Essays on Representation Learning for Political Science Research

by

Patrick Y. Wu

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Political Science and Scientific Computing)
in the University of Michigan
2021

Doctoral Committee:

Professor Walter R. Mebane, Jr., Chair
Professor Emily Mower Provost
Professor Kevin M. Quinn
Professor Stuart Soroka

Patrick Y. Wu
pywu@umich.edu
ORCID iD: 0000-0001-6060-2562

DEDICATION

To my parents, for always inspiring me.

ACKNOWLEDGMENTS

This dissertation is the product of the time and support of many people that I am so lucky to have in my life.

None of this would have been possible without my dissertation committee. Thanks to Emily Mower Provost for helping me draw connections between my research and computer science research and showing me how to leverage machine learning to answer important social science research questions. Thanks to Kevin Quinn for providing constructive feedback on my work, encouraging me to pursue and develop my interest in computational social science, and being a model of what a good political methodologist looks like. Thanks to Stuart Soroka for helping integrate my research with political communication research and for helping me support the technical parts of my projects with substantive research.

I wanted to express my deepest gratitude to the chair of my dissertation committee, Walter Mebane. Taking your class and attending your office hours convinced me to shift to political methodology. Since then, you have been an incredibly supportive mentor and a close friend. Thank you for always taking the time to meet even during the busiest times, listening to my odd research ideas, encouraging me to develop my interests in natural language processing and computer vision, teaching me what methodological rigor looks like, and being the primary catalyst of my intellectual growth in graduate school. Collaborating with you on two of the projects in this dissertation and numerous others outside of it was the highlight of my time at the University of Michigan, and I cannot wait to continue working with you.

Beyond my committee, I want to thank Ken Kollman, Rob Mickey, Jim Morrow, Chuck Shipan, Yuki Shiraito, Rocío Titiunik, and Yuri Zhukov for all your guidance in academia and for being eager to discuss potential research topics. I wanted to give special thanks to Iain Osgood for being a fantastic teaching mentor. You have profoundly shaped my teaching philosophy, especially in methods courses. You were also one of the first to support my then-emerging interest in machine learning.

I am also grateful to the department for supporting my time at the University of Michigan. Thanks to Megan Gosling for always answering, very promptly and in great detail, any questions I had and for all your support during the job market season. Thanks to Jeremy

Mitchell for providing critical support for the Interdisciplinary Seminar in Social Science Methodology. The workshop would have been a fraction of what it was without you. And thanks to Rob Mickey and Lisa Disch for being amazing directors of graduate studies.

Thanks as well to the Center for Political Studies at the Institute for Social Research. The Roy Pierce Scholars Fund award helped jump-start the first paper, which led to my research interests in representation learning and machine learning.

I would be remiss if I did not acknowledge the academic mentors from my undergraduate days at the University of Chicago. Thanks to Roseanna Ander, Michael Dawson, Rohit Goel, Jennie Jiang, William Landes, Richard Posner, and Michael Silverstein for kickstarting this whole thing.

None of this would have been possible without the endless support of my family. To my parents, Pan Wu and Li Yan, thank you for your bottomless love and encouragement. You both have always been my number one supporters, and it is hard to express my gratitude in a few sentences. I would not be where I am today without you. To my brother, Leo: thanks for always checking in when I needed it the most, letting me talk to you about my research, and always solving any technical problems I have had.

To Samuel Baltz, Marty Davidson, Bonnie Fan, Cameron Fen, Diogo Ferrari, Izzy Gainsburg, Briana Green, Mike Hall, Hakeem Jefferson, Clara Kim, Deanna Kolberg-Shah, Saki Kuzushima, Michael Lerner, Steven Moore, Eitan Paul, Lucy Shen, Wenting Song, Pratiksha Thaker, Logan Woods, Michael Wu, Ben Yu, Jerry Yu, Theresa Yuan, and Alice Yung: thanks for keeping me grounded in these last six years. I couldn't ask for better friends to have in my corner.

To Ben Lempert, thanks for being one of my closest friends and confidant. Graduate school would have been much more challenging without you. To Cat Gao, thanks for always lending an ear, celebrating my successes, picking me up when I'm down, and always being willing to engage in any conversation, no matter how mundane the topic is. I'm glad we've only gotten closer after college. To Nicole Wu, thanks for all the engaging conversations, both about graduate school and topics outside of it, and for always providing a helping hand whenever I needed it the most. To Shen Gong, thanks for being my day one. You've been there for me through the best and the worst. To Allen Zhang, from applying to graduate schools together to finishing graduate school together, I'm glad we got to go on this journey together; thanks for being there every step of the way. To Will Tian, thanks for all the fascinating conversations we've had over the years, and I'm looking forward to a lifetime of discussion. And to Michelle Lam, thank you, for everything.

TABLE OF CONTENTS

LIST OF FIGURES

FIGURE

LIST OF TABLES

TABLE

ABSTRACT

This dissertation consists of three papers about leveraging representation learning for political science research. Representation learning refers to techniques that learn a mapping between input data and a feature vector or tensor with respect to a task, such as classification or regression. These vectors or tensors capture abstract and relevant concepts in the data, making it easier to extract information. In the three papers, I show how representation learning allows political scientists to work with complex data such as text and images effectively.

In the first paper, I propose using word embeddings to calculate partisan associations from Twitter users' bios. It only requires that some users in the corpus of tweets use partisan words in their bios. Intuitively, the word embeddings learn associations between non-partisan and partisan words from bios and extend those associations to all users. I apply the method to a collection of users who tweeted about election incidents during the 2016 United States general election. Which partisan accounts get retweeted, favorited, and followed, and which partisan hashtags are used closely correlate with the partisan association scores. I also apply the method to users who tweeted about masks during the COVID-19 pandemic. I find that users with more Democratic-leaning partisan association scores are more likely to use health advocacy hashtags, such as #MaskUp.

In the second paper, I look at the automated classification of observations with both images and text. Most state-of-the-art vision-and-language models are unusable for most political science research, as they require all observations to have both image and text and require computationally expensive pretraining. This paper proposes a novel vision-and-language framework called multimodal representations using modality translation, or MARMOT. MARMOT presents two methodological contributions: it constructs representations for observations missing image or text, and it replaces computationally expensive pretraining with modality translation. Modality translation learns the patterns between images and their captions. MARMOT outperforms an ensemble text-only classifier in 19 of 20 categories in multilabel classifications of tweets reporting election incidents during the 2016 U.S. general election. MARMOT also shows significant improvements over the results of benchmark multimodal models on the Hateful Memes dataset, improving the best accuracy

and area under the receiver operating characteristic curve (AUC) set by VisualBERT from 0.6473 to 0.6760 and 0.7141 to 0.7530, respectively.

In the third paper, I turn to the issue of computationally studying language usage evolution over time. The corpora that political scientists typically work with are much smaller than the extensive corpora used in natural language processing research. Training a word embedding space over each period, the usual approach to studying language usage evolution, worsens the problem by splitting up the corpus into even smaller corpora. This paper proposes a framework that uses pretrained and non-pretrained embeddings to learn time-specific word embeddings, called the pretrained-augmented embeddings (PAE) framework. In the first application, I apply the PAE framework to a corpus of *New York Times* text data spanning several decades. The PAE framework matches human judgments of how specific words evolve in their usage much more closely than existing methods. In the second application, I apply the PAE framework to a corpus of tweets written during the COVID-19 pandemic about masking. The PAE framework automatically detects shifts in discussions about specific events during the COVID-19 pandemic vis-à-vis the keyword of interest.

CHAPTER 1

Introduction

1.1 Representation Learning and Political Science Research

This dissertation consists of three papers about leveraging representation learning for political science research. Each of the three papers aims to develop a framework around representation learning that can help political scientists more effectively work with complex forms of data such as text, images, and combinations of modalities. The frameworks are tools for political scientists to study a more diverse set of individuals and view these individuals in a higher dimensional way.

Representation learning refers to methods that learn a mapping between raw input data and a feature vector or tensor with respect to a task, such as classification or regression. These vectors or tensors capture abstract and relevant concepts in the data; this makes it easier to extract information from the data (Bengio et al., 2014). Such an approach differs from the typical approach in quantitative political science, which usually uses data in its raw form. For example, if "age" were a covariate, then the actual value, or a simple transformation such as the log transformation, would be used in the analysis. However, such an approach does not work well with nontraditional and complex forms of data such as images or text. In their original format, these data types have feature spaces with too many dimensions and sometimes, in the case of text data, are extremely sparse, making it difficult to identify meaningful patterns and variations in the data. In other words, these forms of data must be represented in another format.

Representation learning methods automatically learn representations of the data that typically have a lower dimension than the data in its original form and can be useful for downstream tasks such as classification or regression. These methods usually use neural networks to map the data, such as words or images, to vectors or tensors. These automatically learned representations work well because different representations encode various explana-

tory factors of variation behind the data (Bengio et al., 2014). Representation learning methods from natural language processing, such as word2vec (Mikolov et al., 2013), GloVe (Pennington et al., 2014), and BERT (Devlin et al., 2018), and computer vision, such as pretrained convolutional neural networks (He et al., 2015), automatically calculate representations of words and images, respectively. But these machine learning methods are often unusable with typical political science data and produce representations that are not easily interpretable. The three papers in this dissertation aim to understand how we can use representation learning to help answer political science research questions.

1.2 Datasets Used

In the three papers, I primarily use social media data. All datasets are from Twitter, except for one of the datasets used in the third paper, which was *New York Times* text data from Rodman (2020). The rise of social media as a site of political activity presents a new, data-rich window into political behavior. Voters now tweet, share, and photograph their civic lives. Political activity on social media takes the form of both image and text; internet users signal politics in vastly different ways, from earnest endorsements of politicians to subtle endorsements of a politically inflected lifestyle; and how language is used on the internet changes over time, often rapidly. Political scientists can use representation learning techniques found in natural language processing and computer vision to study political behavior on the internet.

There are four datasets used across the three papers. The first is a dataset of tweets about election incidents during the 2016 U.S. general election (Mebane et al., 2018). The tweets in this dataset were posted between October 1, 2016 and November 8, 2016. An election incident is an individual's personal experience with voting or some other activity in the election (Mebane et al., 2018). There are a total of $315,180$ tweets that reported one or more apparent election incidents. This dataset is used in the first paper and the second paper.

The second is a dataset of tweets discussing masking during the COVID-19 pandemic. These tweets are a subset of tweets collected by the COVID-CORE project, which collected tweets from the Twitter Decahose using a set of COVID-19 related keywords (Lu & Mei, 2020). Tweets were posted between January 1, 2020 and July 1, 2020. To extract all tweets mentioning masking, I searched for tweets with the phrase "mask," ignoring all characters or spaces that came before or after the word and ignoring the case. This dataset has $757,732$ tweets. This dataset is used in the first paper and the third paper.

The third dataset is the Hateful Memes dataset, recently released by Facebook Research to

develop and test multimodal models (Kiela et al., 2020). This dataset contains 10,000 memes, with half labeled "hateful" and the other half labeled "non-hateful." Properly classifying the memes requires a multimodal model because the image and text may, on their own, not be hateful. However, the combination of the two may be hateful. An example of an unkind (but not hateful) meme that captures this idea would be a meme with an image of a skunk with the text, "Love the way you smell today." Neither the text nor the image is mean on its own. However, the combination of the two modalities makes it an unkind meme. This dataset is used in the second paper.

The last dataset is a collection of *New York Times* articles from 1855 to 2016 that have the word "equality" in their headlines (Rodman, 2020). For each article, the text consists of the headline, the first paragraph, and the abstract. There is a total of $3,105$ articles. Rodman (2020) splits the corpus into seven 25-year time slices with no overlap. This dataset is used in the third paper.

1.3 *Paper 1*: Calculating Partisan Associations of Twitter Users Using Distributed Embeddings with User Bios

In the first paper, we[1] look at how political scientists can use representation learning to create human-interpretable measures. One of the principal problems that faces political research of social media is the lack of social media users' attributes that political scientists often care about. Most notably, users' partisanships are not well-defined for most users. Some users may mention that they are a political party member in either their tweets or bio. Others may follow certain politicians or partisan accounts. Using these characteristics alone to measure partisanship or partisan associations can lead to biased results because, according to a recent Pew Research Center survey (Wojcik & Hughes, 2019), very few Twitter users actively follow multiple partisan accounts and actively tweet about politics. In "Calculating Partisan Associations of Twitter Users Using Distributed Embeddings with User Bios," we develop an approach to calculate partisan associations for a broad set of Twitter users while remaining flexible to changing political environments.

Specifically, we use Twitter user bios to measure partisan associations. The method is simple and intuitive. We use doc2vec (Le & Mikolov, 2014) to map user bios and the words that make up these bios to document and word representations, respectively. In other words,

[1]This paper is co-authored with Walter R. Mebane, Jr., Logan Woods, Joseph Klaver, and Preston Due. I am the primary author on this paper.

doc2vec learns a function that maps discrete words and documents to continuous vectors. In and of themselves, these representations are not interpretable. Still, they can be used to produce an interpretable measure when compared to each other using a similarity function. Specifically, we take the cosine similarity between the document embeddings and the embeddings of specific partisan keywords to calculate partisan association scores. The underlying idea is to learn the non-partisan words in the contextual neighborhoods of explicitly partisan words. Even if someone does not expressly use partisan words in their bio, they may describe themselves with words that the bios that feature explicit partisan expressions tend to contain.

Using the dataset of tweets about election incidents during the 2016 U.S. general election, we show that the partisan association scores capture partisan engagement and sentiment in intuitive ways, such as which partisan accounts Twitter users retweet, favorite, follow, and what partisan hashtags they use. We also calculate partisan association scores of Twitter users linked to a Pew Research Center survey (Wojcik & Hughes, 2019) and find that the partisan association scores predict self-reported partisanship. Using the dataset of tweets discussing masking during the COVID-19 pandemic, we also find that more Democratic-leaning users (users with partisan association scores less than 0) are more likely to use "health advocacy" hashtags, such as #MaskUp or #StayHome. The results align with surveys that find that Democrats are much more likely to support government-imposed measures to prevent the spread of the coronavirus (Deane et al., 2021).

1.3.1 Summary of Contributions of Paper 1

1. We show that distributed word embeddings can be used to calculate partisan association scores of Twitter users using their bios as long as some Twitter users use explicitly partisan words in their bios.

2. Our method can calculate the partisan associations of a much greater number of users compared with existing approaches.

3. We develop an application-specific loss function to select hyperparameters for doc2vec, which can be modified for other applications as well.

4. We develop a sampling approach that increases the reliability of partisan association scores.

5. We develop a pipeline accounting for negative usages of words, as negative usages of words may pollute the learned associations between partisan and non-partisan words.

1.4 *Paper 2*: MARMOT: A Deep Learning Framework for Constructing Multimodal Representations for Vision-and-Language Tasks

The second paper proposes a framework for automatically classifying observations that have both image and text. This is common with data such as social media posts. But in current research with such data, automated classification almost always focuses on either the image or text alone. Using one modality alone can potentially create biases in the classified data, leading to misleading inferences in the downstream analysis of the data. But this is not by accident. State-of-the-art multimodal models in computer science calculate a single representation for both the image and text, encoding patterns within and across the modalities. However, these models are often unusable for the type of data that political scientists are interested in because the models require every observation to have both image and text. Moreover, these models require pretraining on large image caption datasets, such as Microsoft COCO (Lin et al., 2014), in order to learn the relationships between image and text features. This demands computational resources beyond what is typically available for most researchers. Alternative models calculate an image representation and text representation and then combine them together in some way; however, these models learn limited patterns across the two modalities. The challenge, then, is to develop an architecture that can construct representations of observations with various combinations of modalities while ensuring that the model can be both trained and tested on limited computational resources and that the model works well even with small labeled datasets.

We[2] introduce a deep learning multimodal framework that constructs joint representations of images and text called **m**ultimod**a**l **r**epresentations using **mo**dality **t**ranslation, or MARMOT. MARMOT presents two methodological contributions. First, it introduces modality translation, which replaces the computationally expensive multimodal pretraining process required by many state-of-the-art multimodal models. This step learns the patterns between images and their captions. Second, the model can calculate representations even for observations missing text or images. Because the model leverages transfer learning from computer vision and natural language processing, a large labeled dataset is not required. This is particularly important for political science research, where high quality labeled datasets are often costly to produce and are thus typically small.

We apply MARMOT to two datasets. The first is the dataset of tweets about election incidents during the 2016 U.S. general election (Mebane et al., 2018). All tweets have text,

[2]This paper is co-authored with Walter R. Mebane, Jr. I am the primary author on this paper.

but only some tweets have an image. The goal is to classify tweets into various types of election incidents. MARMOT outperforms the text-only classifiers previously used with the tweet data. The second is the Hateful Memes dataset, where all memes have both text and image. The goal is to classify the combination of the text and images found in memes as hateful or not. Although MARMOT forgoes pretraining and can accommodate observations with missing modalities, it improves upon the benchmark state-of-the-art multimodal models in terms of AUC and accuracy.

1.4.1 Summary of Contributions of Paper 2

1. We introduce modality translation, which replaces the computationally expensive multimodal pretraining process required by many state-of-the-art multimodal models.
2. The model can calculate representations even for observations missing text or image.
3. Researchers can use MARMOT with small labeled datasets because MARMOT utilizes pretrained components.
4. Training and inference can be completed using free resources such as Google Colab.

1.5 *Paper 3*: Studying Language Usage Evolution Using Pretrained and Non-Pretrained Embeddings

The third paper examines how word embedding methods can be used to study language usage evolution over time. Changing language usage in politics can indicate shifts in how specific issues are discussed and what is politically salient at a given time. Natural language processing researchers have used distributed word embeddings to study the evolution of particular words (see, e.g., Kulkarni et al., 2015; Hamilton et al., 2016). Typically, separate word embedding spaces are trained using the text from each period of interest. Then, either cosine similarities between specific pairs of words are compared between each time period, or the spaces are aligned using a space alignment technique such as orthogonal Procrustes matrix alignment (see, e.g., Hamilton et al., 2016). The distributed word embedding approaches offer an advantage over topic models because the researcher can examine how usages of specific words evolve. However, the corpora that political scientists usually work with are much smaller than the extensive corpora used in natural language processing research, leading to poorly trained word embeddings. Splitting up the corpus into even smaller corpora worsens the problem.

I propose a framework, based on the theory developed in Arora et al. (2018) and its application in Khodak et al. (2018), that uses both pretrained and non-pretrained embeddings to learn time-specific word embeddings. As described in Arora et al. (2018) and Khodak et al. (2018), the goal is to learn a linear transformation between a word embedding and its contextual embedding. For each unique word w, Arora et al. (2018) and Khodak et al. (2018) use the average of the word embeddings in w's contexts as the contextual embeddings. I use the document embedding of any document that contains the word w as the contextual embedding instead; this avoids having to assume that the word embeddings of contextual words stay constant over all time periods.

I then augment each contextual embedding with a pretrained BERT embedding (Devlin et al., 2019), and I augment each word embedding with a pretrained word2vec embedding (Mikolov et al., 2013). Pretrained embeddings are embeddings trained on a very large, generic dataset such as Wikipedia or a corpus of novels, making these embeddings high quality. However, they do not encode nuances or particular uses of language often found in political science corpora. Each contextual embedding is a concatenation of the document embedding and its respective BERT embedding; each word embedding is a convex combination of its respective non-pretrained and pretrained word embedding. How frequently a word occurs in a corpus determines the weights of the convex combination. The intuition is that the more often a word appears, the higher quality its word embedding. Frequently occurring words will have word embeddings resembling its non-pretrained embedding, while less frequently occurring words will have word embeddings resembling its pretrained embedding. I then learn a linear transformation between the contextual embeddings and word embeddings; then, I apply this linear transformation to a word's context from each period to form the time-specific word embedding. I call this the pretrained-augmented embeddings (PAE) framework.

In the first application, I apply the PAE framework to a corpus of *New York Times* articles that contained the word "equality" in its title (Rodman, 2020). The challenge with this dataset is its corpus size: there are only $3,105$ articles. For each article, there is only the headline, the first paragraph, and the abstract. The PAE framework matches human judgments of how specific words evolve in their usage much more closely than existing methods. In the second application, I apply the PAE framework to the corpus of tweets that discussed masking during the COVID-19 pandemic. I show that the PAE framework automatically detects discussions about specific events during the COVID-19 pandemic vis-à-vis the keyword of interest. I also show that the PAE framework can be combined with the partisan association method in the first paper to study how word usage changes between groups with different partisan associations. I find that the PAE framework can detect different discussion patterns among Democratic-leaning and Republican-leaning users.

1.5.1 Summary of Contributions of Paper 3

1. This is the first work to present empirical evidence that using doc2vec document embeddings can produce time-specific embeddings with better properties than averaged word2vec embeddings.

2. To the best of my knowledge, this is one of the first works that use both pretrained and non-pretrained embeddings together in one framework. The only other work that touches on this is Alghanmi et al. (2020).

3. I introduce a method that can be used with small text datasets while not entirely relying on pretrained embedding spaces. Pretrained embedding spaces often do not encode nuances or particular uses of language usually found in political science texts of interest.

1.6 Future Directions

In all three papers, I argue that political scientists should use representation learning with complex and nontraditional social science data such as text, images, or combinations of modalities. Representation learning creates features that can encode patterns in high-dimensional data. These automatically learned features consistently outperform handcrafted features in classification and regression tasks. These learned representations are also useful for quantitatively studying complex phenomena, such as relationships between words and how words are used differently across time. Paper 1 shows that researchers can use representation learning to create human-interpretable measures useful in downstream analyses of social media users. Paper 2 shows that representation learning can be used to learn patterns within and across different modalities effectively. Paper 3 shows how different types of representations found in natural language processing can be combined to answer political science research questions even with small text corpora.

This dissertation argues that representation learning frameworks designed specifically for political science research can greatly increase the amount of information political scientists can extract from images and text. It is only the beginning of a line of research that aims to look at, more generally, how representation learning can be used to enhance political science research further.

Specific future directions for each project are described at the end of each paper. Here, I briefly touch on two topics: contrastive learning and algorithmic bias. First, I discuss what I believe will be the next significant development for using representation learning in

political science research: contrastive representation learning. Contrastive representation learning refers to a set of techniques that learn by comparing among different samples of the input data (for an overview of contrastive learning, see Le-Khac et al., 2020). That is, rather than calculating representations guided by human labels (as is done in Paper 2), and rather than calculating representations guided by a pseudo-label from a pretext task (as is done in Papers 1 and 3)[3], contrastive learning methods learn representations using input pairs with some measure of similarity between the pairs of inputs. Contrastive learning aims to map "similar" samples closely together and map "dissimilar" samples further away in the embedding space (Le-Khac et al., 2020).

Contrastive learning requires no human annotations, nor does it require the researcher to develop pretext tasks. Instead, it only requires some kind of similarity measure between samples from the data. These embeddings can then be finetuned using a much smaller dataset with human annotations, or these embeddings can be used to make comparisons and discover relationships among the observations. Contrastive learning is potentially beneficial for political science, as frequently, the lack of labeled data limits the use of some datasets. Currently, to the best of my knowledge, no works in political methodology have used contrastive learning techniques for learning representations of input data.

Second, I briefly discuss algorithmic bias. It is well-known that there exist biases in pretrained language models and pretrained image models. This is a critical consideration for political scientists aiming to use pretrained models in their work. Previous works have illustrated how such biases seep into pretrained models and proposed methods of detecting bias in pretrained language models (see, e.g., Bhardwaj et al., 2020; Kurita et al., 2019; Tan & Celis, 2019) and pretrained image models (see, e.g., Crawford & Paglen, 2019). However, there have been very few works in political science that consider these biases when using pretrained models with political science data. Moreover, little work has been done considering biases that exist in multimodal models. Peña et al. (2020) look at how multimodal models can be biased when assessing curriculum vitae with pictures. However, there is currently no systematic framework for analyzing bias in multimodal models. It is also unknown what biases are enhanced or attenuated when using pretrained language models and pretrained image models together. Future work aims to develop frameworks that assess what new biases may emerge when using pretrained models with political science data, how to reduce these biases effectively, and how to detect biases when they emerge.

[3]A pretext task is defined as a predefined task for the network to solve, usually using the data itself to create (pseudo-)labels (Zhang et al., 2017).

1.7 References

Alghanmi, Israa, Espinosa Anke, Luis, & Schockaert, Steven (2020). Combining BERT with static word embeddings for categorizing social media. In *Proceedings of the Sixth Workshop on Noisy User-generated Text (W-NUT 2020)*, (pp. 28–33)., Online. Association for Computational Linguistics.

Arora, Sanjeev, Li, Yuanzhi, Liang, Yingyu, Ma, Tengyu, & Risteski, Andrej (2018). Linear algebraic structure of word senses, with applications to polysemy.

Bengio, Yoshua, Courville, Aaron, & Vincent, Pascal (2014). Representation learning: A review and new perspectives.

Bhardwaj, Rishabh, Majumder, Navonil, & Poria, Soujanya (2020). Investigating gender bias in bert.

Crawford, Kate & Paglen, Trevor (2019). The politics of images in machine learning training sets. https://excavating.ai/. Accessed: 2021-07-29.

Deane, Claudia, Parker, Kim, & Gramlich, John (2021). A year of U.S. public opinion on the coronavirus pandemic. Pew Research Center.

Devlin, Jacob, Chang, Ming-Wei, Lee, Kenton, & Toutanova, Kristina (2018). BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, *abs/1810.04805*. URL: http://arxiv.org/abs/1810.04805.

Devlin, Jacob, Chang, Ming-Wei, Lee, Kenton, & Toutanova, Kristina (2019). Bert: Pre-training of deep bidirectional transformers for language understanding.

