item_counts = Counter()
    pair_counts_total = Counter()

    # store tuples and use a set for fast duplicate checks of tuples
    tuples_list = []
    tuple_signatures = set()

    #a function for sorting available items:
    def get_list(item_counts, current_tuple):
        # iterating over the list
        eligible_items = [item for item in range(t) if item not in current_tuple]
        eligible_items.sort(key=lambda x: (item_counts[x], x))
        if len(eligible_items) < max_item:
            return eligible_items
        else:
            return eligible_items[:max_item]

        #check if a tuple exists in the current list of tuples
    def generate_signature(sublist):
        # Generating a sorted tuple signature for the sublist
        return tuple(sorted(sublist))

    #a function for getting items with the minimum pair frequency and in case of ties, the item with
the
    #lowest item frequency
    def get_min_item_with_heap(dict_1, dict_2):
        # Create a list of tuples: (value from dict_2, value from dict_1, key)
        heap = [(dict_2[key], dict_1.get(key, 0), key) for key in dict_2]
        # Transform the list into a heap
        heapq.heapify(heap)
        # The smallest item based on your criteria is now at the root of the heap
        return heap[0][2]

    while len(tuples_list) < b:
        current_tuple = []
        #for the first item in a tuple, automatically add the item with the least
        #frequency across existing tuples:
        sorted_list = get_list(item_counts, current_tuple)
        current_tuple.append(sorted_list[0])
        #update the item count after appending:
        item_counts[sorted_list[0]] += 1

        #now a while loop for remaining items to be added to the tuple:
        while len(current_tuple) < k:
            #assume that we won't find an item with summed pair frequencies with
            #other items in the tuple < lambda:
            lambda_match = False
            #sort the list each time a new item has been added:
            sorted_list = get_list(item_counts, current_tuple)
            #keep track of the summed pair frequencies of considered items with
            #existing items in the tuple:
            frequency_counts_total = {}
            # only consider items if they would not replicate existing tuples (only
            #relevent for the last item in a tuple):
            for item in sorted_list:
```

```
            phony_tuple = current_tuple + [item]
            phony_signature = generate_signature(phony_tuple)
            if phony_signature not in tuple_signatures:
                #if the duplicate tuple criteria has been satisfied, store the
                #pair frequencies with items already in the tuple:
                frequency_counts = {}
                pair_list = list(itertools.combinations(current_tuple+[item], 2))
                for pair in pair_list:
                    pair = tuple(sorted(pair))
                    frequency_counts[pair] = pair_counts_total[pair]
                frequency_counts_total[item] = sum(frequency_counts.values())
                #if the summed pair counts are less than lambda, we can break
                #the loop because we've found a suitable item:
                if sum(frequency_counts.values()) < lambda_value:
                    current_tuple.append(item)
                #after appending the item, update the item frequencies:
                    item_counts[item] += 1
                    lambda_match = True
                if lambda_match:
                    break

        #if there's no available item with summed pair frequencies with items
        #already in the tuple < lambda, get the item with primarily the mimimum
        #pair frequency and secondarily the smallest item frequency in the case
        #of ties:
        if not lambda_match:
            min_item = get_min_item_with_heap(item_counts, frequency_counts_total)
            current_tuple.append(min_item)
            item_counts[min_item] += 1

    # Update the pair counts here once the tuple is finalized
    current_tuple_pairs = list(itertools.combinations(current_tuple, 2))
    for pair in current_tuple_pairs:
        pair = tuple(sorted(pair))
        pair_counts_total[pair] += 1

    #append tuple to tuple list
    tuples_list.append(current_tuple)
    #for each finalised tuple add it as a set to a wider set for fast duplicate checks
    tuple_signature = generate_signature(current_tuple)
    tuple_signatures.add(tuple_signature)

#now we are going to compute metrics to return with the tuples. We will give the min,
#max, mean and standard deviation of item and pair counts respectively. Will will aslo
#return a frequency table as this is an efficient way of giving information about
#balance, and all the above metrics and more can be deduced from the frequency table

#item metrics:
#because we've used a counter, we need to manually append 0 counts.
items_with_zero_count = int(t - len(item_counts))
item_counts = list(item_counts.values()) + [0] * items_with_zero_count
item_metrics = {}
item_metrics["mean"] = mean(item_counts)
item_metrics["max"] = max(item_counts)
item_metrics["min"] = min(item_counts)
item_metrics["standard deviation"] = np.std(list(item_counts))
#frequency count:
frequency = {}
# iterating over the list
for item in item_counts:
```

```
    # checking the element in dictionary
     if item in frequency:
         frequency[item] += 1
     else:
         frequency[item] = 1
#combine the two dictionaries into one dictionary:
item_info = {"metrics": item_metrics, "frequencies": frequency}