Hamilton, William L., Leskovec, Jure, & Jurafsky, Dan (2016). Diachronic word embeddings reveal statistical laws of semantic change. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, (pp. 1489–1501)., Berlin, Germany. Association for Computational Linguistics.

He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, & Sun, Jian (2015). Deep residual learning for image recognition.

Khodak, Mikhail, Saunshi, Nikunj, Liang, Yingyu, Ma, Tengyu, Stewart, Brandon, & Arora, Sanjeev (2018). A la carte embedding: Cheap but effective induction of semantic feature vectors.

Kiela, Douwe, Firooz, Hamed, Mohan, Aravind, Goswami, Vedanuj, Singh, Amanpreet, Ringshia, Pratik, & Testuggine, Davide (2020). The hateful memes challenge: Detecting hate speech in multimodal memes.

Kulkarni, Vivek, Al-Rfou, Rami, Perozzi, Bryan, & Skiena, Steven (2015). Statistically significant detection of linguistic change. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15, (pp. 625–635)., Republic and Canton of Geneva, CHE. International World Wide Web Conferences Steering Committee.

Kurita, Keita, Vyas, Nidhi, Pareek, Ayush, Black, Alan W, & Tsvetkov, Yulia (2019). Measuring bias in contextualized word representations.

Le, Quoc V. & Mikolov, Tomas (2014). Distributed representations of sentences and documents.

Le-Khac, Phuc H., Healy, Graham, & Smeaton, Alan F. (2020). Contrastive representation learning: A framework and review. *IEEE Access*, *8*, 193907–193934.

Lin, Tsung-Yi, Maire, Michael, Belongie, Serge, Bourdev, Lubomir, Girshick, Ross, Hays, James, Perona, Pietro, Ramanan, Deva, Zitnick, C. Lawrence, & Dollár, Piotr (2014). Microsoft coco: Common objects in context.

Lu, Xuan & Mei, Qiaozhu (2020). COVID-CORE social media dataset.

Mebane, Jr., Walter R., Wu, Patrick, Woods, Logan, Klaver, Joseph, Pineda, Alejandro, & Miller, Blake (2018). Observing election incidents in the united states via twitter: Does who observes matter? Paper presented at the 2018 Annual Meeting of the Midwest Political Science Association, Chicago, April 5–8, 2018.

Mikolov, Tomas, Sutskever, Ilya, Chen, Kai, Corrado, Greg, & Dean, Jeffrey (2013). Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, (pp. 3111–3119)., Red Hook, NY, USA. Curran Associates Inc.

Pennington, Jeffrey, Socher, Richard, & Manning, Christopher D. (2014). GloVe: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, (pp. 1532–1543).

Peña, Alejandro, Serna, Ignacio, Morales, Aythami, & Fierrez, Julian (2020). Faircvtest demo: Understanding bias in multimodal learning with a testbed in fair automatic recruitment.

Rodman, Emma (2020). A timely intervention: Tracking the changing meanings of political concepts with word vectors. *Political Analysis*, *28*(1), 87–111.

Tan, Yi Chern & Celis, L. Elisa (2019). Assessing social and intersectional biases in contextualized word representations.

Wojcik, Stefan & Hughes, Adam (2019). Sizing up twitter users. Pew Research Center.

Zhang, Richard, Isola, Phillip, & Efros, Alexei A. (2017). Split-brain autoencoders: Unsupervised learning by cross-channel prediction.

CHAPTER 2

Calculating Partisan Associations of Twitter Users Using Distributed Embeddings with User Bios

Abstract

We propose using distributed word and document embeddings to calculate partisan associations from bios of Twitter users. The partisan associations are the cosine similarities between the document embeddings associated with the bios and partisan subspaces defined using keywords that refer to the two parties, such as candidate names, party names, slogans, and commonly used hashtags. The proposed method can calculate a partisan association score for any individual with a bio, even if the bio does not contain any explicitly partisan words. It only requires that *some* users in the corpus of tweets use explicitly partisan words in their bios. Intuitively, word embeddings learn which non-partisan words commonly exist in the contextual neighborhood of partisan words. We extend these associations to individuals whose bios only contain non-partisan words to calculate a partisan association score. We apply this method to two applications. First, we use the method with a set of users who tweeted about an election incident during the 2016 United States general election. Which partisan accounts get retweeted, favorited, and followed, and which partisan hashtags are used closely correlate with the partisan association scores. Second, we apply the method to a collection of users who tweeted about masks during the COVID-19 pandemic. We find that users with more Democratic-leaning partisan association scores are more likely to use health advocacy hashtags, such as #MaskUp or #StayAtHome. The GitHub repository for the method proposed in this paper can be found at `github.com/patrickywu/PartisanAssociations`.

Author's Note

This work is co-authored with Walter R. Mebane, Jr., Logan Woods, Joseph Klaver, and Preston Due. I am the first author on this paper. There are references to Wu et al. (2019) throughout this paper, which is a version of this paper that focuses on Application 1. Thanks to Joseph Park and Benjamin Puzycki for their research assistance. This work was supported in part by NSF award SES-1523355, the Roy Pierce Scholars Fund award from the Center for Political Studies at the Institute for Social Research at the University of Michigan, and a fellowship from Michigan Institute for Computational Discovery & Engineering (MICDE). I indicate in footnotes sections I did not primarily write, innovations I did not primarily develop, and analyses I did not primarily conduct.

2.1 Introduction

One of the principal problems facing political research that use Twitter data is the lack of Twitter users' attributes that political scientists often care about, including users' partisanship (Steinert-Threlkeld, 2018). Some have proposed drawing on online behavior such as following or retweeting elite targets to get at partisan or ideological notions (e.g., Barberá, 2015, 2016; Barberá et al., 2019; Bond & Messing, 2015), but such behavior approaches have limitations. Such methods have trouble with users who do not engage in the behaviors. All methods are subject to missing data, but following-based methods may be unnecessarily demanding. For example, Barberá et al. (2019) require "supporters" of a party to follow three or more members of Congress from that party and no member of the other party. Text-focused methods can be similarly restrictive: Temporão et al. (2018) consider only "citizens with at least 25 political bigrams in the dynamic lexicon within each context, and who follow a minimum of 3 active candidates." These measures only capture a biased set of politically active Twitter users, which make up a small minority of all Twitter users (Wojcik & Hughes, 2019). The method proposed in this paper solves this problem by looking at Twitter users' bios, a field that is filled out by most Twitter users, and does not impose any explicit conditions on what information a bio must contain.

We propose using Twitter user bios, also known as "descriptions" in the Twitter API, to measure what we call "partisan associations." The proposed method is simple and intuitive. The method first maps bios to document embeddings and the unique words across all user bios to word embeddings using a method such as doc2vec (Le & Mikolov, 2014). The researcher then selects a set of partisan keywords for each political party; these keywords should exist across some, but not necessarily all, of the user bios.[1] These keywords can be paired in logical opposition (e.g., pairing together "Republican" and "Democrat" or opposing political slogans), or they can be grouped (e.g., selecting a set of Republican-associated keywords and a set of Democratic-associated keywords). Because these keywords exist in the user bios, the words have corresponding word embeddings. Then, depending on the substantive applications, the researcher can pool each political parties' set of word embeddings using an operation such as the mean. If the word pairings are meaningful on their own, the researcher may choose not to pool the keywords together. We then take the difference between the partisan keywords, either across individual pairs or the differences between the pooled word embeddings of each party (Kozlowski et al., 2019). We call this the partisan subspace (or partisan subspaces, if using multiple differences across pairs). Lastly, we take the cosine similarity between each user bio's document embedding and the partisan

[1] Currently, the method only works with two major opposing political parties.

subspace(s); this is the partisan association(s) for each user.

Most users' self-created bios do not include explicitly political terms. However, bios that do not include such terms also contain terms associated with a partisan identity, ideology, or political campaign. The idea of the word embeddings method is to learn the non-partisan words in the contextual neighborhood of explicitly partisan words. Even if someone does not explicitly use partisan words, they may describe themselves with words that other bios that feature explicit partisan expressions tend to contain. We also use the term "partisan association" rather than "ideology" because the term "ideology" tends to refer to an item response theory-type model that posits there is a single latent dimension characterized as ideology (see, e.g., Barberá, 2015; Clinton et al., 2004). Nothing about the method outlined above indicates a single latent dimension; instead, the method above measures how the words in users' bios are associated with selected partisan keywords.

This idea resonates with research that studies the associations between partisan sentiments and seemingly non-partisan identities, activities, hobbies, and interests (see, e.g., Klar, 2014; Huddy et al., 2015; McConnell et al., 2018; Abramowitz & Webster, 2016; Mason, 2018; Hetherington & Weiler, 2018). Cultural objects and personality traits are associated with political actors or parties. When presented with an image of an individual in a given setting or a list of personality traits, individuals frequently assume that the pictured or described individual is a member of a specific party (Goggin et al., 2019; Haieshutter-Rice et al., 2021). While people may or may not describe themselves in explicitly political terms on social media, they may describe hobbies, fandoms, or traits that are correlated with partisan sentiments. Learning the associations between non-partisan terms and partisan terms allows us to compute partisan associations of users who do not explicitly use partisan terms to describe themselves.

This paper proceeds as follows. We first review the literature on word embeddings and partisan associations of non-political elites. We then describe the method of calculating partisan associations for a set of Twitter users. We then apply the method to Twitter users who tweeted once or more about an election incident during the 2016 U.S. general election (Mebane et al., 2018). We find that the partisan associations capture partisan engagement and sentiments in intuitive ways. For example, partisan associations successfully predict what type of partisan accounts the users follow and what partisan hashtags they tend to use. We also apply the method to a dataset of tweets related to masking during the COVID-19 pandemic. These tweets were posted between January 1, 2020 to July 1, 2020. Again, the partisan association scores can predict what type of partisan hashtags they tend to use. Lastly, we touch on the issue of users using partisan keywords in a negative fashion, such as attacking or expressing discontent with a political party or candidate.

2.2 Background

This work is situated at the intersection of the literatures in automated extraction of political positions from text, calculating or estimating ideology for non-elite political actors on social media, and how non-partisan entities are associated with certain parties. We first review the literature in these fields. We then give a background to word embeddings.

2.2.1 Automated Extraction of Political Positions from Texts

Wordscores were one of the first automated methods of extracting political positions from texts (Laver et al., 2003). Wordscores take a training set of texts with known positions on well-defined a priori dimensions, generate word scores from this training set, scores each "virgin text" (unlabeled texts) using these word scores, and then transform the test set's scores to the original metric. Wordscores are calculated using the counts of tokens. The underlying intuition is that certain documents from certain positions along well-defined dimensions will use certain words more than other words. Lowe (2008) identified several issues with this approach, such as showing that it is unable to differentiate between politically-charged words used by different parties, such as "taxes."

Wordfish is another approach to extract political positions from texts automatically. It uses word frequencies to place documents along a single dimension, and assumes that word counts follow a Poisson distribution (Slapin & Proksch, 2008). It does not require a selection of reference texts. Its modeling procedure echoes that of NOMINATE, in that it estimates the underlying dimension's effect on the choice of words in political manifestos (Poole, 2005). But much like NOMINATE, the interpretation of the underlying dimension is made a posteriori, and it is usually assumed to be some variation of an underlying left-right ideological spectrum.

More recently, Gentzkow et al. (2019) measure polarization within Congressional speeches using a discrete choice model, where certain words used in Congressional speeches are assumed to have some payoff. Specifically, they assume that what the speaker says can move public opinion toward his or her preferences. Partisan polarization is measured by how different or similar Democrats and Republicans speak on the floor.

In the last few years, many papers have used word embeddings to measure positions along dimensions. Bolukbasi et al. (2016) construct, using the linearity property of word embeddings, what they call the gender subspace. This subspace is the principal components of the differences of specific gender word pairings, such as $\overrightarrow{\text{woman}} - \overrightarrow{\text{man}}$. Through this, they find that words such as "philosopher" tend to be associated with words related to "man." At the same time, words such as "nurse" tend to be associated with words related to "woman." Kozlowski et al. (2019) take this one step further: they also use dimensions induced by

differences of specific word pairings, but they further assume they correspond to dimensions of cultural meaning. They project words into these dimensions to show that certain words are associated with certain race, class, or gender connotations. For example, they construct a "class" dimension using the differences of words like "rich" and "poor." Then, they find that words such as "volleyball" and "golf" tend to be more closely associated with the "rich" end of the class dimension while words like "camping" and "weightlifting" tend to be more closely associated with the "poor" end of the class dimension.

2.2.2 Political Positions of Social Media Users

Researchers have primarily focused on social network behavior to determine the political positions of social media users. Barberá (2015) develops a Bayesian spatial following model. Under the assumption that social networks are homophilic, the model considers ideology as a latent variable whose value can be inferred by examining which political actors and partisan entities, such as Sean Hannity and Rachel Maddow, the user follows. The intuitive idea is that the more Republican (Democratic) politicians or partisan entities the user follows, the more Republican (Democratic) the user leans. The model does require the user to follow at least one political account; in practice, however, Barberá (2015) restricted users to those who followed at least three political accounts (and, for American users, mentioned "Obama" or "Romney" at least three times in their timelines). Although these requirements seem like a low bar, they are still restrictive. According to a Pew Research Center analysis, in a survey of U.S. adults with Twitter accounts, of all tweets between June 10, 2018 to June 9, 2019, 69% of the users tweeted about politics once or never. Among the 31% of users who tweeted about politics at least twice, only 6% were "prolific political tweeters," meaning that they posted ten tweets during the study period and 25% or more of all tweets mentioned national politics (Wojcik & Hughes, 2019). The range of users that can have their ideal points estimated is somewhat narrow across the Twittersphere.

Besides the Bayesian spatial following model, Bond & Messing (2015) also use a similar spatial modeling approach, using a singular value decomposition approach of an affiliation matrix instead to estimate the ideologies of Facebook users. Temporão et al. (2018), on the other hand, use a text-based approach. Using Wordfish, they first parse the lexicon of political elites to create dynamic dictionaries; these dictionaries are then used to create ideological dimensions that they claim structure political discourse. Then, Temporão et al. (2018) estimate the positions of individual users along these scales by using the previously computed dynamic dictionaries from political elites. Again, there are restrictions: non-political elites must use at least 25 political bigrams in the dynamic dictionaries, and they

must follow 3 active candidates.[2]

2.2.3 Finding the Partisan in the Seemingly Non-Partisan

Hetherington & Weiler (2018) argue that there are stark cultural divides between Republicans and Democrats. They claim that "[h]ow Americans identify politically is now inseparable from their tastes and preferences about a dizzying array of both the most personal and mundane matters" (Hetherington & Weiler, 2018, p. 90). They describe surveys that find that car preferences, coffee brand loyalty, beer choices, food orders, types of books read, television shows watched, and music choices all split along partisan lines (Hetherington & Weiler, 2018, p. 89-121). As they aptly summarize, "[t]hey drive different cars, watch different sports, and drink different beer. Because worldview animates both non-political and political preferences, politics is no longer just about politics. It is about life" (Hetherington & Weiler, 2018, p. 120).

Hetherington & Weiler (2018) attribute these differences to differing worldviews of Republicans and Democrats. They cite personality psychology research that indicates that Democrats are more open to experiences and are more neurotic than Republicans. They describe Democrats' worldviews as fluid, indicating that they "support changing social and cultural norms, are excited by things that are new and novel, and are open to, and welcoming of, people who look and sound different." On the other hand, they describe Republicans' worldviews as fixed, indicating that they "are warier of social and cultural change and hence more set in their ways, more suspicious of outsiders, and more comfortable with the familiar and predictable" (Hetherington & Weiler, 2018, p. xiii). Other works also confirm the divide beyond lifestyle choices and social preferences that Hetherington & Weiler (2018) discuss. For example, Republicans and Democrats split on moral issues (Graham et al., 2012) and Republicans and Democrats speak differently (Cichocka et al., 2016). These differences are not only observed through surveys and experiments—these differences are also well-known by Republicans and Democrats. Haieshutter-Rice et al. (2021) find that the typical survey respondent can readily assign partisanship to objects and activities.

Regardless of why non-partisan items and activities are imbued with a partisan attachment, it is clear that this is a real phenomenon that people are generally aware of. How people describe themselves in their user bios on social media can indicate their partisan association, even if they do not explicitly spell out their partisan attachments. More importantly, based on this line of research, people should be aware that what information they

[2]Their studies were conducted in the context of a Canadian federal election, New Zealand general election, and a Quebec provincial election, which is why they require that users of interest must follow more than two active candidates.

put out—their jobs, interests, and hobbies—indicates some kind of partisan association.

2.2.4 Word Embeddings

A word embedding maps words in a corpus to real vectors, and these vectors are part of the same vector space. These vectors are also known as embeddings. The positionings of these word vectors capture how the words are related to each other. In a class of word embedding methods known as distributed word embeddings, such as word2vec (Mikolov et al., 2013) and GloVe (Pennington et al., 2014), word vectors closer in space have similar meanings, while word vectors farther apart in this space have more different meanings.

There exist many word embedding methods, but we focus specifically on distributional word embedding methods. These word embeddings find their roots in the study of distributional semantics, which studies the similarity between linguistic terms, such as words and phrases, based on their distributions across large samples of language data. The distributional hypothesis states that the contextual information of a word alone can give a viable representation of a word (Firth, 1957; Wittgenstein, 2009).[3] In other words, the meaning of words emerges from the linguistic contexts it inhabits across documents and spoken languages.

Distributed word embedding methods implement this idea. Word embeddings have a long history of development, beginning with vector space models (e.g., Salton, 1962) and, later on, latent semantic analysis (Deerwester et al., 1990). We focus on developing the intuition behind word2vec because doc2vec, the word embedding method used in this paper, is an extension of word2vec.

The basis of word2vec is a shallow two-layer neural network (that is, with one hidden layer) that is trained on a proxy classification task (Mikolov et al., 2013). To obtain the "inputs" to this model, we move a sliding window of fixed size along the text. At each position, the center word in the window is called the target word. The words surrounding the words are called the context words. For example, consider the sentence "Stubbs the cat was the mayor of Talkeetna, Alaska." Table 2.1 contains the breakdown of the sentence into each target word and its respective context.

The goal of distributional representations is to learn the relationship between a target word and its context. We can either predict the target word using the context words or predict the context words using the target word. The former approach is known as continuous bag-of-words (CBOW), and the latter approach is known as skip-gram. Table 2.2 describes the

[3]An excellent example of this phenomenon is the "Jabberwocky" by Lewis Carroll. In that poem, there are numerous made-up words. However, the reader can still "understand" the poem because the context around each made-up word hints at what the word means.

Frame	Window (Size=5)	Target Word	Context
1	[Stubbs, the, cat]	Stubbs	[the, cat]
2	[Stubbs, the, cat, was]	the	[Stubbs, cat, was]
3	[Stubbs, the, cat, was, the]	cat	[Stubbs, the, was, the]
4	[the, cat, was, the, mayor]	was	[the, cat, the, mayor]
5	[cat, was, the, mayor, of]	the	[cat, was, mayor, of]
6	[was, the, mayor, of, Talkeetna]	mayor	[was, the, of, Talkeetna]
7	[the, mayor, of, Talkeetna, Alaska]	of	[the, mayor, Talkeetna, Alaska]
8	[mayor, of, Talkeetna, Alaska]	Talkeetna	[mayor, of, Alaska]
9	[of, Talkeetna, Alaska]	Alaska	[of, Talkeetna]

Table 2.1: The breakdown of the sentence "Stubbs the cat was the mayor of Talkeetna, Alaska." into each target word and its respective context. A window size of 5 is used. Notice that every word in the sentence plays the role of target word, even if there are not enough context words to the left or right of the target word.

input-output pairings created out of both approaches. The objective is to maximize this probability. To do this, we can adjust the values of the word embeddings.

Frame	Skip-Gram Observations	CBOW Observations
1	(Stubbs, the); (Stubbs, cat)	(the, Stubbs); (cat, Stubbs)
2	(the, Stubbs); (the, cat); (the, was)	(Stubbs, the); (cat, the); (was, the)
3	(cat, Stubbs); (cat, the); (cat, was); (cat, the)	(Stubbs, cat); (the, cat); (was, cat); (the, cat)
4	(was, the); (was, cat); (was, the); (was, mayor)	(the, was); (cat, was); (the, was); (mayor, was)
5	(the, cat); (the, was); (the, mayor); (the, of)	(cat, the); (was, the); (mayor, the); (of, the)
6	(mayor, was); (mayor, the); (mayor, of); (mayor, Talkeetna)	(was, mayor); (the, mayor); (of, mayor); (Talkeetna, mayor)
7	(of, the); (of, mayor); (of, Talkeetna); (of, Alaska)	(the, of); (mayor, of); (Talkeetna, of); (Alaska, of)
8	(Talkeetna, mayor); (Talkeetna, of); (Talkeetna, Alaska)	(mayor, Talkeetna); (of, Talkeetna); (Alaska, Talkeetna)
9	(Alaska, of); (Alaska, Talkeetna)	(of, Alaska); (Talkeetna, Alaska)

Table 2.2: The input-output pairings created out of each frame using the example from Table 2.1.

To make this discussion more concrete, we look closer at skip-gram as this is the variation of word2vec used with the partisan association method. The details of CBOW are very similar. For a sequence of words $w_1, ..., w_T$, we want to maximize the average likelihood.

$$L(\theta) = \prod_{t=1}^{T} \prod_{-m \leq j \leq m, j \neq 0} P(w_{t+j}|w_t; \theta)$$

The objective function is the average of the negative log-likelihood.

$$J(\theta) = -\frac{1}{T} \log L(\theta) = -\frac{1}{T} \sum_{t=1}^{T} \sum_{-m \leq j \leq, j \neq 0} \log P(w_{t+j}|w_t; \theta)$$

To calculate the probability, we will use two vectors per word: $\mathbf{v}_w$ when w is the target word,

and $\mathbf{u}_w$ when w is the context word. To calculate the probability that a context word w_O (denoted using O because the context word is considered the output in skip-gram) is in the context of the target word w_I (denoted using I because the target word is considered the input in skip-gram), we use a softmax function.

$$P(w_O|w_I) = \frac{\exp\left(\mathbf{u}_{w_O}^\top \mathbf{v}_{w_I}\right)}{\sum_{\omega \in V} \exp\left(\mathbf{u}_\omega^\top \mathbf{v}_{w_I}\right)} \tag{2.1}$$

where V is the set of unique words in the corpus (also known as the vocabulary).

Equation 2.1 is the setup of a two-layer neural network—a neural network with one hidden layer—with no activation function (Hastie et al., 2009). $\mathbf{v}_{w_I}$ for the target word w_I are the weights from the first layer of the neural network when the input is a one-hot vector for the word w_I. A one-hot vector is simply a vector that is the dimension $|V|$, the number of unique words in a corpus, where the specific dimension for word w is set to 1 and all other dimensions are set to 0. $\mathbf{u}_{w_O}$ is the weight from the second layer for the context word. Word2vec denotes the matrix of weights $\mathbf{v} \in \mathbb{R}^{|V| \times d}$ as the set of word embeddings, where d is the number of hidden nodes chosen by the researcher. For a brief introduction to two-layer neural networks, see Section 2.9.1 in the Supplemental Information.

From training the neural network, the context words w_O that are near the target word w_I will have a high probability $p(w_O|w_I)$ and non-context words $w_{\tilde{O}}$ that are not near the target word w_I will have a low probability $p(w_{\tilde{O}}|w_I)$. So if two words have similar contexts (that is, similar words appear around the two words), the neural network has to output very similar results for the two words. The way to make the output of a neural network similar across these two words is to make the weights associated with the two words similar. In other words, if two words have similar contexts, the neural network will learn similar word vectors for these two words, which means they will appear near each other in vector space.

The denominator of Equation 2.1 is very costly to calculate, as it requires calculating the exponential for every word in the vocabulary for every sample. Common approaches to solving this issue are the hierarchical softmax (Morin & Bengio, 2005), noise contrastive estimation (Gutmann & Hyvärinen, 2010), and negative sampling (Mikolov et al., 2013). The idea behind negative sampling, the most popular approach to calculating word2vec embeddings, is that we can sample a set of words that do not exist in the context of a given word. Then, the objective function becomes

$$J(\theta) = -\left[\log \sigma\left(\mathbf{u}_{w_O}^\top \mathbf{v}_{w_I}\right) + \sum_{i=1}^{k} \log \sigma\left(-\mathbf{u}_{w_{\tilde{O}_i}}^\top \mathbf{v}_{w_I}\right)\right]$$

where $\tilde{o}_i \sim Q$ and k is the number of negative samples. Q is the negative sampling distribution, which is typically uniform or an empirical unigram (Mikolov et al., 2013).

The researcher must set several hyperparameters for word2vec, including:

- Dimensionality: the dimensions of the word vectors
- Window size: the size of the window around the target word
- Minimum count: how many times a word has to appear in the corpus for it to be assigned a vector; if a word occurs too infrequently, it may have a poorly calculated word embedding
- Model type: skip-gram or continuous bag-of-words; typically, skip-gram works better for infrequently occurring words, but CBOW is faster, which is an important consideration for large corpora
- Number of iterations (epochs): how many times to iterate over the corpus during training
- Text preprocessing: whether to stem the words, remove stopwords, remove punctuation, remove numbers, remove very short words, etc.

The selection of these hyperparameters is not trivial and is typically driven by the substantive task of interest. For an overview of hyperparameter selection for word embeddings, see Rodriguez & Spirling (2021). For an overview of text preprocessing, see Denny & Spirling (2018).

A useful property observed about distributed embeddings is that the embeddings are linearly related. Mikolov et al. (2013) found that, over a large corpus, embeddings can solve analogies using basic linear algebra. They find, for example, that $V_{\text{France}} - V_{\text{Paris}} + V_{\text{Germany}} \approx V_{\text{Berlin}}$. Here, the approximately equal sign indicates that the closest word vector, in terms of cosine similarity, to the result of the left-hand side was the word vector associated with "Berlin." Cosine similarity is defined as

$$\cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|}$$

where $\cos(\theta)$ is the cosine similarity. This particular property—that we can solve these analogies using simple algebraic operations with the vector representations of words—has raised much interest around word2vec, especially in computational social science.

Doc2vec (Le & Mikolov, 2014) creates a paragraph, or document, embedding in the same vector space as the word embedding. Instead of only taking words from documents in a

corpus, every document is associated with a paragraph token. This paragraph token can be thought of as another word. The distributed memory of paragraph vectors, or PV-DM, uses the paragraph token and a set of context words to predict a target word. PV-DM is analogous to word2vec's CBOW approach. The distributed bag of words version of paragraph vectors (PV-DBOW) uses a paragraph token to predict the words in that document. Both approaches can learn word and document embeddings within the same vector space, making documents and words comparable to one another using similarity measures such as the cosine similarity.

2.3 Partisan Associations from Word Embeddings

Two features of distributed word embeddings support computing partisan associations: the ability of the word embeddings to encode each word's typical contextual neighborhood and that we can use basic linear algebra arithmetic to learn complex relationships between words. The steps of calculating partisan associations are as follows.

1. Select partisan keywords
2. Choose the hyperparameters of the doc2vec algorithm
3. Calculate what we call the partisan subspace using the differences between the word embeddings corresponding to the partisan keywords
4. Obtain partisan association scores using the partisan subspaces
5. Apply a method motivated by sampling ideas to stabilize the scores

2.3.1 Choosing Partisan Keywords

We choose keywords that reflect interests and ideas about what distinctions are important for the political time period of interest. For example, for a dataset of tweets coming from the 2016 U.S. general election period, we would use keywords like "Trump" and "Clinton," who were the Republican and Democratic candidates, respectively. Other appropriate keywords include the candidates' first names, campaign slogans, popular or widely recognized hashtags, Twitter handles, and party names.

Related methods have used pairs of words (Bolukbasi et al., 2016; Kozlowski et al., 2019). For example, Bolukbasi et al. (2016) used paired-up terms such as "she" and "he," "her" and "his," "woman" and "man," and so on to study the gender bias of certain word embeddings. The plan is to take the difference between these pairs of words and to combine them —or not,

as the case might be—to create the *subspace* of interest. However, depending on the partisan subspace, it is unnecessary to choose pairs of partisan keywords across the two parties. One can also choose a set of Democratic keywords and a set of Republican keywords.

It is also important to consider the word's usage in the corpus. Many users may negatively discuss a candidate or party. Thus, it may appear they are associated with a party when they are actually disparaging that party. Section 2.6 discusses this issue further.

2.3.2 Choosing Hyperparameters for Doc2Vec

After selecting partisan keywords, we select the hyperparameters for doc2vec. The hyperparameters to select are the dimensions of the word and document vectors, the window size, the minimum count, the model type, and the number of iterations. Section 2.2.4 describes the details of these hyperparameters. Rodriguez & Spirling (2021) discuss hyperparameter selection when there is no specific goal for the analysis. An application-specific loss function, such as the one used in Section 2.4, can be designed to select the parameters. Using Rodriguez & Spirling (2021)'s suggested hyperparameters or the default values in the `gensim`, the Python package most popularly used to train document and word embeddings yield good results (Řehůřek & Sojka, 2010). In both applications in this paper, PV-DBOW with skip-gram far outperformed PV-DM. Further research is needed to understand why PV-DBOW works much better for these two applications compared with PV-DM.

2.3.3 Obtaining the Partisan Subspaces

To obtain the partisan subspaces, the method draws on the word embeddings' analogy-solving property. Bolukbasi et al. (2016) observe that the vectors for "she," "he," "man," and "woman" satisfy $V_{\text{she}} - V_{\text{he}} \approx V_{\text{woman}} - V_{\text{man}}$, where V_w is the word vector for word w. The direction given by $V_{\text{she}} - V_{\text{he}}$ captures approximately the same concept as the direction given by $V_{\text{woman}} - V_{\text{man}}$. The vector differences define what Bolukbasi et al. (2016) call the gender subspaces. Given the varying usages of gendered words like "she," "woman," "her," etc., they suggest combining the difference vectors by finding principal components of the subspaces to create an overall gender subspace. Kozlowski et al. (2019) suggest a simpler approach: simply average the subspaces.

We similarly calculate partisan subspaces: take the difference between the word vectors corresponding to the pairs of partisan keywords, as described in Section 2.3.1. These partisan subspaces can be kept separate or combined in some fashion, such as those suggested in Bolukbasi et al. (2016) and Kozlowski et al. (2019).

We also focus on what the partisan keywords have in common. While it is natural to think

in terms of opposing keywords—such as "Republican" versus "Democrat"—it is essential also to recognize what the keywords represent that is similar among them. We denote the average of all keywords' embeddings, given that there is an equal number of partisan keywords for both parties, the *usage* subspace because it reflects how much the bios use explicitly partisan terms, regardless of partisan direction. Differences between pairs of the partisan keywords—like the difference between $V_{\text{Republican}}$ and V_{Democrat}—reflect degrees of association with one side or the other with the common usage subspace removed.

2.3.4 Computing Partisan Association Scores

Obtaining partisan association scores for the subspaces is straightforward. The linearity of word embedding spaces allows document and word embeddings to be directly compared using similarity measures, the most popular of which is the cosine similarity. For each partisan subspace P_j (or partisan subspace P if the partisan subspaces are combined), where $j \in \{1, ..., K\}$ where K is the total number of partisan subspaces, and document embedding D_i, for user $i \in \{1, ..., N\}$ where N is the total number of users with bios, the partisan association score S_{ij} is calculated as $S_{ij} = \frac{D_i \cdot P_j}{\|D_i\| \|P_j\|}$.

Interpretation of these partisan association scores is also intuitive. Assuming that the partisan subspace is calculated, for example, as $V_{\text{Republican keyword}} - V_{\text{Democratic keyword}}$, a partisan association score $S_{ij} > 0$ is a more "Republican" score while $S_{ij} < 0$ is a more "Democratic" score.

2.3.5 Obtaining Stable Partisan Association Scores: Multiple Replicates of Doc2vec

Word2vec and doc2vec produce embedding spaces that are rotation invariant (see, e.g., Park et al., 2017). But, in theory, the cosine similarities between pairs of words should be the same between word embedding spaces trained on the same corpus. However, as described in Section 2.2.4, doc2vec has several sampling aspects. There is sampling variation across replicates: successive independent executions produce different partisan associations. We[4] adopt a sampling solution for the sampling variations. In general, the average of several means from independent samples from the same population has smaller variability than any one of the separate sample means. We view the cosine similarities as sample parameters and use the average of the similarities computed using several replicates. More precisely, word embeddings are real vectors that result from approximating a function that maps words to

[4]Walter Mebane and I jointly developed this approach.

real vectors. The embedding algorithm uses sampling when approximating this function. For N users and K partisan subspaces, cosine similarities computed using the word embeddings are $N \times K$ functions of the sample estimates. We generate several realizations of the $N \times K$ cosine similarities then average them to get $N \times K$ average cosine similarities.

2.3.6 Summary of the Overall Approach

The previous sections fully describe how to calculate partisan associations. Algorithm 2.1 summarizes the calculations as previously described.

2.4 Application 1: Election Incidents Reported on Twitter During the 2016 U.S. General Election

2.4.1 Data

We[5] first look at a dataset of tweets about election incidents during the 2016 United States general election. An election incident is an individual's personal experience with voting or some other activity in the election (Mebane et al., 2018). Mebane et al. (2018) detail motivations for collecting the data and the procedures used. Mebane et al. (2018) used the Twitter API to collect slightly more than six million original tweets (excluding retweets) from the period of October 1, 2016 to November 8, 2016. From these tweets, Mebane et al. (2018) used supervised text classification methods to identify 315,180 tweets that reported one or more apparent election "incidents." Of these tweets, 215,230 distinct users posted the tweets classified as apparent incidents. Of these, 194,336 users had non-empty bios. We apply the partisan association method detailed in the previous section on these bios.

Mebane et al. (2018) also collect additional information for these 215,230 distinct users. These users' timelines were collected between January 24, 2018 to February 3, 2018; 196,276 users had a timeline ranging between 1 and 3,339 tweets. Favorites ("likes") were collected between May 14, 2018 to July 26, 2018; favorites could be collected for 185,531 users. From February 18 to March 5, 2018, Mebane et al. (2018) also collected up to 10,000 Twitter "friends" of each user. A "friend" is another user the user follows. Mebane et al. (2018) also collected timelines for all members of the House and Senate on May 10, 2018.

[5]Most of the analyses in this section are in line with the analyses conducted in Wu et al. (2019).

Algorithm 2.1: Obtaining the partisan association scores of Twitter users using their bios

Data: vocabulary $\mathcal{V}$, corpus of bios $\mathcal{C}_\mathcal{V}$ for N users

Result: Partisan associations Σ of N users across K partisan subspaces

Select a set of keywords for each party. These can be natural opposing pairs of keywords (such as "Republican" and "Democrat") or a set of keywords for each party. Denote the first set of keywords related to party A as W_A and denote the second set of keywords related to party B as W_B; if pairing words, denote the pairs as W_{Ai} and W_{Bi} for pair i.

Select hyperparameters for doc2vec

Initialize Γ, a list collecting the partisan associations for each replicate

Set M total number of replicates of the doc2vec embedding space

for $m \in \{1, ..., M\}$ **do**

    Train doc2vec embedding space using $\mathcal{C}_\mathcal{V}$

    `// Calculate partisan subspaces`

    if *set of keywords for each party without pairing* **then**

        $K = 1$

        $P = \text{mean}(W_A) - \text{mean}(W_B)$ `// or any other way of aggregating keywords`

    else

        if *not combining partisan subspaces* **then**

            $K =$ number of unique subspaces

            for $k \in \{1, ..., K\}$ **do**

                $P_k = W_{Ak} - W_{Bk}$

            end

        else

            $K = 1$

            $P = \text{mean}(W_A) - \text{mean}(W_B)$ `// or any other way of aggregating keywords`

        end

    end

    `// Calculate partisan associations`

    for $i \in \{1, ..., N\}$ **do**

        for $j \in \{1, ..., K\}$ **do**

            $S_{ij} = \frac{D_i \cdot P_j}{\|D_i\| \|P_j\|}$

        end

    end

    $\Gamma_m \leftarrow S$

end

$\Sigma = \text{Elementwise-Mean}(\Gamma) \in N \times K$

2.4.2 Choosing Partisan Keywords

We[6] chose keywords that reflected what distinctions were important during the 2016 presidential campaign. We focus on the major party candidates and their campaign slogans and the prominent party names. We use five Republican keywords and five Democratic keywords matched pairwise by type. Four of these keywords are candidate names: "Trump," "Clinton," "Donald," and "Hillary." Two of the keywords are the candidates' Twitter handles: "realDonaldTrump" and "HillaryClinton." Two are the campaigns' slogans: "MAGA" and "StrongerTogether." Two are the party names: "Republican" and "Democrat." While other words may also serve as explicitly partisan keywords, these words suffice to represent important partisan aspects of the presidential campaign period.

We do not include words that express only disapproval or contempt for candidates or parties. For example, we do not include as a keyword "NeverTrump." We envision that use of the keywords primarily relates to positive or at least neutral sentiments toward the referent entity; we look at this more closely in Section 2.6.

2.4.3 Choosing Hyperparameters

To select hyperparameters, we[7] use an application-specific loss function developed in Wu et al. (2019). In short, Wu et al. (2019) propose choosing hyperparameters that minimize loss defined using a set of models for the number of times each user retweeted or favorited ("liked") tweets from a set of partisan accounts. Wu et al. (2019) use two definitions for the set of partisan Twitter accounts. The first set is partisan accounts retweeted by members of the U.S. House and Senate. Any account retweeted more than 90% disproportionately by one party's members in Congress and by at least 50 members of one party is considered a partisan account; We refer to these as Democratic or Republican accounts. There are 320 Democratic accounts and 328 Republican accounts. The second set of partisan accounts are those listed as influential partisan or ideological accounts by at least one of three sources (Joyce, 2016a,b; Faris et al., 2017). We refer to these as left or right accounts. There are 55 left accounts and 78 right accounts, and the two sets overlap.

Then, for each specification, we use the average of ten sample replicates (see Section 2.4.5 for more details). The count model is a zero-inflated negative binomial (ZINB) regression model (Zeileis et al., 2008; Jackman, 2017). The loss function measures whether the coefficients in the count part of the model have the correct (opposite) signs and how

[6]Walter Mebane and I jointly chose the partisan keywords.

[7]Walter Mebane primarily designed the application-specific loss function. Walter and I created the criteria for the left and right accounts, while Walter created the criteria for the Democratic and Republican accounts. Walter and Preston Due collected the counts of retweets.

strongly a test rejects the hypothesis that the coefficients are equal. We then take the cosine similarities between the ten keywords and each bio to specify the linear predictors for the means in the count and the zero-inflation parts of the ZINB regression model. For each pair of keywords $\mathcal{J} = \{$(trump, clinton), (donald, hillary), (republican, democrat), (realdonaldtrump, hillaryclinton), (maga, strongertogether)$\}$ and denoting the pairs of keywords as $j = (k_{1j}, k_{2j})$, where k_{1j} is the vector of cosine similarities of the first word in the pair across all users and k_{2j} is defined similarly for the second word in the pair, the zero-inflated negative binomial model is specified as

$$\mu_{C_{ij}} = a_{0j} + a_{1j}k_{1ij} + a_{2j}k_{2ij}$$

for the count part of the model for user i, and

$$\mu_{Z_{ij}} = b_{0j} + b_{1j}k_{1ij} + b_{2j}k_{2ij} + b_{3j}\log(m_i + 1)$$

for the zero-inflation part of the model for user i and where m_i is the total number of retweets in the timeline for user i. The expectation is that $\text{sign}(a_{1j}) \neq \text{sign}(a_{2j})$ with $\text{sign}(a_{1j}) > 0$ for counts of retweets or favorites of Republican accounts and $\text{sign}(a_{2j}) < 0$ for counts or favorites of Democratic accounts.

To measure how well the estimated coefficients match these expectations, we use an application-specific loss function that uses the indicator function $\mathcal{I}\left(\text{sign}(\hat{a}_{1j}) = sign(\hat{a}_{2j})\right)$, and the chi-square statistic $\chi\left(\text{H0} : a_{1j} = a_{2j}\right)$ for the test of the hypothesis that $a_{1j} = a_{2j}$:

$$\mathcal{L} = \sum_{j \in \mathcal{J}} \left(5\mathcal{I}\left(\text{sign}(\hat{a}_{1j}) = sign(\hat{a}_{2j})\right) - \frac{\chi\left(\text{H0} : a_{1j} = a_{2j}\right)}{\sqrt{n_j}} \right)$$

The loss is then calculated for each configuration of hyperparameters. The details of these various hyperparameter configurations can be found in Section 2.9.2 in the Supplemental Information.

Ultimately, the hyperparameters chosen for this particular application are embeddings with dimension 75, a window size of 10, an initial learning rate of 0.03 with a per-epoch decrease of 0.00025, and ignoring words that occur less than 8 times in the corpus. Word and document embeddings are simultaneously trained using PV-DBOW with skip-gram. Training lasts 100 epochs (the number of iterations through the corpus) using 8 cores. We remove stopwords, stem words, remove punctuation, remove numbers, and remove single-character words before training the embeddings. See Section 2.9.2 in the Supplemental Information for details about the loss values for other configurations.

2.4.4 Partisan Subspaces

We use the approach outlined in Section 2.3.4 to calculate the partisan association scores for all users with nonempty bios. We use multiple partisan subspaces, choosing to not collapse the subspaces into a single subspace (Bolukbasi et al., 2016; Kozlowski et al., 2019). The four partisan subspaces are as follows:

- Names Subspace: $P_{\text{names}} = \frac{V_{\text{Trump}} + V_{\text{Donald}}}{2} - \frac{V_{\text{Clinton}} + V_{\text{Hillary}}}{2}$
- Parties Subspace: $P_{\text{parties}} = V_{\text{Republican}} - V_{\text{Democrat}}$
- Handles Subspace: $P_{\text{handles}} = V_{\text{realDonaldTrump}} - V_{\text{HillaryClinton}}$
- Slogans Subspace: $P_{\text{slogans}} = V_{\text{MAGA}} - V_{\text{StrongerTogether}}$

Wu et al. (2019) describe in greater detail this choice, but the primary motivating factors are as follows. First, Wu et al. (2019) find that many of the expected analogies do not hold up among the partisan keywords. See Section 2.9.3 in the Supplemental Information for more details about the analogies calculated. Second, on substantive grounds, the support of candidates did not always match the support of the party. Exit polls show that approximately 90% of Republicans and Democrats voted for Donald Trump and Hillary Clinton, respectively (Sides et al., 2018); the American National Election Study (2019) show that approximately 44% and 38% of self-reported independents who voted did so for Donald Trump and Hillary Clinton, respectively. Third, Wu et al. (2019) found, using the partisan associations calculated using bios of Twitter users that responded to a Pew survey and answered two questions about party identification and ideology (Wojcik & Hughes, 2019), in a factor analysis that only the partisan association scores S_{parties} and S_{slogans} load positively on party identification. This analysis seems to suggest that there are substantive differences in interpretation between the partisan association scores calculated using the separate subspaces. We discuss this in Section 2.4.7 in greater detail.

2.4.5 Calculating Partisan Association Scores

To obtain the partisan association scores, we averaged across ten trained doc2vec embedding spaces using the hyperparameters selected as described in Section 2.4.3. Again, averaging across different trained embedding spaces stabilizes the calculated cosine similarity scores between the document embeddings and the partisan subspaces. For example, between pairs of replicates, the cosine similarity scores between the ten keywords and the document embeddings of user bios ranged between 0.92 to 0.96. After averaging, the correlation between pairs of averages of the ten replicates was between 0.98 to 0.99.

Table 2.3 describes the product-moment correlation coefficients between the partisan association scores calculated using the four different partisan subspaces and the "usage" subspace. The usage subspace is the average of all ten partisan keywords' embeddings. It captures to what extent users tend to use partisan language regardless of partisan direction. For more information about the joint distribution of the partisan association scores, see Section 2.9.4 in the Supplemental Information.

	usage	names	parties	handles	slogans
usage	1.00	0.0077	0.091	0.13	0.13
names		1.00	0.53	0.65	0.69
parties			1.00	0.48	0.64
handles				1.00	0.72
slogans					1.00

Table 2.3: Product-moment correlation coefficients between the partisan association scores calculated using the four different partisan subspaces and the usage subspace. $n = 194,336$.

We also show the conditional distributions of the partisan association scores given the use of three other campaign-related words: "ImWithHer", a popular hashtag used among supporters of Hillary Clinton; "MakeAmericaGreatAgain", a popular hashtag used among supports of Donald Trump, and "NeverTrump", a hashtag used by both Democrats and Republicans that did not support the candidacy of Donald Trump. Figure 2.1 shows the empirical distributions of the partisan association scores for users whose bios used one of the three terms. The distributions of the partisan association scores make sense for each term. The distributions for users who used the term "ImWithHer" tend to be on the left of users who used the term "MakeAmericaGreatAgain," and the distribution of users of "NeverTrump" is more in the center. This pattern holds for all partisan subspaces.

2.4.6 Partisan Associations and Retweets, Favorites, Hashtags, and Following Members of Congress

We[8] use zero-inflated negative binomial (ZINB) regression models to suggest that partisan association scores are measures of partisan engagement and sentiment. Specifically, the outcome variables for the regression models are the counts of the number of partisan retweets, the number of partisan favorites (or "likes"), the number of partisan hashtags used, and the

[8]Walter Mebane and Preston Due collected the counts of favorites, hashtags, and followers of members of Congress. Walter and Preston created the criteria for the hashtag types. Walter and I jointly worked on the results of this section.

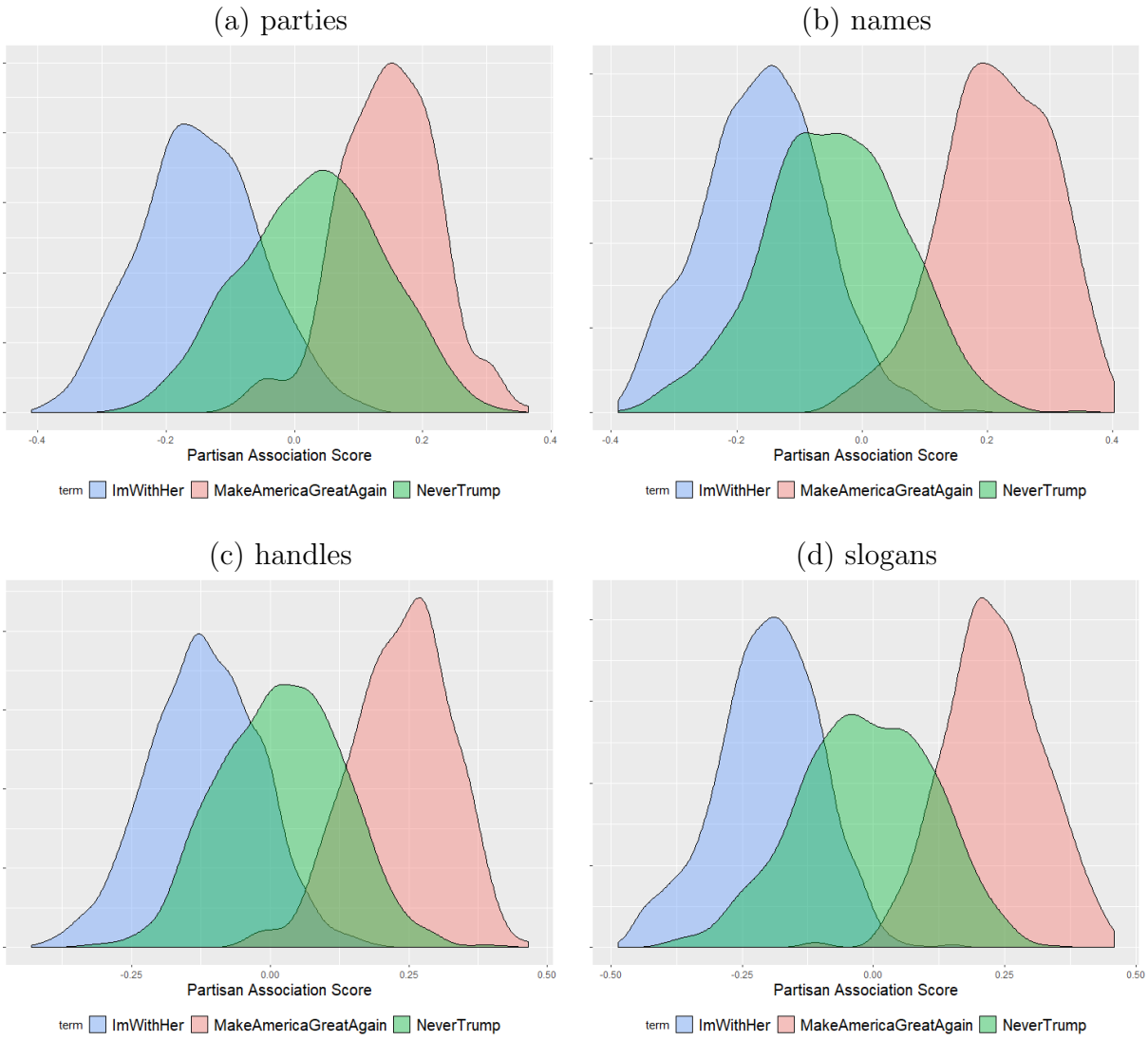


Figure 2.1: Distribution of the partisan association scores of users who used one of the out-of-sample words

number of members of Congress they follow by party. These counts cover online activity spanning the years before and after the 2016 U.S. general election period.

We use the number of retweets and favorites of accounts based on the definition of "Democratic" and "Republican" accounts described in Section 2.4.3. We also identified a set of "left" and "right" hashtags. The details about the hashtag selection process are described in Wu et al. (2019); the list of hashtags can be found in the Supplemental Information in Section 2.9.5.

The expectation is that users who have more positive partisan association scores values more frequently retweet and favorite more Republican accounts, more frequently use right-leaning hashtags, and follow more Republican members of Congress.[9] The zero-inflation part of the model detects users who do not engage in such behaviors. For partisan subspace k, the linear predictors μ_C and μ_Z for the count and zero-inflation parts of the model are, respectively,

$$\mu_{C_{ik}} = a_{0k} + a_{1k}\Sigma_{ik} + a_{2k}\texttt{usage}_i \tag{2.2}$$

$$\mu_{Z_{ik}} = b_{0k} + b_{1k}\Sigma_{ik} + b_{2k}\texttt{usage}_i + b_{3k}\log(t_i + 1) + b_{4k}\log(m_i + 1) \tag{2.3}$$

where t_i is the number of tweets and m_i is the number of either retweets, favorites, hashtags, or follows for user i. Σ_{ik} is the kth partisan association score for user i, where Σ is defined in Algorithm 2.1. We expect that $a_{1k} > 0$ for Republican/right targets, and $a_{1k} < 0$ for Democratic/left targets. We also expect that those who associate more with partisan language, regardless of direction, should participate more in partisan retweets, favorites, hashtag usage, and follow more political elites, so for all targets, we expect that $a_{2k} > 0$. It may also be the case that a partisan of one party does not engage with the accounts of the other party at all; we should see that $b_{1k} < 0$ for Republican and right targets and $b_{1k} > 0$ for Democratic and left targets. Figure 2.2 contains the results of the ZINB models.