#pair metrics:
#because we've used a counter, we need to manually append 0 counts.
#we've already calculated the number of unique pairs and therefore can derive 0 counts
pairs_with_zero_count = int(unique_pairs - len(pair_counts_total))
pair_counts = list(pair_counts_total.values()) + [0] * pairs_with_zero_count
pair_metrics = {}
pair_metrics["mean"] = mean(pair_counts)
pair_metrics["max"]= max(pair_counts)
pair_metrics["min"] = min(pair_counts)
pair_metrics["standard deviation"] = np.std(list(pair_counts))
frequency_pair = {}
# iterating over the list
for pair in pair_counts:
    # checking the element in dictionary
     if pair in frequency_pair:
         frequency_pair[pair] += 1
     else:
         frequency_pair[pair] = 1
pair_info = {"metrics": pair_metrics, "frequencies": frequency_pair}

#return the list of tuples and the item and pair metrics
return tuples_list, item_info, pair_info
```

2.2 Item and Pair frequency distributions for GPT and Human annotations

Table A1. Item Frequencies Across 2N 4-tuples

Frequency	Number of items
7	2
8	46
9	2

Note: Standard Deviation = 0.28
r = 8

Table A2. Pair Frequencies Across 2N 4-tuples

Frequency	Number of pairs
0	625
1	600

Note: $\lambda = 0.49$
Δ Standard Deviation = 0

Table A3. Item Frequencies Across 3N 4-tuples

Frequency	Number of items
11	3
12	44
13	3

Note: Standard Deviation = 0.35
r = 12

Table A4. Pair Frequencies Across 3N 4-tuples

Frequency	Number of pairs
0	342
1	866
2	17

Note: Δ Standard Deviation = 0.030
$\lambda = 0.73$

2.3 Item and pair metrics for timed iterations

Note for all Tables: Item Δ SD and Pair Δ SD refers to the difference between the Standard Deviation of the theoretically most balanced possible BWS design given t, b and k and the generated design. The number of blocks (N blocks) is expressed as the coefficient of t (the multiple of t. Given that all iterations were on designs where perfectly item balanced tuples existed, the coefficient is always an integer. Item Δ SD and Pair Δ SD are rounded to 2 significant figures. Time is rounded to 3 significant figures.

Table A5. Algorithm Iterations with λ = 0.25

	k = 4				*k* = 5				*k* = 6				*k* = 7			
	Iteration				Iteration				Iteration				Iteration			
Metrics	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
N items (*t*)	193	385	577	817	225	401	625	801	201	361	601	801	217	385	553	841
N blocks (*b/t*)	16	32	48	68	14	25	39	50	10	18	30	40	9	16	23	35
Time (seconds)	0.224	1.72	5.80	16.4	0.327	1.70	6.03	12.8	0.206	1.12	5.69	12.4	0.258	1.34	4.17	15.1
Item Δ SD	0.34	0.31	0.29	0.31	0.33	0.33	0.33	0.32	0.34	0.35	0.35	0.34	0.34	0.35	0.34	0.35
Item range	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
Pair Δ SD	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Pair range	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table A6. Algorithm Iterations with λ = 0.50

	k = 4				*k* = 5				*k* = 6				*k* = 7			
	Iteration				Iteration				Iteration				Iteration			
Metrics	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
N items (*t*)	193	409	601	793	201	401	601	801	201	381	601	801	217	385	589	805
N blocks (*b/t*)	32	68	100	132	25	50	75	100	20	38	60	80	18	32	49	67
Time (seconds)	0.514	4.97	17.0	39.4	0.59	4.67	16.0	38.0	0.675	4.05	16.9	39.0	0.799	4.30	16.4	42.5
Item Δ SD	0.43	0.44	0.44	0.40	0.46	0.45	0.45	0.44	0.45	0.46	0.47	0.48	0.48	0.47	0.43	0.47
Item range	2	2	2	2	2	2	2	2	2	2	2	2	3	2	2	2
Pair Δ SD	0	0	0	0	0	0	0	0	0	0	0	0	6.8×10^{-4}	7.3×10^{-4}	3.3×10^{-4}	7.4×10^{-5}
Pair range	1	1	1	1	1	1	1	1	1	1	1	1	2	2	2	2

Table A7. Algorithm Iterations with λ = 0.75

	k = 4				*k* = 5				*k* = 6				*k* = 7			
	Iteration				Iteration				Iteration				Iteration			
Metrics	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
N items (*t*)	193	401	593	801	225	401	625	801	201	401	601	801	217	393	609	785
N blocks (*b*/*t*)	48	100	148	200	42	75	117	150	30	60	90	120	27	49	76	98
Time (seconds)	1.14	10.4	34.5	86.9	1.93	10.6	42.0	93.3	1.64	13.0	43.4	107	2.20	13.6	53.2	108
Item Δ SD	0.39	0.43	0.42	0.45	0.46	0.46	0.51	0.54	0.47	0.50	0.50	0.47	0.44	0.52	0.50	0.49
Item range	2	2	2	2	2	2	3	3	2	3	2	2	2	3	3	2
Pair Δ SD	0.024	0.021	0.020	0.020	0.033	0.030	0.029	0.029	0.047	0.041	0.037	0.037	0.055	0.052	0.046	0.043
Pair range	2	2	2	2	2	2	2	2	2	2	2	2	2	2	3	3

Table A8. Algorithm Iterations with λ = 1

	k = 4				*k* = 5				*k* = 6				*k* = 7			
	Iteration				Iteration				Iteration				Iteration			
Metrics	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