Figure 2.2 shows that our expectations of the coefficients in the count part of the model are confirmed. The zero-inflation part of the models partly matches expectations among the left and right hashtags. We have no expectations for the coefficients of `usage` in the zero-inflation part of the models. We do see that $\hat{b}_{2k}$ is more positive for Democratic and left targets than Republican and right targets. These coefficients suggest that a higher association with partisan language is typically associated with more engagement with Republican accounts or hashtags and less with Democratic accounts or hashtags.

We repeat this analysis with the "left" and "right" accounts as defined in Section 2.4.3.

[9]This method does not factor in the possibility that these behaviors do not necessarily indicate an endorsement and that the propensity for users to engage in these social media behaviors varies.

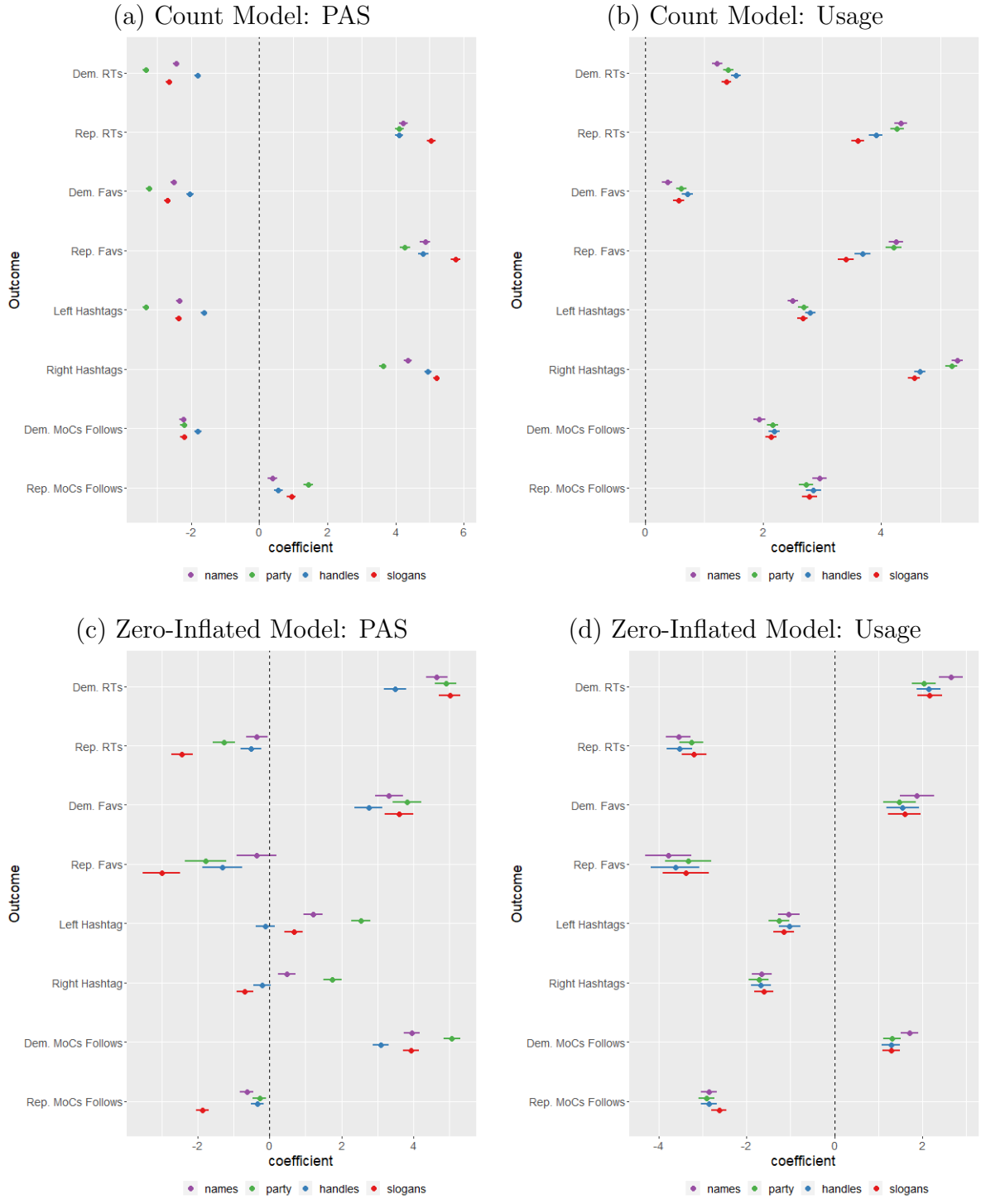


Figure 2.2: Coefficients from the zero-inflated negative binomial regressions of retweets, favorites, hashtags, and follows of members of Congress on partisan association scores (PAS) and the usage score.

The results are largely in line with Figure 2.2, and can be found in the Supplemental Information in Section 2.9.6.

2.4.7 Partisan Associations and 2018 Pew Research Center Survey Data

We[10] look at a 2018 Pew survey to see if partisan associations derived from user bios can be plausibly related to survey questions about party identification and ideology. The data are the predicted partisan association scores of a subset of survey respondents with Twitter bios and the responses to selected survey questions for "users who agreed to provide their Twitter handles" from "respondents from Iposos' KnowledgePanel, a probability-based pane of U.S. adults" that was conducted between November 21, 2018 and December 17, 2018 (Wojcik & Hughes, 2019). We supplied Pew researchers with a Python script that uses the doc2vec models trained using the 2016 bios to predict vectors from the bios of survey respondents. From this, we have partisan association scores of $1,276$ survey respondents. From the survey responses, we create two variables. For party identification, we construct a scale with five levels: Democrat, Democratic leaner, Independent, Republican leaner, and Republican. To measure ideology, we use the survey's eleven-point scale: "10" corresponds to "very liberal" and "0" corresponds to "very conservative." We flip the ideology scale such that 0 corresponds to "very liberal" and "10" corresponds to "very conservative" to keep the expected signs of the coefficients consistent. Section 2.9.7.1 in the Supplemental Information details exactly how the variables were constructed.

We first used ordered logistic regression models to examine the associations between the partisan association scores and party identification and the association between the partisan association scores and ideology. Figure 2.3 reports the coefficients of the partisan association scores and the usage score. The signs of the coefficients are as expected: a respondent with higher partisan association scores is associated with self-reporting that they lean Republican or that they report having a more conservative ideology. Moreover, the coefficients corresponding with the usage score also align with the results from Figure 2.2: more explicitly partisan bios tend to be associated with users who self-report leaning Republican or having a more conservative ideology. Although the signs of the usage scores' coefficients are positive for all four partisan association scores, it is not significant at the 95% level for the "party" and "slogans" partisan association scores when partisan leaning is the outcome variable. The coefficient on the usage score is also not significant at the 95%

[10]Walter Mebane and I jointly worked with researchers at Pew Research Center to obtain partisan association scores of survey respondents. Walter conducted the factor analysis.

level for the "party" partisan association score when ideology is the outcome variable.

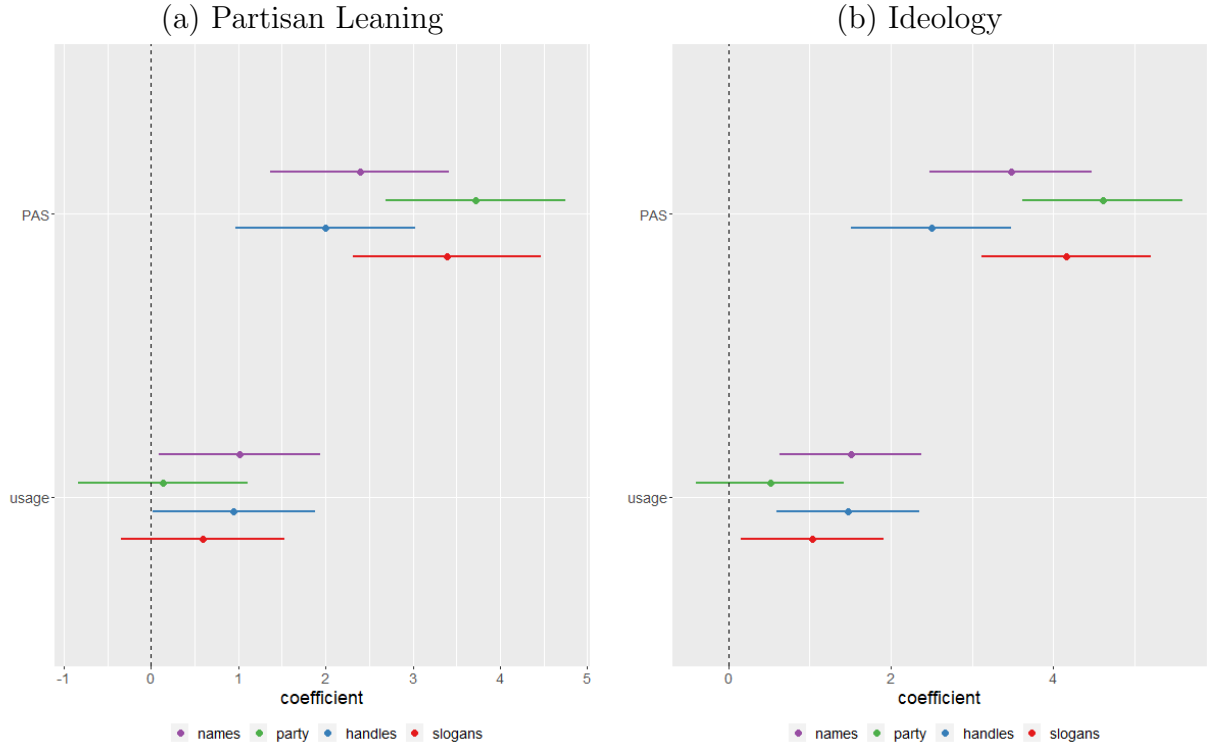


Figure 2.3: Coefficients from ordered logistic regression models of survey respondents' self-reported partisan leaning and ideology.

Wu et al. (2019) show, however, that there is a nuanced difference between the partisan identification response and the ideology response among the partisan association scores. In the one-factor model, all partisan association scores have a positive loading. With a two-factor loading, only two of the partisan association scores positively load on party identification, but all partisan association scores load negatively on the latent variable keyed to ideology. This key result motivates the use of separate partisan subspaces. For further details about the factor analysis, see Section 2.9.7.2 in the Supplemental Information.

2.4.8 Comparing Partisan Association Scores with the Bayesian Spatial Following Model

We[11] also calculate the ideal points of the Twitter users using Barberá (2015)'s Bayesian spatial following model. Again, the intuition behind the Bayesian spatial following model is

[11] Walter Mebane and Preston Due collected the set of friends for each user.

that the more Republican (Democratic) politicians or partisan accounts the user follows, the more Republican (Democratic) the user leans. Instead of estimating the political accounts' ideologies from scratch, which would require us to select a set of political accounts and require users to follow three or more of these political accounts, we estimated the ideology of users using a set of political accounts with pre-estimated ideologies. These political accounts already have their ideal points estimated using a larger set of Twitter users (Barberá, 2015). These accounts were selected in 2018.[12]

Accounts must follow at least one of the accounts from the list of political and media outlet accounts to have an ideology estimated. We estimate the ideal point (the mean of the θ parameter) and the level of political interest (the mean of the β parameter) for each user. The latter accounts for the differences in the number of political accounts each user follows (Barberá, 2015). These estimates are analogous to the partisan association scores and the usage score, respectively.

Of the 215,230 total Twitter users in this dataset, we estimated the ideology of 119,179 users using the Bayesian spatial following model. Using the bios, we calculated the partisan associations of 194,336 users. In other words, the coverage of users using bios is much greater than using network information, which is the partisan association method's most significant advantage over the Bayesian spatial following model.

Among the 108,801 users with both an estimated ideal point using the Bayesian spatial following model and partisan association scores, the estimated ideal points and the partisan association scores are positively but weakly correlated. Table 2.4 describes the correlation coefficients between the estimated ideal points of Barberá (2015), calculated using the posterior mean of the parameter θ for each user, and the four partisan association scores. The correlation coefficient between the level of political interest and the usage score is -0.02, suggesting that how explicitly partisan a bio is has little association with how many political accounts a user follows.

Using zero-inflated negative binomial models, the estimated ideal points, θ, of the Bayesian spatial following model also predict retweets of partisan accounts, favorites of partisan accounts, and usage of partisan hashtags. The coefficients of the ZINB models are in Figure 2.4. The signs of the coefficients in the count part of the model of the ideal points are as expected: users that have more positive ideal points tend to retweet and favorite "Republican" accounts and use "right" hashtags; users that have more negative ideal points tend to retweet and favorite "Democratic" accounts and use "left" hashtags.[13] However, the

[12]The list of accounts and their pre-estimated ideologies can be found here: https://github.com/pablobarbera/twitter_ideology/blob/master/2018-update/accounts-twitter-data-2018-07.csv.

[13]"Republican" and "Democratic" accounts are as defined in Section 2.4.3; "right" and "left" hashtags are defined in Section 2.9.5.

	θ	β	names	parties	handles	slogans	usage
θ	1.00	-0.60	0.20	0.25	0.17	0.25	0.13
β		1.00	-0.16	-0.20	-0.11	-0.19	-0.02
names			1.00	0.55	0.68	0.71	0.06
parties				1.00	0.50	0.66	0.18
handles					1.00	0.75	0.14
slogans						1.00	0.12
usage							1.00

Table 2.4: Correlation coefficients among the ideal points θ and β estimated using the Bayesian spatial following model (Barberá, 2015), the partisan association scores, and the usage score using the 108,801 users with both an estimated ideal point and partisan association scores.

signs of the coefficients in the count part of the model on the political interest measure, β, are unexpected for the models where the outcome variables are retweets of "Republican" accounts and favorites of "Republican" accounts. The negative sign suggests that a more politically interested user will retweet or favorite *less* "Republican" accounts, which does not match our expectations.

Figure 2.5 details the coefficients of the zero-inflated negative binomial models with the same set of outcomes using the "names" partisan association scores across the 108,801 users. The signs of the coefficients in the count part of the model are as expected. The coefficients of the zero-inflated negative binomial models with the same set of outcomes using the other three partisan association scores are detailed in Section 2.9.8 in the Supplemental Information; all signs of the coefficients in the count part of the model are as expected.

Further research is needed to understand what aspects of partisanship the Bayesian spatial following model and the partisan association scores are capturing. The correlations and the zero-inflated negative binomial regression coefficients hint that the two capture different aspects of partisanship on social media. However, what precisely these differing aspects look like is still unknown.

2.5 Application 2: Tweets about Masking During the COVID-19 Pandemic

2.5.1 Data

We apply the partisan association method to tweets about masking during the COVID-19 pandemic. Tweets were published between January 1, 2020 and July 1, 2020. We derive

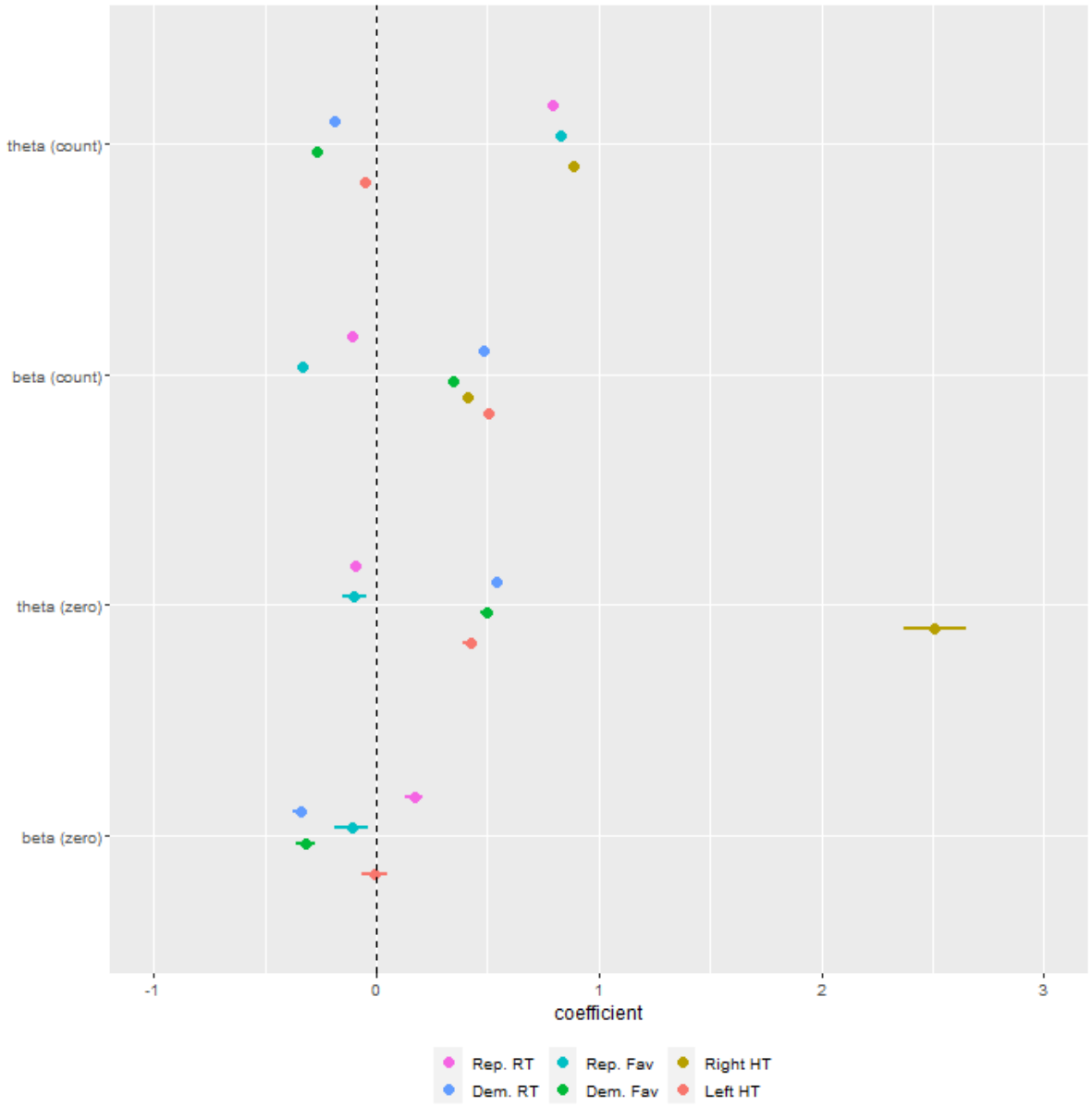


Figure 2.4: Coefficients from the zero-inflated negative binomial regressions of retweets, favorites, and hashtags on the estimated ideal points (the mean of the parameter θ for each user) and political interest (the mean of the parameter β for each user).

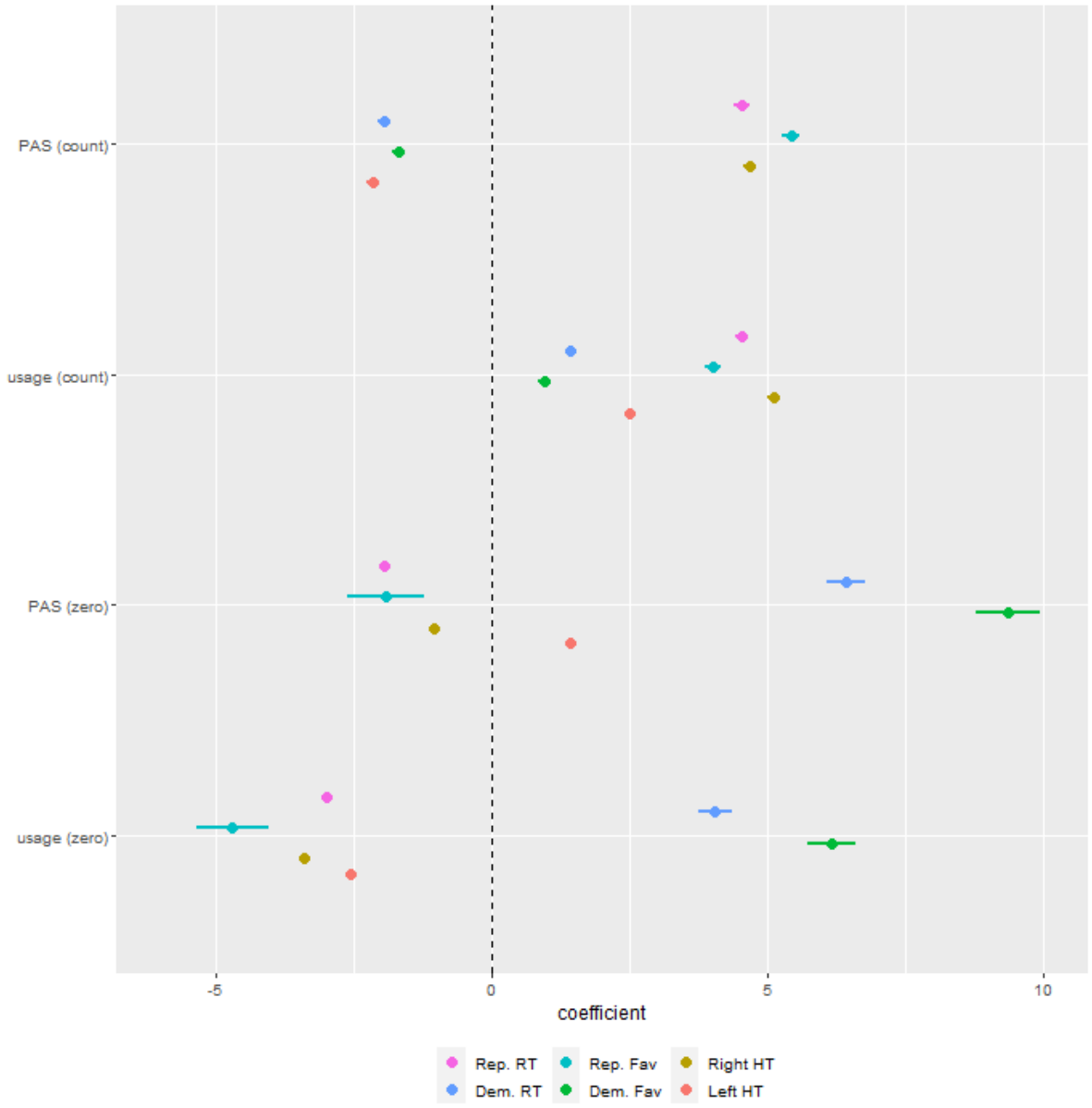


Figure 2.5: Coefficients from the zero-inflated negative binomial regressions of retweets, favorites, and hashtags on the "names" partisan association scores and the usage scores for users with estimated ideal points.

the masking tweets from the COVID-CORE dataset (Lu & Mei, 2020). The COVID-CORE dataset was created from Twitter Decahose data, which represents a 10% sample of all tweets. Lu & Mei (2020) curated a set of base keywords to select the tweets that make up the COVID-CORE dataset. Section 2.9.9 in the Supplemental Information describes the exact set of keywords. To extract all tweets mentioning masking, we searched for tweets with the phrase "mask," ignoring all characters or spaces that came before or after the word and ignoring case. We then kept only English tweets using the `lang` field in the Twitter API. In total, there were 757,732 tweets. Of these tweets, 636,110 were written by users with bios. There are 636,072 unique users with bios. For users that appear more than once in the dataset, we used their latest bio.

2.5.2 Calculating Partisan Association Scores

There is no corresponding survey data to suggest whether to combine partisan subspaces or to keep them separate. The data also did not contain any information about followers or friends, so we did not use an application-specific loss function. Therefore, for hyperparameters, we used nearly all of `gensim`'s suggested hyperparameter values (Řehůřek & Sojka, 2010). We used vectors of dimension 150, 75 epochs, a window size of 5, a starting learning rate of 0.025 decreasing down to 0.0001, a minimum count of 20 occurrences for a word to be mapped to a vector, and DBOW with skip-gram. We used the same set of preprocessing steps as described in Application 1 in Section 2.4.

To calculate partisan associations for each Twitter user, we first selected a set of keywords associated with Republicans and Democrats. Because many users negatively use terms related to Republicans or Democrats (e.g., "I dislike Democrats"), we chose terms that would most likely be hashtags or standalone slogans.[14] These are terms not typically used negatively. Moreover, the Democratic primary occurred during the period the tweets came from, and Joe Biden did not become the presumptive nominee until April 8, 2020. This meant that Democratic keywords related to candidate names represented related but distinct ideals of the Democratic Party. We chose words that supporters of any Democratic candidates would plausibly use. Specifically, we chose the terms "MAGA" and "KAG" as Republican keywords, and we chose the terms "resist" and "TheResistance" as Democratic keywords. We call this the hashtag partisan subspace. To create the usage subspace, we averaged all four terms' word embeddings. Again, the usage subspace represents how partisan a bio is, regardless of partisan direction.

[14]Section 2.6 further discusses the problem of negative usages of partisan keywords.

2.5.3 Health Advocacy Hashtags

Many tweets from this period contained health advocacy hashtags. These tweets typically implored others to wear a mask, stay at home, stay safe, and so on. Petersen & Gerken (2021) look at hashtags in an exploratory fashion on Twitter during the COVID-19 pandemic. From their findings, the hashtags we include under this category are: "SocialDistancing", "WearAMask", "MaskUp", "StayHome", "StayAtHome", "StaySafeAtHome", "StayHomeStaySafe", and "FlattenTheCurve".

A Pew Research Center survey showed that, at the beginning of the pandemic, both Democrats and Republicans were supportive of government-imposed measures to curb the coronavirus (Deane et al., 2021). But by early April 2020, 8 in 10 Democrats were concerned that restrictions would be lifted too quickly, while 5 in 10 Republicans shared this concern. This 30-point difference grew to a 40-point difference by May 2020. The expectation, then, is that more Democratic-leaning users would be more likely to use these health advocacy hashtags.

To examine this possibility, we use logistic regression. We assigned tweets that used at least one of the health advocacy hashtags to 1; we assigned other tweets to 0.[15] $8,405$ tweets used at least one health advocacy hashtag. The health advocacy hashtag usage binary variable is the outcome variable. The expectation is that the coefficient $\beta_{PAS} < 0$, as the hypothesis is that the more Republican-leaning a user is, the less likely they are to using a health advocacy hashtag. We also include usage scores as a control.

Figure 2.6 contains the coefficients and confidence intervals of the logistic regression. The coefficient on the partisan association scores (PAS) using the hashtag partisan subspace confirms the findings of Deane et al. (2021): Republican-leaning users are less likely to use health advocacy hashtags in their tweets. The coefficient on usage scores also suggests that more partisan users, in general, are less likely to use health advocacy hashtags.

We also use the same set of "left" hashtags and "right" hashtags as described in Application 1 in Section 2.4.6. We create similar binary variables for the left hashtag usage and right hashtag usage. 404 tweets contained at least one left hashtag, and $1,499$ tweets contained at least one right hashtag. These hashtags may be slightly outdated for 2020, but Donald Trump is still president, and many of the hashtags, such as "MAGA," are still relevant for the 2020 U.S. general election. Again, the expectation is that the coefficient $\beta_{PAS} < 0$ for the logistic regression where left hashtag usage is the outcome variable and $\beta_{PAS} > 0$ for the

[15] We did not use count data because, among users that used one of the health advocacy hashtags, the vast majority used 1 per tweet. Because there were very few repeat users in the COVID masking tweets dataset, almost all users would have only used 1 health advocacy hashtag if they used any health advocacy hashtags at all.

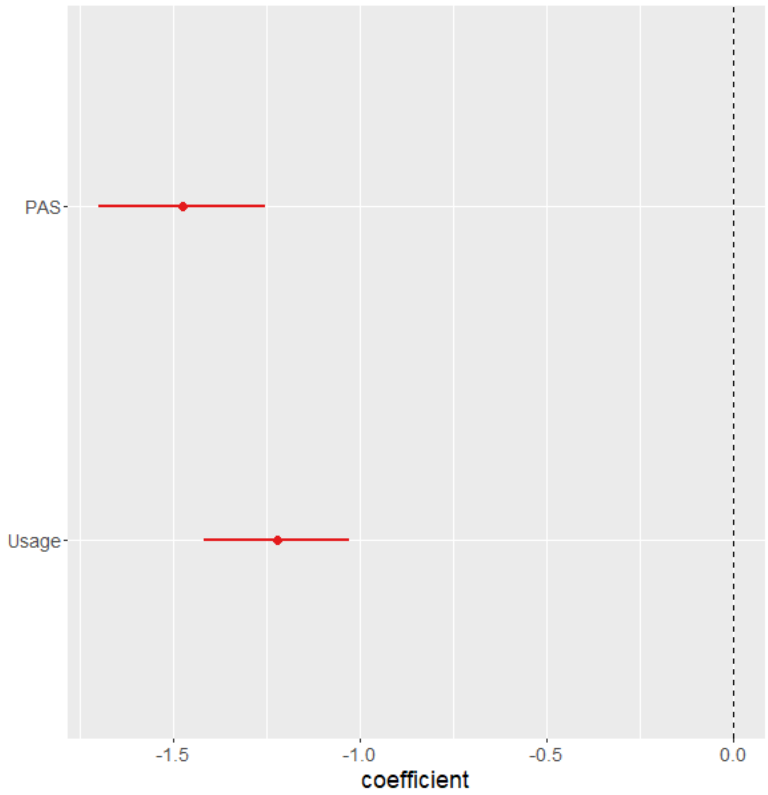


Figure 2.6: Logistic regression coefficients regressing the usage of health advocacy hashtags on partisan association scores (using the hashtag partisan subspace) and usage.

logistic regression where the right hashtag usage is the outcome variable.

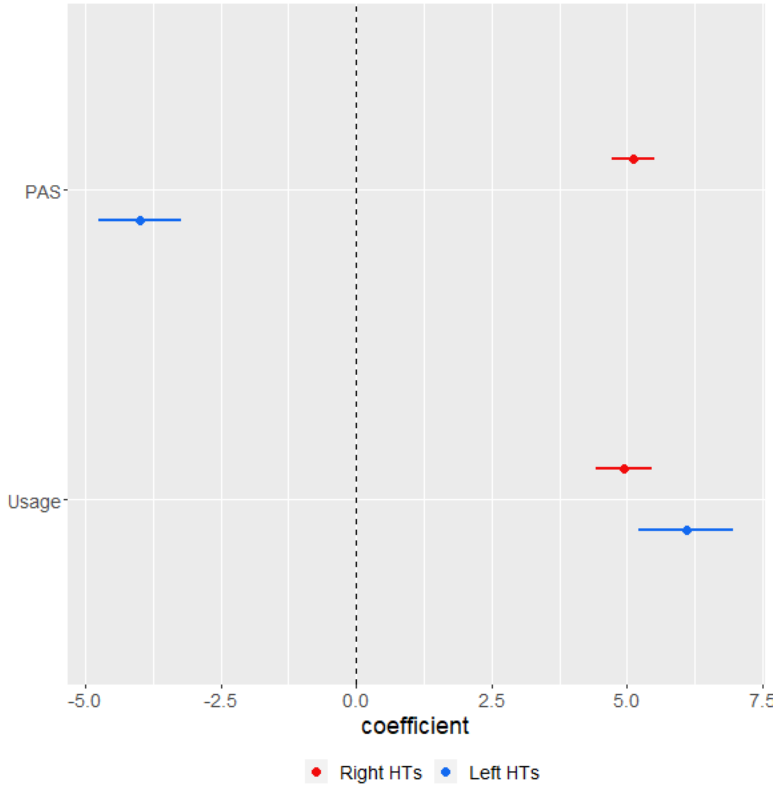


Figure 2.7: Logistic regression coefficients regressing the usage of left and right hashtags on partisan association scores (using the hashtag partisan subspace) and usage.

The coefficients shown in Figure 2.7 confirms the expectations. The coefficient on the usage score also indicates that a user is more likely to use a left or right hashtag if their bio is more explicitly partisan, regardless of partisan direction.

2.6 Negative Usages of Partisan Keywords

2.6.1 Detecting Negative Usages of Partisan Keywords

Bios that contain one or more of the partisan keywords sometimes express negativity or hostility toward the referent. In Application 1, described in Section 2.4, bios that contain one or more of the keywords occasionally expressed negativity towards one of the referents. In a sample of bios containing a keyword or related word, 18.5% contain a negative expression: 9.5% are negative towards a Republican target, 7.6% are negative towards a Democratic target, and 9% are negative towards referents from both parties.

The keywords in Application 2, described in Section 2.5, were chosen explicitly to avoid negative usages. We[16] focus the discussion on the dataset of tweets used in Application 1. Inspired by Gong et al. (2017), we develop a classifier that differentiates between negative and not-negative usages of the partisan keywords. We label keywords as used in a negative manner or not, train a support vector machine (SVM) classifier, then classify unlabeled keywords.[17] Beyond text features, the classifier also uses features describing whether both Democratic and Republican keywords are in the bio, how many Republican or Democratic keywords are used, dummy variables for keywords, the average partisan association score across all four partisan subspaces, and the average partisan association score across all four partisan subspaces with the keyword removed. For more information about the design of the SVM classifier and the labeled dataset, see Section 2.9.10 in the Supplemental information. Table 2.5 shows the confusion matrix for the negative usages SVM classifier. These results match or exceed the results found on a comparable dataset used by Gong et al. (2017).

	Precision	Recall	F1	Support
Not Negative Usage	0.90	0.92	0.91	254
Negative Usage	0.66	0.62	0.63	65
Macro Average	0.78	0.77	0.77	319

Table 2.5: Confusion matrix for the SVM classifier of negative usage of partisan keywords.

After obtaining the instances where a keyword was used negatively, we calculate four not-negative partisan subspaces and four negative partisan subspaces. In other words, keywords classified as being used negatively were appended with "_hostile" (such as "democrat_hostile"); we then calculated separate partisan association scores using the paired keywords without this appended phrase and those with it. Table 2.6 looks at the correlations between not-negative and negative partisan associations. Because the slogan "StrongerTogether" was never used in a negative manner, there are no negative partisan association scores for the slogans subspace.

We call the partisan association scores without differentiating between negative and not-negative usages the agnostic partisan association scores. Table 2.6 suggests that the not-negative partisan association scores relate to one another in similar ways as the agnostic partisan association scores do. However, the correlations among the negative partisan association scores are less clear. Because these are negative usages of the partisan keywords, we expected negative correlations. However, the magnitudes of these negative correlations vary

[16]Walter Mebane collected the tweets and labeled the usages of keywords as negative or not, with research assistance from Joseph Park and Benjamin Puzycki. Walter calculated the correlations.

[17]For an overview of support vector machines, see Chapter 12 of Hastie et al. (2009).

	not-negative				negative			
	usage	names	parties	handles	usage	names	parties	handles
usage: not-negative	1.00	0.075	0.17	0.10	0.67	0.10	-0.072	-0.064
names: not-negative		1.00	0.58	0.69	0.17	-0.080	0.062	-0.16
parties: not-negative			1.00	0.52	0.087	0.0032	-0.099	-0.020
handles: not-negative				1.00	0.19	-0.078	0.0098	-0.058
usage: negative					1.00	-0.029	-0.12	-0.039
names: negative						1.00	0.019	-0.083
parties: negative							1.00	-0.40
handles: negative								1.00

Table 2.6: Correlations between not-negative and negative partisan association scores.

and are not very strong.

Table 2.7 looks at the product-moment correlation coefficients between the agnostic and the not-negative and negative partisan association scores. The correlations show that the agnostic partisan association space is strongly and positively related to the not-negative partisan association scores: the product-moment correlation coefficients between corresponding agnostic and not-negative partisan association scores range from 0.95 to 0.98. Moreover, the agnostic partisan association scores relate negatively to the corresponding negative partisan association scores.

		not-negative					negative			
		usage	names	parties	handles	slogans	usage	names	parties	handles
agnostic	usage	0.98	0.10	0.15	0.11	0.11	0.75	0.12	-0.062	-0.066
	names	0.10	0.95	0.57	0.66	0.69	0.015	-0.25	0.12	-0.14
	parties	0.096	0.57	0.97	0.50	0.66	0.12	0.0055	-0.16	0.014
	handles	0.11	0.68	0.50	0.98	0.71	0.22	-0.11	0.061	-0.16
	slogans	0.12	0.75	0.67	0.73	0.98	0.15	0.059	-0.0096	-0.16

Table 2.7: The product-moment correlation coefficients between the agnostic partisan association scores/usage score, the not-negative partisan association scores/usage score, and the negative partisan association/usage score.

These results suggest that the negative usages of keywords do not significantly affect the partisan association scores, at least for this particular application. To see if this is the case in downstream analysis, we reanalyze the following of members of Congress using the zero-inflated negative binomial model, as done in Section 2.4.6.

2.6.2 Previous Analysis Revisited: Following Members of Congress

The models used are described fully in Equations 2.2 and 2.3. We[18] expect $a_{1k} < 0$ for agnostic and not-negative partisan association scores and $a_{1k} > 0$ for negative partisan association scores when the outcome variable is the number of Democratic members of Congress followed; likewise, we expect $a_{1k} > 0$ for agnostic and not-negative partisan association scores and $a_{1k} < 0$ for negative partisan association scores when the outcome variable is the number of Republican members of Congress followed. We expect $a_{2k} > 0$ for all partisan association scores. Figure 2.8 shows the coefficients of the count model part of the ZINB regression model.

The results confirm that $a_{2k} > 0$ as expected. The results also confirm that $a_{1k} < 0$ for agnostic and not-negative partisan association scores and $a_{1k} > 0$ for negative partisan association scores when the outcome is the number of Democratic Congressional members followed. However, we see that $a_{1k} > 0$ for agnostic and not-negative partisan association scores and $a_{1k} < 0$ for all negative partisan association scores except for the names subspace. Many Republicans expressed discontent with Donald Trump's candidacy in the 2016 election, so this pattern may reflect the activities of these individuals.

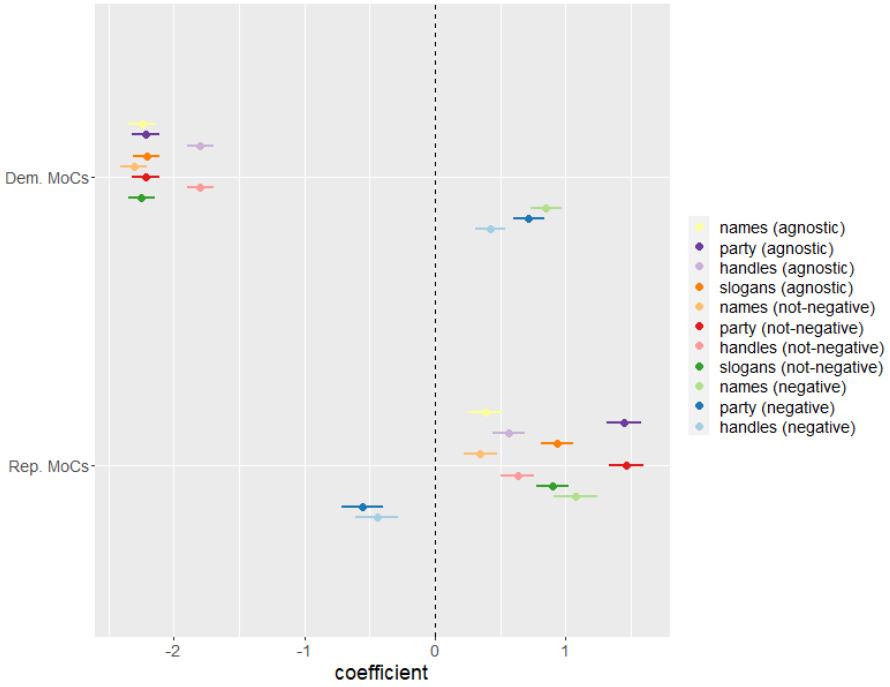
2.7 Conclusion and Future Directions

This paper proposes using the bios of Twitter users to learn about users' partisan associations. Previous approaches of calculating partisan association-like measures—such as using an ideal point model—rely on explicitly partisan behaviors on Twitter that only a minority of users engage in. The partisan association method proposed in this paper can calculate a partisan association score for any individual who has a bio, even if the bio does not contain any explicitly partisan words. It only requires that *some* users in the corpus of tweets use explicitly partisan words in their bios.

The partisan association method is straightforward and does not require many computational resources. First, select a set of partisan keywords relevant to the time period of the tweets. Then, train document and word embeddings in the same embedding space using doc2vec (Le & Mikolov, 2014), where the document embeddings are representations of the user bios. Next, to form the partisan subspace(s), either take pairs of partisan keywords and take the difference between their corresponding word embeddings or average the corresponding word embeddings for each party and then take the difference. Finally, to calculate the

[18]Walter Mebane primarily conducted the analysis in this section.

(a) Partisan Association Scores



(b) Usage

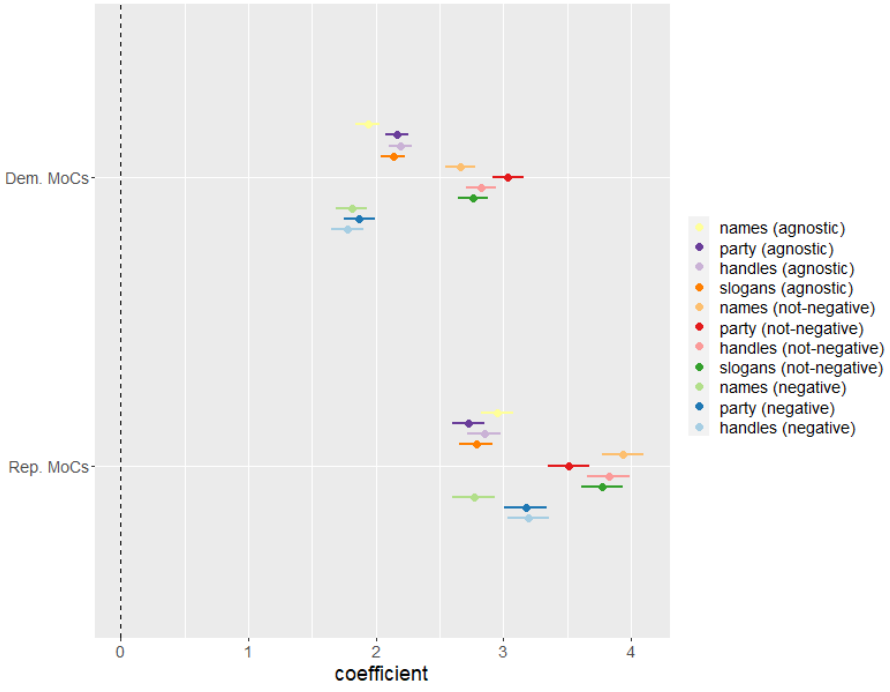


Figure 2.8: Coefficients from the zero-inflated negative binomial models regressing follows of Democratic members of Congress and Republican members of Congress (as the outcome variables) on agnostic, not-negative, and negative partisan association scores.

partisan association scores, calculate the cosine similarity between the partisan subspace(s) and the embeddings of the user bios.

The method is also intuitive. Distributed word embeddings learn which words commonly occur around partisan words. These non-partisan words that surround a partisan word are often interests, hobbies, and job titles. Although these referents are not explicitly partisan, they are inflected in partisan manners. Users who only provide non-partisan words can still, perhaps unwittingly, implicitly signal their partisan associations. We call it partisan "associations" (as opposed to, say, ideology or party identification) because the heart of the method learns the associations between partisan and non-partisan words. Moreover, partisan associations have no notion of being an Independent. A user who falls near the center is simply someone who uses words in their bio that are neither strongly associated with words used by Republican or Democratic users. How it is related to other notions of ideology and party identification remains an open research question, although the analysis in Section 2.4.7 suggests that these concepts are connected.

We apply the method to two datasets. The first dataset is a corpus of tweets posted between October 1, 2016 and November 8, 2016. Mebane et al. (2018) used this dataset to study reported election incidents during the 2016 U.S. general election. The partisan association method is strongly associated with partisan behaviors on Twitter: it correlates with which partisan accounts users retweet and favorite, which partisan hashtags are used in tweets, and which members of Congress users follow. It also correlates with self-reported party identification and ideology scales taken from a 2018 Pew Research Center survey about social media use (Wojcik & Hughes, 2019). The second dataset is a corpus of tweets written during the COVID-19 pandemic about masking, a subject that is, unlike an election, not explicitly partisan. The partisan association method predicts that more Democratic-leaning users are more likely to use health advocacy hashtags, such as #MaskUp or #StayAtHome, matching survey findings that Democrats were more likely to support government-imposed restrictions on curbing the spread of the coronavirus (e.g., Deane et al., 2021). We also develop a support vector machine classifier that aims to study how negative usages of partisan keywords, such as expressing discontent about a party or candidate, can affect the method.

There are still many future directions of the project. First, there are no confidence regions for the partisan association scores. Bootstrapping methods typically used with word embedding methods (e.g., Kozlowski et al., 2019; Rodriguez et al., 2021) present a challenge with the partisan association method because there are as many document embeddings as there are confidence regions to estimate. Second, there are currently better natural language processing methods for detecting negative usages of words, such as sarcasm and hostility detection. These approaches typically use pretrained language models such as bidirectional

encoder representations from transformers (Devlin et al., 2018). At the moment, Wu et al. (2020) are working on a paper developing a new negativity-detection approach with partisan words. Using the same dataset, they achieve a macro F1 score of 0.89—a major improvement over the macro F1 score of 0.77 using the SVM classifier. Using a better negativity detection method may drastically change the product-moment correlation coefficients found in Tables 2.6 and 2.7. Lastly, it is still an open question of whether a minimum number of users need to use explicitly partisan words before the method breaks down. In Application 1, only approximately 2% of users use at least one partisan keyword; yet, this seems to be enough to calculate high-quality partisan association scores for that set of users.

As literary theorist Mikhail Bakhtin argued, words uttered are shaped both by the world the speaker lives in and the audience of the speaker (Bahktin, 1982). (Most) Twitter users do not utter words randomly but use specific words to describe the world they live in and what they feel is most important to convey to the readers of their timelines. Their purposeful actions make using neural word embedding methods to compute partisan associations for users based on their bios feasible. The partisan associations have properties we would like to see in measures of partisan engagement and sentiment.

2.8 References

Abramowitz, Alan I. & Webster, Steven (2016). The rise of negative partisanship and the nationalization of u.s. elections in the 21st century. *Electoral Studies*, *41*, 12–22.

American National Election Study (2019). American national election study 2016 time series. URL: https://electionstudies.org/data-center/2016-time-series-study/.

Bahktin, Mikhail M. (1982). Discourse in the novel. In M. Holquist (Ed.), *The Dialogic Imagination: Four Essays* (pp. 259–422). Austin: University of Texas Press. Translated by Caryl Emerson and Michael Holquist.

Barberá, Pablo (2015). Birds of the same feather tweet together: Bayesian ideal point estimation using twitter data. *Political Analysis*, *23*(1), 76–91.

Barberá, Pablo (2016). Less is more? how demographic sample weights can improve public opinion estimates based on twitter data. Working Paper.

Barberá, Pablo, Casas, Andreu, Nagler, Jonathan, Egan, Patrick J., Bonneau, Richard, Jost, John T., & Tucker, Joshua A. (2019). Who leads? who follows? measuring issue attention and agenda setting by legislators and the mass public using social media data. *113*(4), 883–901.

Bolukbasi, Tolga, Chang, Kai-Wei, Zou, James, Saligrama, Venkatesh, & Kalai, Adam (2016). Man is to computer programmer as woman is to homemaker? debiasing word embeddings.

Bond, Robert & Messing, Solomon (2015). Quantifying social media's political space: Estimating ideology from publicly revealed preferences on facebook. *109*(1), 62–78.

Cichocka, Aleksandra, Bilewicz, Michał, Jost, John T., Marrouch, Natasza, & Witkowska, Marta (2016). On the grammar of politics—or why conservatives prefer nouns. *Political Psychology*, *37*(6), 799–815.

Clinton, Joshua, Jackman, Simon, & Rivers, Douglas (2004). The statistical analysis of roll call data. *American Political Science Review*, *98*(2), 355–370.

Deane, Claudia, Parker, Kim, & Gramlich, John (2021). A year of U.S. public opinion on the coronavirus pandemic. Pew Research Center.

Deerwester, Scott, Dumais, Susan T., Furnas, George W., Landauer, Thomas K., & Harshman, Richard (1990). Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, *41*(6), 391–407.

Denny, Matthew J. & Spirling, Arthur (2018). Text preprocessing for unsupervised learning: Why it matters, when it misleads, and what to do about it. *Political Analysis*, *26*(2), 168–189.

Devlin, Jacob, Chang, Ming-Wei, Lee, Kenton, & Toutanova, Kristina (2018). BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, *abs/1810.04805*. URL: http://arxiv.org/abs/1810.04805.

Faris, Robert, Roberts, Hal, Etling, Bruce, Bourassa, Nikki, Zuckerman, Ethan, & Benkler, Yochai (2017). Partisanship, propaganda, and disinformation: Online media and the 2016 u.s. presidential election. *Berkman Klein Center Research Publication*.

Firth, John Rupert (1957). *Studies in Linguistic Analysis*. Blackwell.

Gentzkow, Matthew, Shapiro, Jesse M., & Taddy, Matt (2019). Measuring group differences in high-dimensional choices: Method and application to congressional speech. *Econometrica*, *87*(4), 1307–1340.

Goggin, Stephen, Henderson, John A., & Theodoridis, Alexander G. (2019). What goes with red and blue? mapping partisan and ideological associations in the minds of voters. *Political Behavior*, *Online First*. URL: https://doi-org.proxy.lib.umich.edu/10.1007/s11109-018-09525-6.

Gong, Hongyu, Bhat, Suma, & Viswanath, Pramod (2017). Geometry of compositionality. *Proceedings of the AAAI Conference on Artificial Intelligence*, *31*(1).

Graham, Jesse, Nosek, Brian A., & Haidt, Jonathan (2012). The moral stereotypes of liberals and conservatives: Exaggeration of differences across the political spectrum. *PLOS ONE*, *7*(12), 1–13.

Gutmann, Michael & Hyvärinen, Aapo (2010). Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In Teh, Yee Whye & Titterington, Mike (Eds.), *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, (pp. 297–304)., Chia Laguna Resort, Sardinia, Italy. PMLR.

Haieshutter-Rice, Dan, Neuner, Fabian G., & Soroka, Stuart (2021). Cued by culture: Political imagery and partisan evaluations. *Political Behavior*.

Hastie, Trevor, Tibshirani, Robert, & Friedman, Jerome (2009). *Elements of Statistical Learning: Data Mining, Inference, and Prediction* (2 ed.). Springer.

Hetherington, Marc & Weiler, Jonathan (2018). *Prius or Pickup?: How the Answers to Four Simple Questions Explain America's Great Divide*. Boston: Houghton Mifflin Harcourt.

Huddy, Leonie, Mason, Lilliana, & Aarøe, Lene (2015). Expressive partisanship: Campaign involvement, political emotion, and partisan identity. *109*(1), 1–17.

Jackman, Simon (2017). *pscl: Classes and Methods for R Developed in the Political Science Computational Laboratory*. Sydney, New South Wales, Australia: United States Studies Centre, University of Sydney. R package version 1.5.2, URL: https://github.com/atahk/pscl/.

Joyce, Gemma (2016a). Ranked: The most influential republicans on twitter.