N items (*t*)	196	400	601	796	201	401	601	801	201	396	601	801	211	385	595	799
N blocks (*b*/*t*)	65	133	200	265	50	100	150	200	40	79	120	160	35	64	99	133
Time (seconds)	2.11	19.7	73.7	173	2.46	20.5	73.9	184	2.74	21.8	81.0	201	3.7**2**	22.4	94.6	229
Item Δ SD	0.48	0.46	0.47	0.47	0.47	0.46	0.38	0.54	0.46	0.48	0.52	0.48	0.55	0.53	0.53	0.52
Item range	3	2	2	2	2	2	3	3	2	3	3	3	3	3	3	3
Pair Δ SD	0.38	0.37	0.37	0.38	0.40	0.39	0.39	0.39	0.43	0.41	0.40	0.40	0.45	0.44	0.42	0.42
Pair range	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3

3 Confusion Matrices

Table A9. Confusion Matrix for Text-davinci-003 IPE Sentiment Classification with Version 1

		True Class			
		Negative	Neutral	Positive	Doesn't mention
Predicted Class	Negative	49	2	0	2
	Neutral	23	16	19	37
	Positive	0	2	13	0
	Doesn't mention	1	4	0	17

Table A10. Confusion Matrix for Text-davinci-003 IPE Sentiment Classification with Version 2

		True Class			
		Negative	Neutral	Positive	Doesn't mention
Predicted Class	Negative	64	4	0	0
	Neutral	8	17	13	26
	Positive	0	3	19	0
	Doesn't mention	1	0	0	30

Table A11. Confusion Matrix for GPT-4 IPE Sentiment Classification with Version 2

		True Class			
		Negative	Neutral	Positive	Doesn't mention
Predicted Class	Negative	73	11	0	3
	Neutral	0	3	1	0
	Positive	0	6	30	0
	Doesn't mention	0	0	1	53

Table A12. Confusion Matrix for Text-davinci-003 Party Sentiment Classification with Version 2

		True Class			
		Negative	Neutral	Positive	Doesn't mention
Predicted Class	Negative	81	12	2	0
	Neutral	4	18	13	5
	Positive	0	11	46	2
	Doesn't mention	0	0	0	1

Table A13. Confusion Matrix for GPT-4 Party Sentiment Classification with Version 2

		True Class			
		Negative	Neutral	Positive	Doesn't mention
Predicted Class	Negative	85	15	0	0
	Neutral	0	2	0	0
	Positive	0	23	65	1
	Doesn't mention	0	0	0	8

Table A14. Confusion Matrix for Out-partisan Party Negative Binary Classification

		True Class	
		Slightly negative	Very negative
Predicted Class	Slightly negative	20	7
	very negative	5	18

4 Prompts for human annotators

4.1 Task 1

Section 1 instructions

In each question you are shown a tweet by a US Republican politician that potentially references another US Republican politician The potentially referenced US Republican politician that I would like you to focus on is presented under the tweet and referred to as the 'target'. Any terms contained in the tweet that are associated with the target are presented under the tweet. If the target's twitter handle is contained in the tweet, it is also presented below the tweet. Decide if the tweet is 1) positive, 2) negative, or 3) neutral about the target. Provide the answer as '1', '2' or '3'. If the tweet doesn't reference the target, provide the answer as '4'.

Section 2 instructions

In each question you are shown a tweet by a US Republican politician that potentially references a US Democratic politician. The potentially referenced US Democratic politician that I would like you to focus on is presented under the tweet and referred to as the 'target'. Any terms contained in the tweet that are associated with the target are presented under the tweet. If the target's twitter handle is contained in the tweet, it is also presented below the tweet. Decide if the tweet is 1) positive, 2) negative, or 3) neutral about the target. Provide the answer as '1', '2' or '3'. If the tweet doesn't reference the target, provide the answer as '4'.

Section 3 instructions

In each question you are shown a tweet by a US Democratic politician that potentially references a US Republican politician. The potentially referenced US Republican politician that I would like you to focus on is presented under the tweet and referred to as the 'target'. Any terms contained in the tweet that are associated with the target are presented under the tweet. If the target's twitter handle is contained in the tweet, it is also presented below the tweet. Decide if the tweet is 1) positive, 2) negative, or 3) neutral about the target. Provide the answer as '1', '2' or '3'. If the tweet doesn't reference the target, provide the answer as '4'.

Section 4 instructions