Joyce, Gemma (2016b). React: The most influential democrats on twitter.

Kinder, Donald R. & Kalmoe, Nathan P. (2017). *Neither Liberal nor Conservative: Ideological Innocence in the American Public*. Chicago.

Klar, Samara (2014). Partisanship in a social setting. *58*(3), 687–704.

Kozlowski, Austin C., Taddy, Matt, & Evans, James A. (2019). The geometry of culture: Analyzing the meanings of class through word embeddings. *American Sociological Review*, *84*(5), 905–949.

Laver, Michael, Benoit, Kenneth, & Garry, John (2003). Extracting policy positions from political texts using words as data. *American Political Science Review*, *92*.

Le, Quoc V. & Mikolov, Tomas (2014). Distributed representations of sentences and documents.

Lowe, William (2008). Understanding wordscores. *Political Analysis*, *16*.

Lu, Xuan & Mei, Qiaozhu (2020). COVID-CORE social media dataset.

Martin, Andrew D., Quinn, Kevin M., Park, Jong Hee, Vieilledent, Ghislain, Malecki, Michael, Blackwell, Matthew, Poole, Keith, Reed, Craig, Goodrich, Ben, Ihaka, Ross, The R Development Core Team, The R Foundation, L'Ecuyer, Pierre, Matsumoto, Makoto, & Nishimura, Takuji (2018). *MCMCpack: Markov Chain Monte Carlo (MCMC) Package*. https://CRAN.R-project.org/package=MCMCpack.

Mason, Lilliana (2018). *Uncivil Agreement: How Politics Became Our Identity.* Chicago: University of Chicago Press.

McConnell, Christopher, Margalit, Yotam, Malhotra, Neil, & Levendusky, Matthew (2018). The economic consequences of partisanshipin a polarized era. *62*(1), 5–18.

Mebane, Jr., Walter R., Wu, Patrick, Woods, Logan, Klaver, Joseph, Pineda, Alejandro, & Miller, Blake (2018). Observing election incidents in the united states via twitter: Does who observes matter? Paper presented at the 2018 Annual Meeting of the Midwest Political Science Association, Chicago, April 5–8, 2018.

Mikolov, Tomas, Chen, Kai, Corrado, Greg S., & Dean, Jeffrey (2013). Efficient estimation of word representations in vector space.

Mikolov, Tomas, Sutskever, Ilya, Chen, Kai, Corrado, Greg, & Dean, Jeffrey (2013). Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, (pp. 3111–3119)., Red Hook, NY, USA. Curran Associates Inc.

Morin, Frederic & Bengio, Yoshua (2005). Hierarchical probabilistic neural network language model. In Cowell, Robert G. & Ghahramani, Zoubin (Eds.), *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, volume R5 of *Proceedings of Machine Learning Research*, (pp. 246–252). PMLR. Reissued by PMLR on 30 March 2021.

Park, Sungjoon, Bak, JinYeong, & Oh, Alice (2017). Rotated word vector representations and their interpretability. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, (pp. 401–411)., Copenhagen, Denmark. Association for Computational Linguistics.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Pennington, Jeffrey, Socher, Richard, & Manning, Christopher D. (2014). GloVe: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, (pp. 1532–1543).

Petersen, Kai & Gerken, Jan M. (2021). #covid-19: An exploratory investigation of hashtag usage on twitter. *Health Policy*, *125*(4), 541–547.

Poole, Keith (2005). *Spatial Models of Parliamentary Voting*. Cambridge University Press.

Quinn, Kevin (2004). Bayesian factor analysis for mixed ordinal and continuous responses. *Political Analysis*, *12*(4), 338–353.

Řehůřek, Radim & Sojka, Petr (2010). Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, (pp. 45–50)., Valletta, Malta. ELRA.

Rodriguez, Pedro L. & Spirling, Arthur (2021). Word embeddings: What works, what doesn't, and how to tell the difference for applied research. Forthcoming.

Rodriguez, Pedro L., Spirling, Arthur, & Stewart, Brandon M. (2021). Embedding regression: Models for context-specific description and inference.

Rumelhart, David E., Hinton, Geoffrey E., & Williams, Ronald J. (1986). Learning representations by back-propagating errors. *Nature*, *323*(6088), 533–536.

Salton, Gerard (1962). Some experiments in the generation of word and document associations. In *Proceedings of the December 4-6, 1962, Fall Joint Computer Conference*, AFIPS '62 (Fall), (pp. 234–250)., New York, NY, USA. Association for Computing Machinery.

Sides, John, Tesler, Michael, & Vavreck, Lynn (2018). *Identity Crisis: The 2016 Presidential Campaign and the Battle for the Meaning of America*. Princeton: Princeton University Press.

Slapin, Jonathan B. & Proksch, Sven-Oliver (2008). A scaling model for estimating time-series party positions from texts. *American Journal of Political Science*, *52*.

Steinert-Threlkeld, Zachary C. (2018). *Twitter as Data (Elements in Quantitative and Computational Methods for the Social Sciences)*. Cambridge: Cambridge University Press.

Temporão, Mickael, Kerckhove, Corentin Vande, van der Linden, Clifton, Dufresne, Yannick, & Hendrickx, Julien M. (2018). Ideological scaling of social media users: A dynamic lexicon approach. *Political Analysis*, *26*(4), 457–473.

Wittgenstein, Ludwig (2009). *Philosophical Investigations*. Wiley-Blackwell.

Wojcik, Stefan & Hughes, Adam (2019). Sizing up twitter users. Pew Research Center.

Wu, Patrick Y., Mebane, Jr., Walter R., Woods, Logan, Klaver, Joseph, & Due, Preston (2019). Partisan associations of twitter users based on their self-descriptions and word embeddings.

Wu, Patrick Y., Woods, Logan T., & Mebane, Jr., Walter R. (2020). Identifying hostile usages of partisan words on twitter using bert. Working Paper.

Zeileis, Achim, Kleiber, Christian, & Jackman, Simon (2008). Regression models for count data in R. *Journal of Statistical Software*, *27*(8). URL: http://www.jstatsoft.org/v27/i08/.

2.9 Supplemental Information

2.9.1 A Primer on Two-Layer Neural Networks with No Activation Functions

A two-layer feedforward neural network is defined as $\mathbf{s} = (\mathbf{XV})\mathbf{U}$, where $\mathbf{X}$ is the one-hot vector of dimension $1 \times |V|$, $\mathbf{V}$ is a weight matrix of dimension $|V| \times d$, and $\mathbf{U}$ is a weight matrix of dimension $d \times |V|$. $|V|$ is the number of unique words and d is the number of hidden nodes chosen by the researcher. For word2vec, d is the dimension of the word embeddings. $\mathbf{s}$ is a vector containing the score of each unique word for input $\mathbf{X}$. The scores are inputted into a softmax function, which is the loss function. This setup matches the setup found in Equation 2.1. Word2vec does not use an activation function (Mikolov et al., 2013).

The ultimate goal of the model is to learn weights that minimize prediction error. The weights are learned using backpropagation (Rumelhart et al., 1986). To calculate the gradient of the loss function, we calculate the gradient of the loss function with respect to each weight one layer at a time, iterating from the last layer to the first layer. Weights are then updated using a gradient descent step.

2.9.2 Application 1: Hyperparameter Selection

Table 2.8 contains the list of hyperparameter specifications assessed using the application-specific loss function described in Section 2.4.3. Table 2.9 describes the loss function for each configuration.

2.9.3 Application 1: Analogies

One of the key properties of distributed word embeddings is their linearity property: simple analogies can be solved using vector addition and subtraction. For example, Mikolov et al. (2013) show that $V_{\text{Paris}} - V_{\text{France}} + V_{\text{Italy}} \approx V_{\text{Rome}}$, where the $\approx$ means the most similar vector measured by cosine similarity. Table 2.10 lists the analogies that we calculated using the partisan keywords. We note when a word was not in the top 10 most similar words.

The subspaces usually perform as expected. Although it did not always return the expected word, the closest word vector was usually a Republican-associated word. The parties subspace, however, performed the worst. The party names seem to be capturing a different aspect of partisan expression compared to the other partisan subspaces. We also see that combining the names together performs better than using the names individually, suggesting that it may not necessarily be "Trump" and "Clinton" that are the optimal pair: "Trump"

label	hyperparameter description
defaults	n=300, window=10, min_count=8, dbow, alpha=.03, remove stopwords, stemming, remove punctuation, remove numbers, remove single words, workers=20
spec 11	n = 150
spec 12	changing window size from 10 to 5
spec 13	changing window size from 10 to 15
spec 14	changing minimum count from 8 to 3
spec 15	changing minimum count from 8 to 13
spec 16	changing distributed bag-of-words (dbow) to distributed memory (dm)
spec 17	changing alpha=0.03 to alpha=0.08
spec 18	not removing stopwords
spec 19	not stemming words
spec 20	not removing punctuation
spec 21	not removing numbers
spec 22	not removing single words
spec 23	not removing stopwords, not stemming words
spec 24	not removing stopwords, not stemming words, not removing single words
spec 25	no preprocessing
spec 26	no punctuation but removing hashtags and @ signs on keywords
spec 27	no preprocessing at all (except lowercasing) but removing hashtags and @ signs on keywords
spec 28	no preprocessing at all, including no lowercasing, but removing hashtags and @ signs on keywords (and lowercased keywords)
spec 29	n = 75
spec 30	n = 500
spec 31	not removing stopwords and n = 150
spec 32	not stemming words and n = 150
spec 33	not removing stopwords and not stemming words and n = 150
spec 34	n=50
spec 35	n=37
spec 36	n=25
spec 37	n = 75, changing minimum count from 8 to 3
spec 38	n = 75, changing minimum count from 8 to 13
spec 39	n = 75, changing minimum count from 8 to 3, not removing single words
spec 40	n = 75, changing minimum count from 8 to 13, not removing single words
spec 41	changing minimum count from 8 to 3, not removing single words
spec 42	changing minimum count from 8 to 13, not removing single words
spec 44	n = 75, 8 workers instead of 20
spec 45	n = 75, 1 worker (8 specified seeds) instead of 20

Table 2.8: Hyperparameter specifications assessed

Note: each specification includes the default hyperparameter settings except for the noted changed details. “spec 1” is the defaults. There were no specifications designated spec 2–10.

	Democratic Republican			left right		
label	retweet	favorite	sum	retweet	favorite	sum
defaults	-77.3	-65.6	-142.8	-87.5	-78.3	-165.8
spec 11	-90.3	-76.8	-167.1	-102.2	-91.8	-194.0
spec 12	-75.9	-62.7	-138.6	-83.2	-75.0	-158.2
spec 13	-75.1	-63.2	-138.4	-84.5	-75.0	-159.4
spec 14	-77.4	-66.5	-143.9	-88.1	-79.5	-167.6
spec 15	-79.0	-67.3	-146.4	-89.3	-80.9	-170.1
spec 16	11.0	7.3	18.2	0.6	5.3	5.9
spec 17	-38.7	-34.4	-73.1	-46.0	-44.1	-90.1
spec 18	-74.7	-60.6	-135.3	-81.8	-72.9	-154.7
spec 19	-74.3	-61.4	-135.7	-83.5	-74.3	-157.8
spec 20	-60.1	-51.2	-111.3	-70.4	-60.4	-130.8
spec 21	-69.9	-59.5	-129.4	-80.3	-70.9	-151.1
spec 22	-75.4	-63.9	-139.3	-86.9	-75.3	-162.2
spec 23	-67.2	-56.1	-123.3	-77.1	-68.4	-145.5
spec 24	-71.7	-59.1	-130.8	-79.9	-73.1	-153.0
spec 25	-59.5	-52.2	-111.7	-67.7	-64.2	-131.9
spec 26	-66.3	-57.7	-123.9	-79.0	-67.7	-146.7
spec 27	-58.9	-51.3	-110.2	-67.0	-63.1	-130.1
spec 28	-62.2	-52.9	-115.1	-72.4	-65.9	-138.3
spec 29	-95.1	-80.9	-176.0	-107.4	-99.0	-206.4
spec 30	-64.3	-55.4	-119.7	-74.7	-65.4	-140.1
spec 31	-89.9	-75.1	-164.9	-100.9	-90.9	-191.8
spec 32	-83.3	-70.2	-153.5	-95.0	-86.2	-181.2
spec 33	-82.3	-69.5	-151.8	-93.5	-85.0	-178.5
spec 34	-91.8	-80.0	-171.8	-105.6	-97.3	-202.9
spec 35	-91.9	-81.5	-173.3	-105.9	-98.9	-204.8
spec 36	-89.5	-83.2	-172.7	-101.2	-100.7	-201.9
spec 37	-91.6	-78.1	-169.7	-105.3	-94.5	-199.8
spec 38	-91.6	-76.7	-168.3	-104.2	-94.0	-198.2
spec 39	-70.8	-61.6	-132.4	-84.6	-71.8	-156.4
spec 40	-93.6	-80.0	-173.6	-108.6	-97.3	-205.9
spec 41	-93.3	-79.1	-172.4	-105.9	-95.6	-201.5
spec 42	-68.4	-60.0	-128.4	-80.5	-69.3	-149.8
spec 44	-102.74	-93.81	-196.55	-89.49	-77.11	-166.60

Table 2.9: Hyperparameter loss function values

Table 2.10: Analogies among the partisan keywords

Analogy Arithmetic	Closest Word Vector		Cosine Similarity	
	Actual	Expected	Actual	Expected
Trump - Clinton + Hillary	MAGA	Donald	0.815	0.777
Trump - Clinton + Democrat	Republican	Republican	0.694	0.694
Trump - Clinton + hillaryclinton	TrumpTrain	RealDonaldTrump	0.629	0.609
Trump - Clinton + StrongerTogether	ImWithYou	MAGA	0.647	0.624
Donald - Hillary + Clinton	Trump	Trump	0.757	0.757
Donald - Hillary + Democrat	Republican	Republican	0.742	0.742
Donald - Hillary + hillaryclinton	realdonaldtrump	realdonaldtrump	0.606	0.606
Donald - Hillary + StrongerTogether	Trump	MAGA	0.693	—
Republican - Democrat + Clinton	Donald	Trump	0.677	0.653
Republican - Democrat + Hillary	Clinton	Donald	0.745	0.731
Republican - Democrat + hillaryclinton	thedemocrat	realdonaldtrump	0.640	0.544
Republican - Democrat + StrongerTogether	NeverTrump	MAGA	0.659	—
realdonaldtrump - hillaryclinton + Clinton	Trump	Trump	0.724	0.724
realdonaldtrump - hillaryclinton + Hillary	Trump	Donald	0.779	0.735
realdonaldtrump - hillaryclinton + Democrat	Republican	Republican	0.658	0.658
realdonaldtrump - hillaryclinton + StrongerTogether	TrumpPence	MAGA	0.698	0.672
MAGA - StrongerTogether + Clinton	TrumpPence	Trump	0.718	0.718
MAGA - StrongerTogether + Hillary	TrumpPence	Donald	0.797	0.714
MAGA - StrongerTogether + Democrat	conservative	Republican	0.749	0.691
MAGA - StrongerTogether + hillaryclinton	TrumpPence	realdonaldtrump	0.711	0.654
(Trump + Donald - Clinton - Hillary)/2 + Democrat	Republican	Republican	0.754	0.754
(Trump + Donald - Clinton - Hillary)/2 + hillaryclinton	realdonaldtrump	realdonaldtrump	0.644	0.644
(Trump + Donald - Clinton - Hillary)/2 + StrongerTogether	ImWithYou	MAGA	0.658	0.614

Note: [a] Not one of the top 10 closest words.

might be better paired with "Hillary."

Because the partisan subspaces seem to be capturing different aspects of partisan association, we eschew generating an overall partisan association score using an averaged partisan subspace.

2.9.4 Application 1: Further Details About the Joint Distribution of the Partisan Association Scores

Figure 2.9 shows the pairwise distributions between the partisan association scores and the usage score; Figure 2.10 shows the pairwise distributions between the partisan association scores. Walter Mebane created these plots.

2.9.5 Application 1: Left and Right Hashtags

The details about the hashtag selection can be found in Wu et al. (2019). Walter Mebane and Preston Due developed the process of identifying these hashtags. The left hashtags, in all lowercase, are: "aprx", "smartnews", "stoprush", "goptaxscam", "medicareforall", "trumpcare", "impeachtrump", "trumprussia", "lgbtq", "theresistance", "trumpshutdown", "aca", "gunsense", "resistance", "imwithher", "demsinphilly", "womensmarch2018", "dreamers", "shitholepresident", "resist", "gopshutdown", "taxscambill", "removenunes", "singlepayer", "notmypresident", "feelthebern", and "lgbt". The right hashtags, in all lowercase, are: "pjnet", "trumpadmin", "ccot", "qanon", "schumershutdown", "tucker", "iranprotests", "prolife", "buildthewall", "boycottnfl", "lockherup", "presidenttrump", "americafirst", "tcot", "draintheswamp", "releasethememo", "deepstate", "fisamemo", "teaparty", "liberals", "trumptrain", "makeamericagreatagain", "winning", "ldsconf", "trump2016", "maga", "fisa", "hannity", "fakenews", "benghazi", "fakenewsawards", "democrats", "foxnews", "isis", "neverhillary", and "taxreform".

2.9.6 Application 1: Retweets and Favorites of "Left" and "Right" Twitter Accounts

We repeat the analysis from Section 2.4.6 using the retweets and favorites of left and right Twitter accounts. Figure 2.11 contains the results. The results are largely in line with the results found in Figure 2.2.

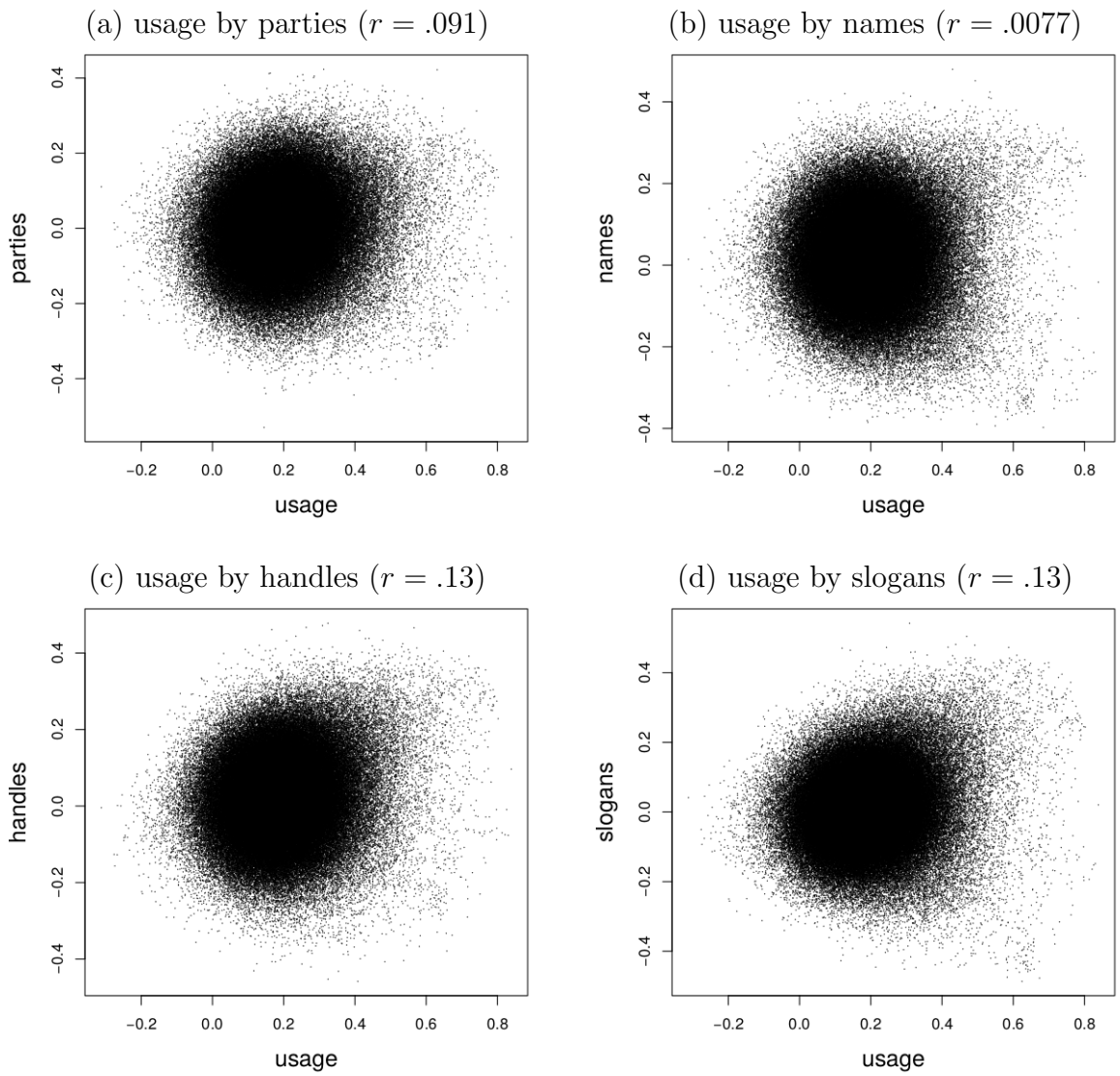


Figure 2.9: Scatterplots of partisan association scores and the usage score. r is the product-moment correlation.

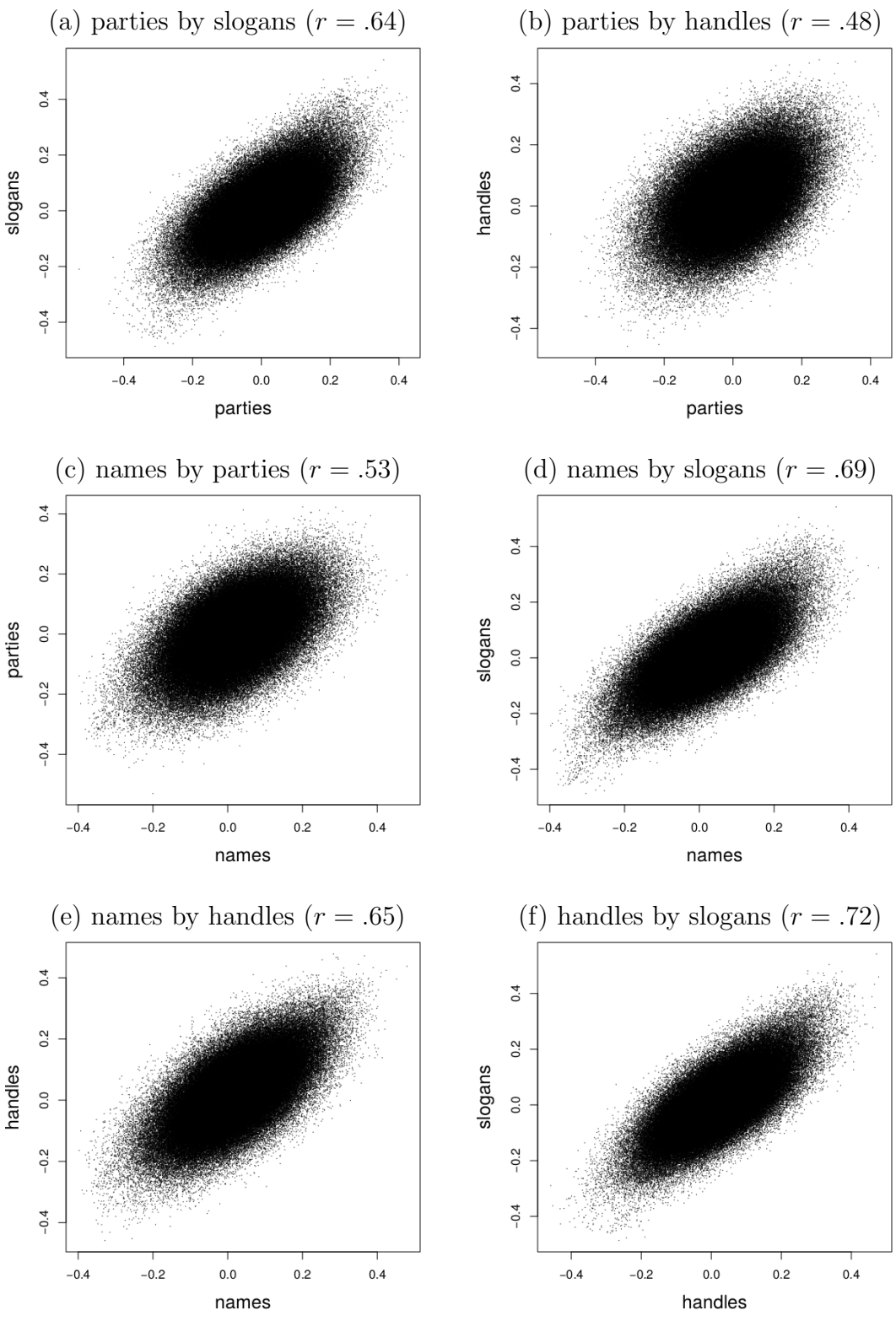


Figure 2.10: Scatterplots of comparisons between the partisan association scores. r is the product-moment correlation.

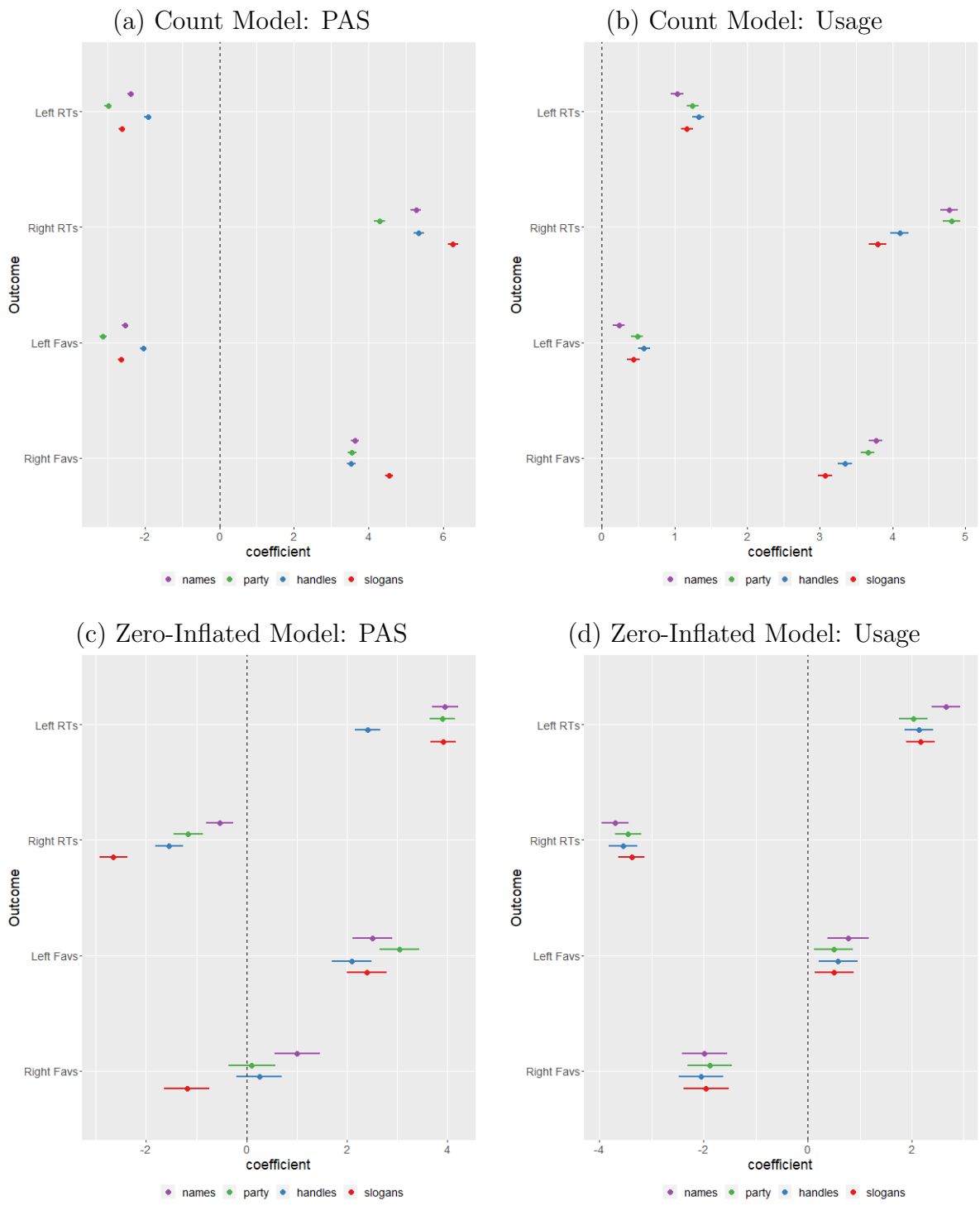


Figure 2.11: Coefficients from the zero-inflated negative binomial regressions of retweets and favorites of left and right accounts on partisan association scores (PAS) and the usage score.

2.9.7 Application 1: 2018 Pew Survey Data

2.9.7.1 Constructing the Variables from the Pew Survey Responses

The 2018 Pew survey data was collected shortly after the 2018 November midterm election. The data are a subset of "users who agreed to provide their Twitter handles" from "respondents from Ispos' KnowledgePanel, a probability-based panel of U.S. adults" conducted between November 21, 2018 and December 17, 2018 (Wojcik & Hughes, 2019). We supplied Pew researchers with a Python program that uses the embeddings models trained using the 2016 bios from incident-tweeting users to predict vectors from strings. The Pew Researchers then applied the program to the bios of their Twitter users and gave us the resulting cosine similarites matched to the users' responses for party identification and ideology survey questions. We have cosine similarities for $n = 1,276$ survey respondents.

The survey used two questions to measure party identification. The first question asked was, "In politics today, do you consider yourself a: Republican, Democrat, Independent, or Something Else?" Those who did not respond "Republican" or "Democrat" are then asked, "As of today, do you lean more to the Republican Party or the Democratic Party?" To measure the ideology, the survey uses an eleven-point scale. All respondents are asked, "Please click where you would place YOURSELF on the scale below." In the data, 0 corresponds to "very conservative" and 10 corresponds to "very liberal." Table 2.11, created by Walter Mebane, shows the joint distribution of the party identification and ideology survey responses among the users for whom we have partisan association scores.

	ideology											
party identification	0	1	2	3	4	5	6	7	8	9	10	all
Democrat	5	2	2	8	18	65	52	114	105	87	83	545
Dem. Lean	2	2	5	3	13	77	53	40	30	16	19	261
Independent	1	0	0	0	2	7	2	0	1	0	2	26
Rep. Lean	11	11	15	14	37	46	8	1	0	0	2	146
Republican	43	44	48	53	43	48	8	5	2	2	2	298
all	62	59	70	78	113	243	123	160	138	105	108	

Table 2.11: Pew Research Center survey data partisanship and ideology distribution.

2.9.7.2 Factor Analysis with Pew Survey Data

Wu et al. (2019) conduct a factor analysis using the party identification and ideology survey variables.[19] They treat the five-level party identification and eleven-level ideology variables

[19]Walter Mebane conducted the analysis in this section and wrote most of this section.

as ordinal and the partisan association scores as continuous (Quinn, 2004) and first use a one-factor model. Table 2.12 contains the results of the 1-factor model. Model A shows that a single factor reasonably accomodates party identification and partisan association scores.[20] As the partisan association scores' loadings are positive, this suggests that partisan association scores relate to partisanship in a manner similar to the way party identification does. The estimated loadings are also similar to the loading of 1.0 fixed for party identification.[21] Adding the ideology survey question to the one-factor model also does not change much (Table 2.12, Model B). The factor scores have a posterior mean variance similar to that for Model A, and the loadings for the partisan association scores are similar. The loading of the ideology survey variable is smaller in magnitude than are the loadings for any of the partisan association scores.

	95% Credible Intervals			
	Model A		Model B	
manifest variable	factor 1		factor 1	
party identification[a]	1.0[b]		1.0[b]	
ideology[a]	—		−.525	−.374
`parties`	.763	.886	.756	.877
`names`	.843	.960	.828	.946
`handles`	.845	.963	.822	.937
`slogans`	.980	1.090	.953	1.070

Table 2.12: Party, ideology, and partisan association scores: one-factor models' loadings

Note: nonordinal manifest variables are studentized. [a] ordinal variable. [b] fixed loading. Threshold 95% credible intervals, without ideology: $\gamma_{P2} = [.521, .647]$, $\gamma_{P3} = [.583, .712]$, $\gamma_{P4} = [.950, 1.11]$; with ideology: $\gamma_{P2} = [.529, .664]$, $\gamma_{P3} = [.591, .730]$, $\gamma_{P4} = [.973, 1.14]$, $\gamma_{I2} = [.284, .464]$, $\gamma_{I3} = [.565, .776]$, $\gamma_{I4} = [.809, 1.04]$, $\gamma_{I5} = [1.11, 1.35]$, $\gamma_{I6} = [1.65, 1.90]$, $\gamma_{I7} = [1.90, 2.17]$, $\gamma_{I8} = [2.27, 2.55]$, $\gamma_{I9} = [2.66, 2.96]$, $\gamma_{I10} = [3.08, 3.39]$. Latent variable variances: model A, $\sigma^2 = .705$; model B, $\sigma^2 = .726$.

But the underlying structure of the data is arguably more complicated. Table 2.13 shows that adding a latent dimension that is keyed to ideology but excludes party identification reveals that only the party and slogans partisan association scores relate to party identification via loadings that have unambiguous signs, but all the partisan association scores relate unambiguously to the latent variable that is keyed to ideology. The posterior of the party

[20] Estimates come from `MCMCmixfactanal` in Martin et al. (2018).

[21] The loadings can be compared because the continuous scale underlying the party identification survey variable has a standard normal prior and the partisan asociation differences are studentized (Quinn, 2004, p. 341).

latent variable does increase compared to the one-factor model, and the posterior variance of the ideology latent variable is similar. The loading for the ideology survey variable on the party latent variable is an order of magnitude larger than in the one-factor model, so ideology remains related to partisanship. The cutpoints of the ideology survey variable are also much larger than they are in the one-factor specification. These do make sense, though: the ideology survey variable separates strongly to either side of "Independent," but it also exhibits variation between "Democrat" and "Lean Democratic" and between "Republican" and "Lean Republican."

	95% Credible Intervals			
manifest variable	factor 1		factor 2	
party identification[a]	1.0[b]		0.0[b]	
ideology[a]	−6.45	−3.03	1.0[b]	
`parties`	.091	.230	−.730	−.624
`names`	−.018	.132	−.829	−.723
`handles`	−.064	.090	−.839	−.735
`slogans`	.001	.157	−.931	−.831

Table 2.13: Party, ideology, and partisan association scores: two-factor model's loadings

Note: nonordinal manifest variables are studentized. [a] ordinal variable. [b] fixed loading. Threshold 95% credible intervals: $\gamma_{P2} = [.687, .857]$, $\gamma_{P3} = [.775, .954]$, $\gamma_{P4} = [1.28, 1.49]$; $\gamma_{I2} = [.903, 2.24]$, $\gamma_{I3} = [1.77, 3.87]$, $\gamma_{I4} = [2.60, 5.46]$, $\gamma_{I5} = [3.63, 7.41]$, $\gamma_{I6} = [5.48, 10.9]$, $\gamma_{I7} = [6.36, 12.6]$, $\gamma_{I8} = [7.55, 15.0]$, $\gamma_{I9} = [8.72, 17.3]$, $\gamma_{I10} = [9.94, 19.7]$. Latent variable variances: $\sigma_1^2 = 1.03$, $\sigma_2^2 = .971$, $\sigma_{12} = -.000855$.

The differences between the one-factor and two-factor specifications are more about the structure of the latent space than about accounting for the manifest variables' variability. The residual variances of the partisan association differences are similar for the two-factor specification and the one-facter specification: the two-factor specification does slightly worse for the party partisan association scores but better for the other partisan association scores.[22] The party and ideology latent variables are nearly uncorrelated, but adding the second latent common factor does not get at something that was largely missed in the one-factor specification. Rather the issue is largely how to represent the latent common space. In the specification of Table 2.13 the second dimension is an aspect of ideology that is largely orthogonal to party.

[22] 95% credible intervals for the residual variances in the one-factor model (Model B) are $\Psi_{tc} = [.388, .475]$, $\Psi_{rd} = [.473, .567]$, $\Psi_{rh} = [.399, .486]$, $\Psi_{ms} = [.227, .301]$, while for the two-factor specification they are $\Psi_{tc} = [.369, .455]$, $\Psi_{rd} = [.485, .577]$, $\Psi_{rh} = [.358, .442]$, $\Psi_{ms} = [.200, .278]$.

The clearest view of what is going on in the data comes from considering the partisan association scores separately instead of as differences. That is, instead of calculating the cosine similarities with the partisan subspaces, we calculate the cosine similarities with each term separately. Table 2.14 shows estimates for a three-factor specification with the ten partisan association scores (studentized) as manifest variables. The three factors include separate party and ideology latent variables along with a usage latent variable. For usage the loading for `democrat` is set equal to 1.0 to set the scale of the latent variable. The usage latent variable has the largest posterior variance ($\sigma_2^2 = .871$), party has the second largest ($\sigma_1^2 =, 673$) and ideology the smallest ($\sigma_3^2 = .629$). The latent variables are largely uncorrelated.[23]

manifest	95% Credible Intervals					
variable	factor 1		factor 2		factor 3	
party identification[a]	1.0[b]		0.0[b]		0.0[b]	
ideology[a]	−.463	−.238	0.0[b]		1.0[b]	
`republican`	.292	.564	.820	.921	−.584	−.302
`democrat`	−.006	.276	1.0[b]		−.228	.084
`trump`	−.271	.031	.408	.642	−1.160	−.969
`donald`	−.278	.006	.366	.587	−1.060	−.875
`clinton`	−.635	−.382	.663	.860	−.668	−.385
`hillary`	−.694	−.425	.714	.915	−.649	−.349
`realdonaldtrump`	−.220	.047	.378	.582	−.984	−.792
`hillaryclinton`	−.564	−.327	.655	.798	−.288	−.019
`MAGA`	−.131	.157	.407	.615	−1.090	−.903
`StrongerTogether`	−.657	−.427	.606	.779	−.387	−.116

Table 2.14: Party, ideology, and partisan association scores: three-factor model's loadings

Note: nonordinal manifest variables are studentized. [a] ordinal variable. [b] fixed loading. $n = 1276$. Threshold 95% credible intervals: $\gamma_{P2} = [.504, .631]$, $\gamma_{P3} = [.566, .702]$, $\gamma_{P4} = [.929, 1.10]$; $\gamma_{I2} = [.313, .505]$, $\gamma_{I3} = [.607, .835]$, $\gamma_{I4} = [.872, 1.12]$, $\gamma_{I5} = [1.20, 1.45]$, $\gamma_{I6} = [1.78, 2.04]$, $\gamma_{I7} = [2.06, 2.33]$, $\gamma_{I8} = [2.44, 2.73]$, $\gamma_{I9} = [2.84, 3.15]$, $\gamma_{I10} = [3.28, 3.63]$. Latent variable variances: $\sigma_1^2 =, 673$, $\sigma_2^2 = .871$, $\sigma_3^2 = .629$, $\sigma_{12} = -.000955$, $\sigma_{13} = .00100$, $\sigma_{23} = -.000152$.

Now it's clear that some of the associations relate unambiguously to partisanship, as measured on the survey by party identification, while others do not. Related to the party dimension in intuitive ways are `republican`, `clinton`, `hillary`, `hillaryclinton` and `strongertogether` but not `democrat`, `trump`, `donald`, `realdonaldtrump` or `MAGA`.

[23]Correlations between factor score posterior means are$r_{12} = -.00125$, $r_{13} = .00154$, $r_{23} = -.000206$.

republican tends to increase as the party latent variable increases, while clinton, hillary, hillaryclinton and strongertogether tend to decrease as the party latent variable increases. All the partisan associations are positively related to the usage latent variable, although trump, donald, realdonaldtrump and MAGA have smaller loadings than do the other partisan associations. All the scores except democrat relate unambiguously to ideology: more "liberal" values of the latent variable go with lower values on all the partisan associations except democrat. The loadings for republican, trump, donald, realdonaldtrump and maga on the ideology latent variable are respectively more negative than are the loadings for democrat, clinton, hillary, hillaryclinton and strongertogether.

The pattern of loadings in Table 2.14 of the partisan associations for the ideology latent variable makes it clear that the negative loadings of the partisan association differences for the ideology latent variable in Table 2.13 in the main paper do not mean that republican, trump, donald, realdonaldtrump and maga tend to decrease as the latent variable increases—becomes more "liberal"—while democrat, clinton, hillary, hillaryclinton and strongertogether tend to increase. Instead, democrat changes indeterminedly and clinton, hillary, hillaryclinton and strongertogether each tends to decrease as the ideology latent variable increases, it's just that republican, trump, donald, realdonaldtrump and maga respectively decrease more.

Likewise, it's not that republican tends to increase and democrat tends to decrease as the party latent variable increases. Only republican has an unambiguous relation with that latent variable, but that relation tends to have larger magnitude than does the ambiguous relation of democrat. So republican−democrat acquires a positive loading for the party latent variable in the two-factor model of Table 2.13.

Table 2.13 does not show that the score differences are all good measures of ideology—orthogonal to party—nor that they are poor measures of partisanship. Of course ideology-orthogonal-to-party is not the notion that most have in mind when they refer to ideology. Plus most Americans do not think in ideological terms in a consistent way that coherently spans many policies (Kinder & Kalmoe, 2017). It is intriguing that in the wake of the 2018 election only republican and the Hillary-focused partisan associations unambiguously relate to the party latent variable. It is natural to think this pattern reflects divisions in the Republican party regarding Trump, but given our data such can only be a speculation. Perhaps reactions to Trump also explain why democrat is not unambiguously related to either party or ideology latent variables: perhaps the resistance prompted many to lean Democratic only tactically; again given our data we can only speculate about this.

The best thing to say is that the partisan association differences are excellent measures of partisanship with an admixture of whatever "liberal" versus "conservative" means to survey

respondents. Also it's fair to say that—two years after the presidential election Hillary Clinton ran in—the `clinton`, `hillary`, `hillaryclinton` and `StrongerTogether` partisan associations are more about party than they are about "ideology," but they are not not about "ideology."

2.9.8 Application 1: Coefficients of the Zero-Inflated Negative Binomial Model Across the Other Three Partisan Association Scores Across Users with an Estimated Ideal Point

The analysis in Section 2.4.8 in Figure 2.5 only uses the "names" partisan association scores. We replicate the same analysis for the "parties" partisan association scores (Figure 2.12), "handles" partisan association scores (Figure 2.13), and "slogans" partisan association scores (Figure 2.14). The signs of the coefficients of the models using the other partisan association scores are in line with expectations.

2.9.9 Application 2: COVID-CORE Dataset Keywords

Lu & Mei (2020) used the following keywords to obtain tweets about the coronavirus and COVID-19 pandemic between January 1, 2020 and July 1, 2020: "covid 19", "covid 2019", "covid", "19 ncov", "2019 ncov", "ncov", "corona virus", "pandemic", "wuhan virus", "chinese virus", "china virus", "sars cov 2", "hcov 19", "ncov 19". They also considered common variations, such as "covid-19", using regular expressions. Case was ignored as well.

2.9.10 Design of the Support Vector Machine Classifier for Negative Usages of Keywords

We used a training set of $1,000$ bios that used at least one partisan keyword. Each usage of a keyword was labeled as either negative or not-negative usage. If a bio contained more than one keyword, we separated the bio into multiple observations. For example, if the bio states, "I support Donald Trump," we separate this bio into two observations: one for "Donald" and one for "Trump." We create a "tall" dataset in this fashion to accommodate instances where a person may simultaneously use one keyword in a not-negative fashion while using another in a negative manner. Creating such a "tall" dataset we get $1,271$ observations in the training set, each labeled as a not-negative usage of a keyword or a negative usage of the keyword.

The main innovation of the approach is in the creation of features beyond text features. These features are as follows.

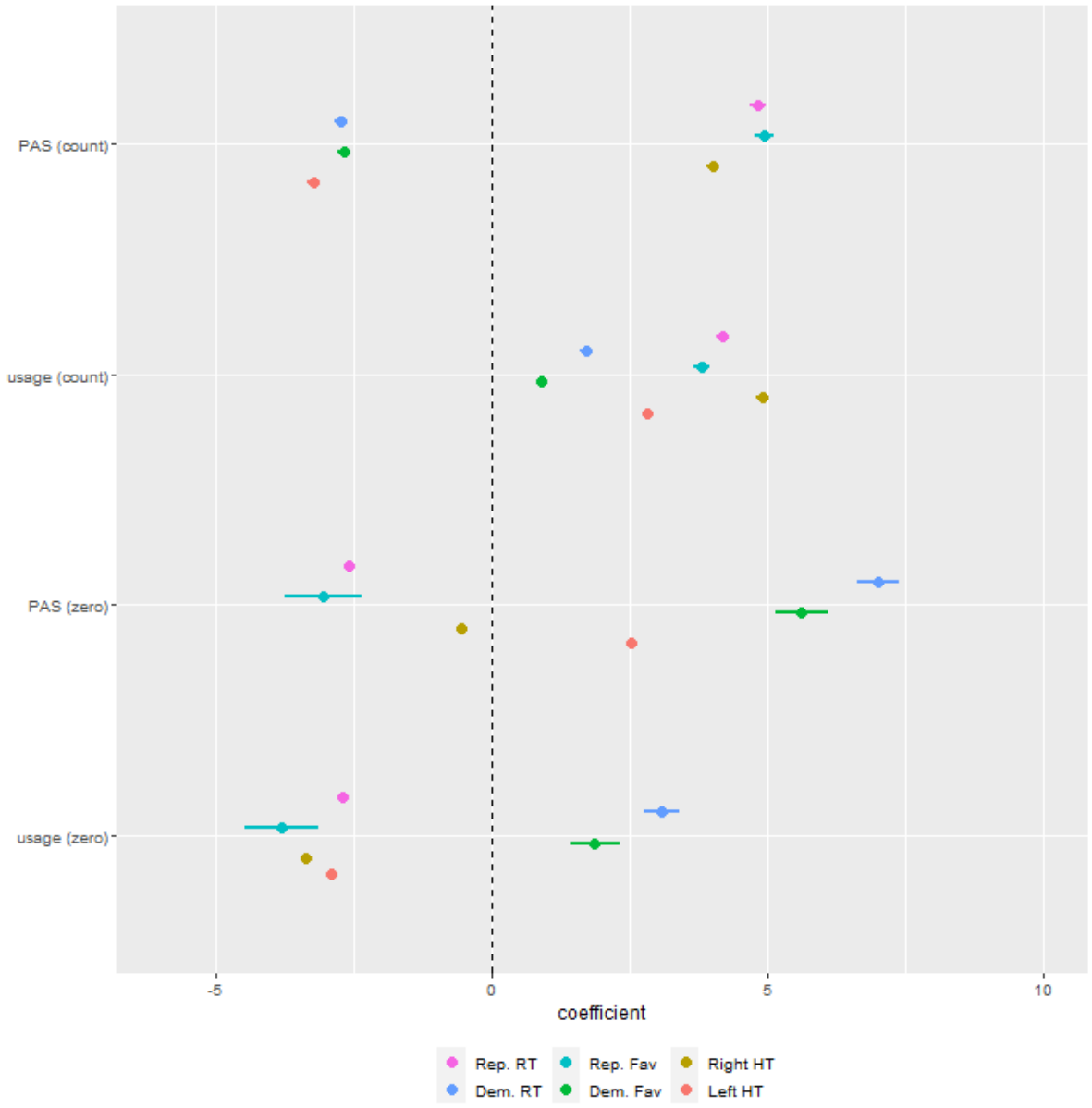


Figure 2.12: Coefficients from the zero-inflated negative binomial regressions of retweets, favorites, and hashtags on the "parties" partisan association scores and the usage scores for users with estimated ideal points.

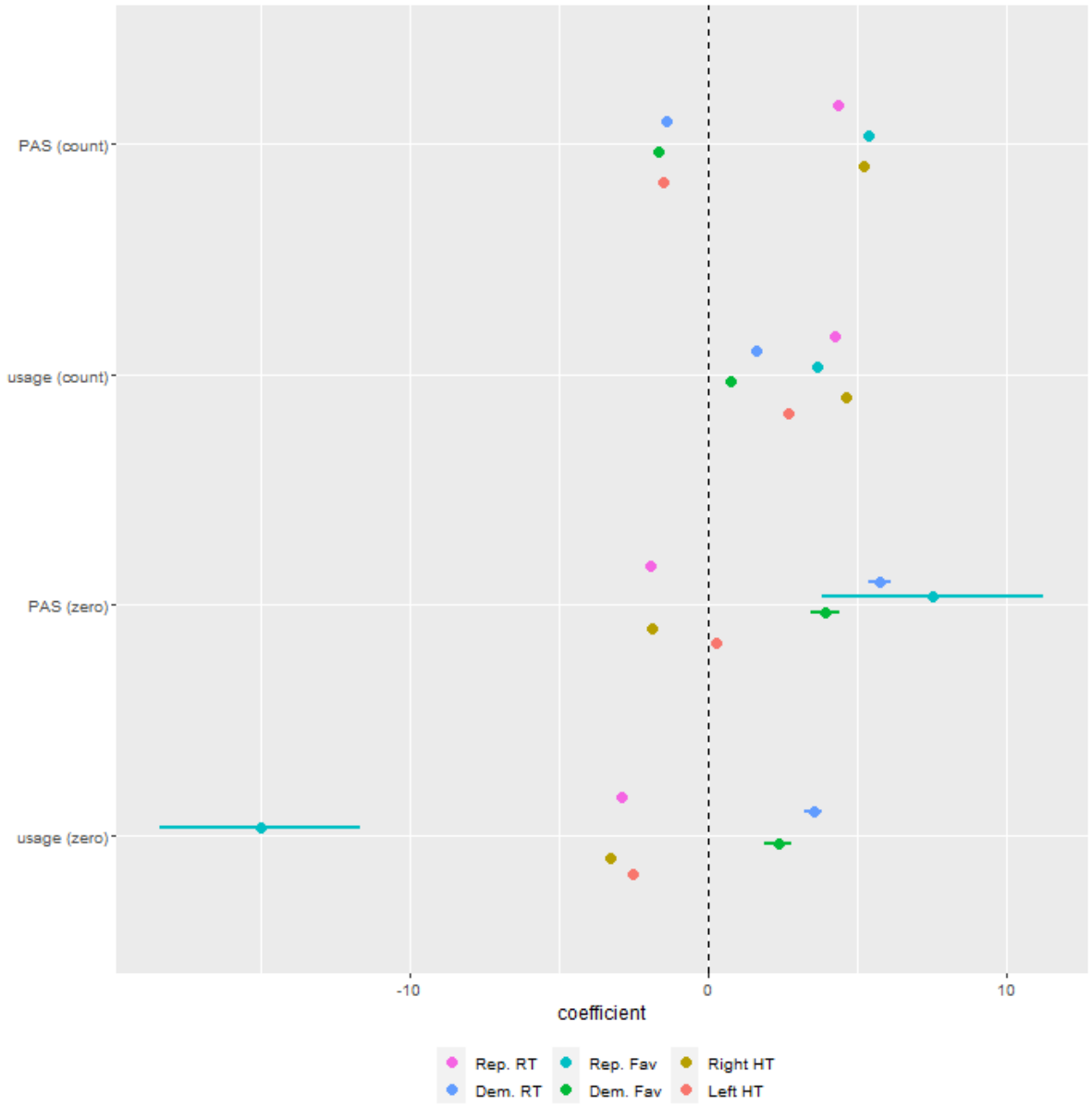


Figure 2.13: Coefficients from the zero-inflated negative binomial regressions of retweets, favorites, and hashtags on the "handles" partisan association scores and the usage scores for users with estimated ideal points.