In each question you are shown a tweet by a US Democratic politician that potentially references another US Democratic politician. The potentially referenced US Democratic politician that I would like you to focus on is presented under the tweet and referred to as the 'target'. Any terms contained in the tweet that are associated with the target are presented under the tweet. If the target's twitter handle is contained in the tweet, it is also presented below the tweet. Decide if the tweet is 1) positive, 2) negative, or 3) neutral about the target. Provide the answer as '1', '2' or '3'. If the tweet doesn't reference the target, provide the answer as '4'.

4.2 Task 2

Section 1 instructions

In each question you are shown a tweet by a US Republican politician that potentially references the US Democratic Party. Any terms contained in the tweet that are associated with the US Democratic Party are presented under the tweet. If the US Democratic Party's twitter handle is contained in the tweet, it is also presented under the tweet. Decide if the tweet is 1) positive, 2) negative, or 3) neutral about the US Democratic Party. Provide the answer as '1', '2' or '3'. If the tweet doesn't reference the US Democratic Party, provide the answer as '4'.

Section 2 instructions

In each question you are shown a tweet by a US Republican politician that potentially references the US Republican Party. Any terms contained in the tweet that are associated with the US Republican Party are presented under the tweet. If the US Republican Party's twitter handle is contained in the tweet, it is also presented under the tweet. Decide if the tweet is 1)

positive, 2) negative, or 3) neutral about the US Republican Party. Provide the answer as ‘1’, ‘2’ or ‘3’. If the tweet doesn’t reference the US Republican Party, provide the answer as ‘4’.

Section 3 instructions

In each question you are shown a tweet by a US Democratic politician that potentially references the US Democratic Party. Any terms contained in the tweet that are associated with the US Democratic Party are presented under the tweet. If the US Democratic Party’s twitter handle is contained in the tweet, it is also presented under the tweet. Decide if the tweet is 1) positive, 2) negative, or 3) neutral about the US Democratic Party. Provide the answer as ‘1’, ‘2’ or ‘3’. If the tweet doesn’t reference the US Democratic Party, provide the answer as ‘4’.

Section 4 instructions

In each question you are shown a tweet authored by a US Democratic politician that potentially references the US Republican Party. Any terms contained in the tweet that are associated with the US Republican Party are presented under the tweet. If the US Republican Party’s twitter handle is contained in the tweet, it is also presented under the tweet. Decide if the tweet is 1) positive, 2) negative, or 3) neutral about the US Republican Party. Provide the answer as ‘1’, ‘2’ or ‘3’. If the tweet doesn’t reference the US Republican Party, provide the answer as ‘4’.

4.3 Task 3

Instructions

Below is a list of 4 tweets. Each tweet is authored by either a US Democratic or US Republican politician. Each author references the party he/she belongs to in the tweet by either mentioning a term or twitter handle associated with the party. Therefore, if a tweet is authored by a US Democratic politician, it references the US Democratic Party and if a tweet is authored by a US Republican politician, it references the US Republican Party. The name of the author’s party and any terms contained in the tweet that are associated with the party are presented under the tweet. If a twitter handle associated with the author’s party is contained in the tweet, it is also presented below the tweet. Decide which tweet is the most positive and least positive about the author’s party. Provide the answers on the answer sheet as ‘1’, ‘2’, ’3’ or ‘4’.

4.4 Task 4

Instructions

Below is a list of 4 tweets. Each tweet is authored by either a US Democratic or US Republican politician. Each author references his/her rival party in the tweet by either mentioning a term or twitter handle associated with the party. Therefore, if a tweet is authored by a Democratic politician, it references the US Republican Party and if a tweet is authored by a Republican politician, it references the US Democratic Party. The name of the author’s rival party and any terms contained in the tweet that are associated with the rival party are presented under the tweet. If a twitter handle associated with the author’s rival party is contained in the tweet, it is also presented below the tweet. Decide which tweet is the most negative about the author’s rival party. Provide the answer on the answer sheet as ‘1’, ‘2’, ’3’ or ‘4’.