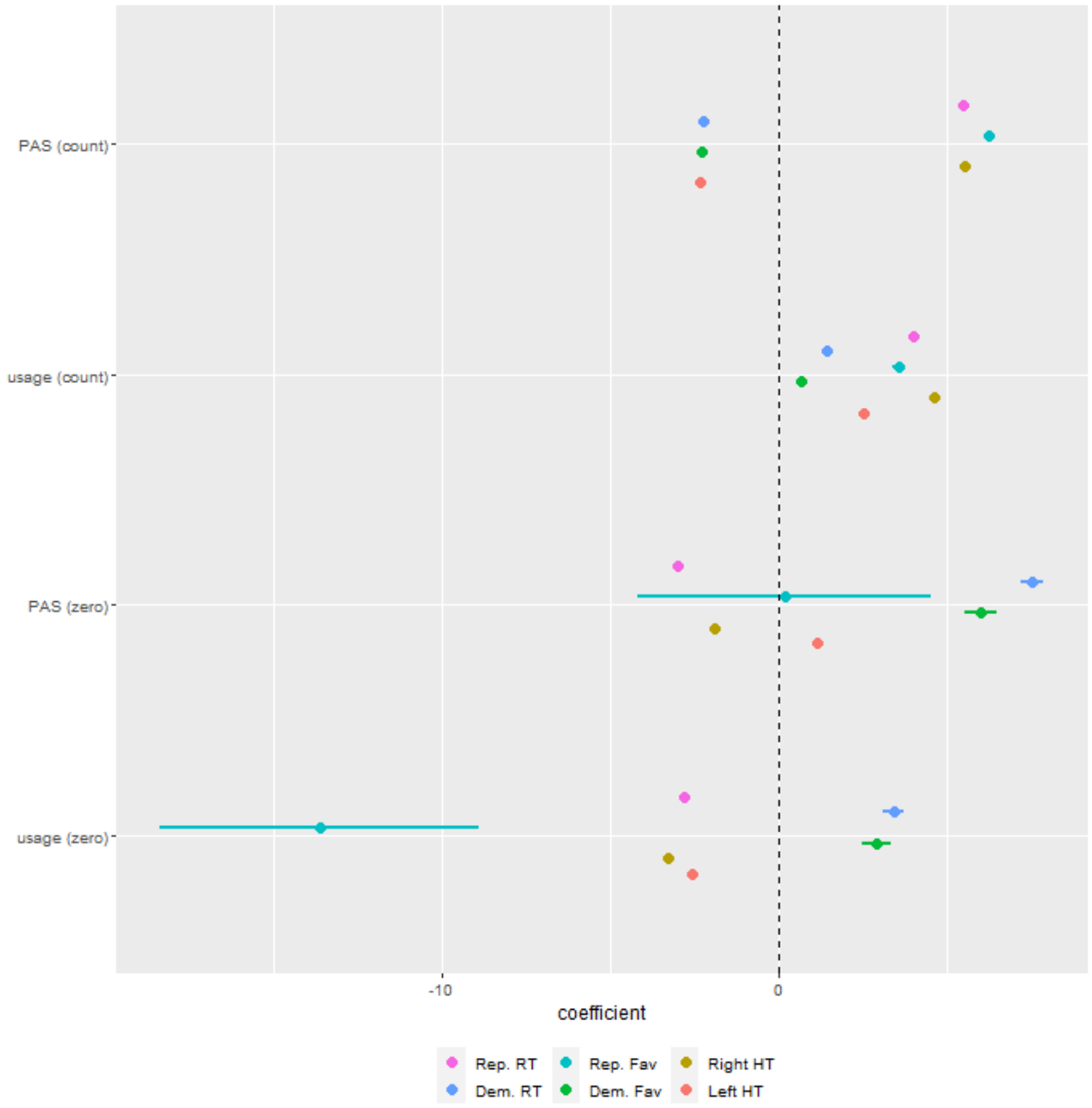


Figure 2.14: Coefficients from the zero-inflated negative binomial regressions of retweets, favorites, and hashtags on the "slogans" partisan association scores and the usage scores for users with estimated ideal points.

1. `both`: whether the user used both a Democratic and Republican keyword in their bio
2. `Republican Word Count`: how many Republican keywords the user used
3. `Democratic Word Count`: how many Democratic keywords the user used
4. `Indicator Variables for the Keywords`: indicator variables for each of the 10 keywords which is 1 for the keyword currently being classified as negative or not, and 0 for the other keywords
5. `Indicator Variables for All Keywords`: indicator variables for each of the 10 keywords which is 1 for any keyword used in the profile and 0 for keywords not used
6. `Partisan Association Score`: the averaged partisan association score generated, using the average of the five Democratic keywords subtracted from the average of the five Republican keywords, when not taking into account negative or not-negative usages of the keywords
7. `Partisan Association Score, Partisan Keywords Removed`: the overall partisan association score generated, using the average of the five Democratic keywords subtracted from the average of the five Republican keywords, when taking the cosine similarity between the average partisan subspace and the inferred document vector of the bio when all partisan keywords are removed from the bio

The underlying intuition behind these features, as suggested by Gong et al. (2017), are that the features measure how "out of place" a word is. In other words, if the partisan association of the user dramatically shifts with the removal of a partisan word, then there is an incongruity between the bio and that partisan word, suggesting that the partisan word may not be used in an positive (or at least neutral) manner. We use a Gaussian kernel support vector machine classifier; hyperparameters were chosen using tenfold cross-validation. We implemented the classifier using `scikit-learn` (Pedregosa et al., 2011).

CHAPTER 3

MARMOT: A Deep Learning Framework for Constructing Multimodal Representations for Vision-and-Language Tasks

Abstract

Political activity on social media presents a data-rich window into political behavior, but the vast amount of data means that almost all content analyses of social media require a data labeling step. However, most automated machine classification methods ignore the multimodality of posted content, focusing either on text or images. State-of-the-art vision-and-language models are unusable for most political science research: they require all observations to have both image and text and require computationally expensive pretraining. This paper proposes a novel vision-and-language framework called **m**ultimod**a**l **r**epresentations using **mo**dality **t**ranslation (MARMOT). MARMOT presents two methodological contributions: it can construct representations for observations missing image or text, and it replaces the computationally expensive pretraining with modality translation. MARMOT outperforms an ensemble text-only classifier in 19 of 20 categories in multilabel classifications of tweets reporting election incidents during the 2016 U.S. general election. Moreover, MARMOT shows significant improvements over the results of benchmark multimodal models on the Hateful Memes dataset, improving the best result set by VisualBERT in terms of accuracy from 0.6473 to 0.6760 and area under the receiver operating characteristic curve (AUC) from 0.7141 to 0.7530. The GitHub repository for MARMOT can be found at github.com/patrickywu/MARMOT.

Author's Note

This paper is co-authored with Walter R. Mebane, Jr. I am the first author on this paper. This work was supported in part by an NSF RIDIR grant under award number SES-1925693

("The Sub-National Data Archive System for Social and Behavioral Data") and a fellowship from the Michigan Institute for Computational Discovery & Engineering (MICDE).

3.1 Introduction

This paper introduces a novel deep learning framework for constructing representations for vision-and-language tasks usable for political science and communications research of social media. This framework seeks to improve the classification or labeling step of research on social media content, which usually ignores either the image or the post's text. For example, Barberá et al. (2019) use latent Dirichlet allocation (Blei et al., 2003) to cluster text of tweets by topic. Mebane et al. (2018) use active learning with support vector machines and an ensemble classifier to sort the text of tweets into categories and subcategories. Pan & Siegel (2020) use a crowdsourcing approach to label the text of tweets into specific categories and sentiment. Casas & Webb Williams (2019) also use a crowdsourcing approach to label images from tweets into the emotions they were supposed to invoke. All such examples label, classify, or cluster posts using one modality.

A unimodal focus can potentially create biases in the processed data, leading to misleading inferences in the downstream analysis of the data. Both image and text must be considered to reduce these potential biases. For example, consider the following tweet in Figure 3.1. If we were interested in classifying tweets as reports of being able to vote with no reported problems, this tweet would fit those criteria: the text indicates that the person finished something, along with the vote hashtag, and the image of the "I Voted" sticker indicates that the person voted. Thus, the combination of the text and the image indicates that the person could successfully vote, meeting the labeling criteria. However, if another tweet contained a similar image but the text indicated that they encountered problems voting, such a tweet would not fit the labeling criteria. Only by jointly considering the image and text can we definitively conclude that the person in Figure 3.1 was able to vote with no reported problems.

An approach to machine classification of multimodal posts would be to jointly map image-text combinations to d-dimensional vectors, an approach known as representation learning. Representation learning methods automatically learn the mapping of observations to vector representations. Popular examples of representation learning methods include word2vec (Mikolov et al., 2013), GloVe (Pennington et al., 2014), and doc2vec (Le & Mikolov, 2014).

Representation learning methods exist that produce joint representations of images and text. Some of these models, known as late fusion approaches, use separate models for image and text and then combine the outputs of these two models (Liu et al., 2018). These models usually allow for missing modalities among observations but fail to effectively learn patterns between images and text. Other models, known as early fusion approaches, input both image and text features into a single model. Early fusion models efficiently learn patterns between

Figure 3.1: "Done and done #vote"

the image and text directly (Liu et al., 2018). Most state-of-the-art approaches use an early fusion approach.

State-of-the-art approaches such as ViLBERT (Lu et al., 2019) and VisualBERT (Li et al., 2019) are not well-suited for political science and communications research. Early fusion approaches usually require all observations have image and text, which is almost always not true for social media data that social scientists are interested in—posts may contain text, image(s), or some combination of both. Most state-of-the-art early fusion models, most of which are based on transformers (Vaswani et al., 2017), are also pretrained on image annotation datasets, such as Microsoft COCO (Lin et al., 2014). These image annotation datasets contain images with associated captions. This pretraining serves two purposes. First, it adapts the underlying transformers-based pretrained language model, originally trained to accept text input only, to accept as input both text and image features. Second, it learns the relationship between the text and image. However, because most real-world multimodal observations that need to be classified are not simply a caption describing what is happening in an image, additional pretraining is needed using the data of interest in order for the model to adapt to the target domain. Pretraining is also computationally expensive, requiring computational resources not available to most social scientists.

To this end, we propose **m**ultimod**a**l **r**epresentations from **mo**dality **t**ranslation, or MARMOT, a novel transformers-based architecture that produces multimodal representations. MARMOT aims to solve the two issues that make state-of-the-art multimodal models unusable for political science and communications research. First, we use attention masks to

handle missing modalities explicitly. Attention masks, typically used in machine translation and question-answering tasks, are used to prevent the self-attention mechanism of the transformer from attending to missing modalities, meaning that representations can be constructed even for observations missing modalities. Second, to capture the spirit of pretraining while avoiding expensive computational costs, we propose modality translation. Instead of pretraining our model on an image annotation dataset, we directly generate captions for each observation containing an image using a pretrained image captioner. To avoid having to adapt the underlying transformers-based pretrained language model to accept both text and image features, we use a transformer decoder initialized with pretrained BERT weights (Devlin et al., 2018). The image captions, derived from a pretrained image captioner such as self-critical sequence training (Rennie et al., 2016), are inputted into the BERT decoder, and the image features, derived from a pretrained image network such as ResNet-152 (He et al., 2015), are inputted at the encoder-decoder attention layer. The BERT decoder constructs what we call the translated image. Modality translation maps image features to the relevant parts of the text feature space. The last step jointly inputs the text, image captions, and translated image features into a transformer encoder initialized with pretrained BERT weights. The output of the transformer encoder is the joint image-text representation.

MARMOT makes two methodological contributions. First, it introduces modality translation. Modality translation replaces the computationally expensive pretraining process and allows the model to learn directly from the data of interest. Second, the model can calculate representations even for observations missing an image or text.

We apply MARMOT to data classification tasks on two datasets. The first is a dataset of tweets reporting election incidents during the 2016 U.S. general election (Mebane et al., 2018). All tweets contain text but some tweets contain an image. MARMOT outperforms the text-only classifier used in Mebane et al. (2018) on 19 of 20 categories in multilabel classifications of tweets (and equals the performance in the last category). The second is the Hateful Memes dataset, recently released by Facebook Research to test multimodal models (Kiela et al., 2020). The goal is to classify each meme as hateful or not. Detecting hateful speech and memes is of interest to both computer science (e.g., Davidson et al., 2017; MacAvaney et al., 2019) and political science (e.g., Siegel et al., 2019; Siegel & Badaan, 2020). This dataset also contains text and an image for each observation, making MARMOT comparable to other state-of-the-art multimodal models. MARMOT improves upon the results set by benchmark state-of-the-art multimodal models on this dataset, even outperforming pretrained multimodal models. It improves the best benchmark result set by VisualBERT (Li et al., 2019) pretrained on MS COCO in terms of accuracy from 0.6473 to 0.6760 and in terms of area under the receiver operating characteristic curve (AUC) from 0.7141 to 0.7530.

The model finished in the top 1% of all participants in the Hateful Memes challenge.

The paper proceeds as follows. First, we review the literature on multimodal models. We then detail the architecture of MARMOT. We apply MARMOT to a dataset of election incidents reported on Twitter during the 2016 U.S. general election and the Hateful Memes dataset. We then make concluding remarks and discuss future directions of the project.

3.2 Approaches to Classifying Multimodal Data

A *modality* is defined as some item that provides information to a reader or viewer, such as text, audio, images, or video. It is also common for multiple modalities to exist together, providing readers with the task of jointly considering the modalities to understand the information conveyed (Mogadala, 2015). Our paper considers image and text, but other works have focused on other combinations of modalities as well, such as text and video (see, e.g., Sun et al., 2019).

The goal is to develop a model usable for social scientists labeling data that consist of both images and text. That means that the model must work with small datasets, it must be able to run with modest computational resources, and it must handle data that may have observations missing modalities. We look at the previous models and works that we leverage within MARMOT to develop a model that satisfies these requirements.

3.2.1 Late Fusion vs. Early Fusion Models

Early works on multimodal learning largely focused on late fusion approaches, a set of multimodal learning models where individual modalities are inputted into separate models. The outputs of these separate models are then combined through a policy. Initial works used major features of the images and bag-of-words approaches (Tian et al., 2013). Later works used deep learning methods such as deep convolutional neural networks (Zahavy et al., 2016). Social science methodologists have also developed late fusion approaches. Zhang & Pan (2019) use a late fusion approach with a convolutional neural network for images and a recurrent neural network for text in order to identify Weibo posts that discuss offline collective action. The main advantage of late fusion approaches is that observations can be missing modalities: one could use the representation generated by the text model alone, the representation from the image model alone, or the combined representation from the two models. Nevertheless, because there are separate models for each modality, such an approach cannot learn interactions or patterns across the modalities in a meaningful fashion.

Early fusion approaches, on the other hand, create joint representations of images and

text. A single model is used to learn within and across both modalities, which is its key advantage. It assumes, however, that one model is suitable for both modalities. Wang et al. (2019) note that early fusion multimodal networks often perform poorly because using a single optimization strategy is almost always suboptimal for a model that deals with multiple modalities.

Notwithstanding these issues, early fusion approaches have quickly risen in popularity with the development of the transformer and pretrained language models such as BERT. Some of these self-supervised architectures include VisualBERT (Li et al., 2019), Visual-Linguistic BERT (VL-BERT, Su et al., 2019), Vision and Language BERT (ViLBERT, Lu et al., 2019), Learning Cross-Modality Encoder Representations from Transformers (LXMERT, Tan & Bansal, 2019), and Multimodal Bitransformers (MMBT, Kiela et al., 2019).

3.2.2 Attention and Transformers

The transformer consists of four components: the attention mechanism, layer normalization, residual connections, and the feedforward layer. To support understanding the transformers that MARMOT is based on and the attention mask feature of MARMOT, we review the attention mechanism and transformers here. Brief introductions to feedforward neural networks, residual connections, and layer normalization can be found in Sections 3.8.1, 3.8.2, and 3.8.3, respectively, in the Supplemental Information. More technical introductions about attention and transformers can be found at Rush (2018) and Bloem (2019).

3.2.2.1 Attention

Transformers use attention (Bahdanau et al., 2014) in the self-attention layer and the encoder-decoder attention layer. Rather than directly defining attention, we first turn to the more intuitive self-attention. Attention is a generalization of self-attention.

Self-attention is a sequence-to-sequence operation, meaning that a sequence of vectors is inputted and a sequence of vectors is outputted. Self-attention relates all positions of a sequence with one another in order to compute a new representation of the same sequence. To make this idea more concrete, we start with a simplified version of self-attention. Denote the input vectors as $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N$ and denote the output vectors, the new representation of the sequence of vectors $\mathbf{x}$, as $\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_N$. We can assume that all vectors $\mathbf{x}$ and $\mathbf{z}$ have dimension k. To calculate $\mathbf{z}_i$, simplified self-attention simply takes a weighted sum over all

input vectors $\mathbf{x}_j$, for $j \in \{1, ..., N\}$:

$$\mathbf{z}_i = \sum_j w_{ij} \mathbf{x}_j \tag{3.1}$$

The weight w_{ij} is not a learned parameter, but is calculated from a similarity function over $\mathbf{x}_i$ and $\mathbf{x}_j$, such as the dot product: $w'_{ij} = \mathbf{x}_i^T \mathbf{x}_j$. Because the dot product calculates a weight that is between negative and positive infinity, we apply a softmax function to map the weights w'_{ij} to $[0, 1]$ and to ensure they sum to 1 over the entire sequence:

$$w_{ij} = \frac{\exp(w'_{ij})}{\sum_l \exp(w'_{il})} \tag{3.2}$$

w_{ij} are known as the attention weights because they indicate how much attention the ith output vector $\mathbf{z}_i$ should pay to the jth input vector $\mathbf{x}_j$. The core goal of self-attention is propagating information between input vectors.

There are a few more modifications to give self-attention more representational power. Notice that each input vector $\mathbf{x}_i$ is used in three ways: (1) it is compared to every other input vector $\mathbf{x}_j$, $j \in \{1, ..., N\}$, to calculate the weights for output $\mathbf{z}_i$; (2) it is compared to every other input vector to calculate the weights for all other outputs $\mathbf{z}_j$, $j \in \{1, ..., i-1, i+1, ..., N\}$; (3) and it is used as a part of the weighted sum in Equation 3.1 to compute each output vector. These roles are called the query, the key, and the value, respectively. To allow the query, key, and value to differ, we can define them as $\mathbf{q}_i = \mathbf{W}_q \mathbf{x}_i$, $\mathbf{k}_i = \mathbf{W}_k \mathbf{x}_i$ and $\mathbf{v}_i = \mathbf{W}_v \mathbf{x}_i$, respectively, where $\mathbf{W}_q$, $\mathbf{W}_k$, and $\mathbf{W}_v$ are $k \times k$ weight matrices. To give even more representational power, we can use h query, key, and value weight matrices, yielding h separate sets of output vectors that can be combined via a linear transformation. This is known as multiheaded attention.

The dot product is rarely used as the similarity function to calculate the weights because the softmax function is sensitive to large input values, affecting gradients in backpropagation. Because the average value of the dot product grows with k, we divide the dot product by $\sqrt{k}$, called the scaled dot product.

The following three equations reflect the above discussion and define self-attention:

$$w'_{ij} = \mathbf{q}_i^T \mathbf{k}_j \tag{3.3}$$

$$w_{ij} = \text{softmax}(w'_{ij}) \tag{3.4}$$

$$\mathbf{z}_i = \sum_j w_{ij} \mathbf{v}_j \tag{3.5}$$

Notice that self-attention is permutation invariant: nothing about Equations 3.3 to 3.5 takes into account the order of the vectors in the sequence $\mathbf{x}$. The attention weights derived for the sentence "Trump beat Clinton in the 2016 U.S. general election" would be the same as the attention weights derived for the sentence "Clinton beat Trump in the 2016 U.S. general election," even though the word order reveals a key distinction in the two sentences. We use positional embeddings to make the two sentences distinct. Positional embeddings map the word position to either learnable embeddings or fixed embeddings. These positional embeddings are then added to the input. Absolute position embeddings, where a unique embedding is learned for each position, are the most popular positional embedding choice. Fixed sinusoidal positional embeddings also work well in many contexts (Vaswani et al., 2017).

Generalized attention resembles self-attention in every manner except that the queries, keys, and values are not all based on the same sequence of vectors $\mathbf{x}$. The encoder-decoder attention layer defines the query as $\mathbf{q}_i = \mathbf{W}_q\mathbf{x}_i$, the key as $\mathbf{k}_i = \mathbf{W}_k\mathbf{y}_i$, and the value as $\mathbf{v}_i = \mathbf{W}_v\mathbf{y}_i$, where $\mathbf{x}_i$ comes from a sequence of vectors $\mathbf{x}$ and $\mathbf{y}_i$ comes from a separate sequence of vectors $\mathbf{y}$. Other than this, Equations 3.3, 3.4, and 3.5 are still used to calculate the outputted sequence(s) of vectors.

3.2.2.2 Transformers

Attention mechanisms were usually paired with recurrent neural networks (RNN), such as long short-term memory models (Hochreiter & Schmidhuber, 1997; Chang & Masterson, 2020; Xu et al., 2015). However, Vaswani et al. (2017) argued that the attention mechanism alone was enough to learn dependencies between words. The architecture they built around the attention mechanism is called the transformer. Because the transformer only uses attention, it entirely dispenses with recurrence and more effectively models long-term dependencies between words within the text. In an RNN, words that appear near the beginning of the document may be "forgotten" by the end of the document. In (self-)attention, every word is related to every other word, regardless of the distance between words.

The transformer consists of an encoder and decoder. See Figure 3.2 for an overview of the transformer architecture. Transformers were initially designed for machine translation tasks, where word embeddings $\mathbf{x}$ of one language were inputted into the transformer encoder, and the word embeddings α of the other language were inputted into the decoder. The output vectors of the transformer encoder $\mathbf{y}$ would be inputted into the encoder-decoder attention layer of the transformer decoder as the keys and values, while the queries came from the self-attention layer of the transformer decoder.

Many language models, such as BERT, exclusively use the transformer encoder. The

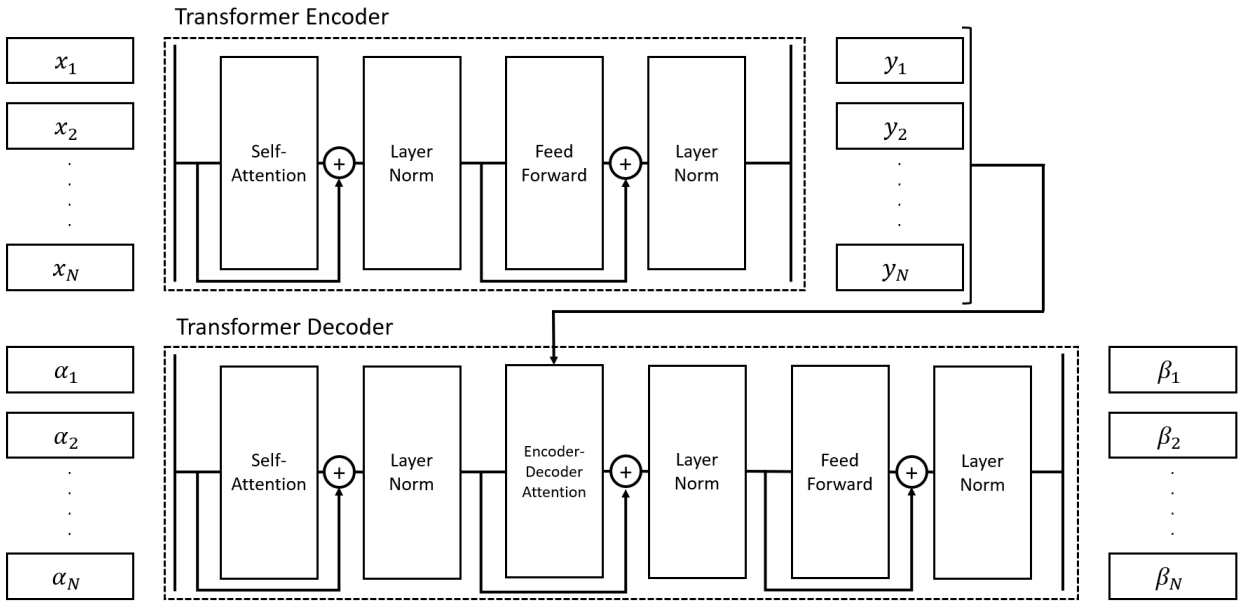


Figure 3.2: The transformer architecture (Vaswani et al., 2017). The outputs of the transformer encoder are inputted at the encoder-decoder attention layer of the transformer decoder as the keys and values, while the queries come from the self-attention layer in the transformer decoder. The plus sign with a circle around it indicates a residual connection.

transformer encoder is illustrated in the upper half of Figure 3.2. The transformer encoder consists of a self-attention layer, followed by layer normalization, a feedforward layer where the same feedforward neural network is applied separately to each input, and one last layer normalization; there are residual connections around the self-attention layer and the feedforward layer.

The transformer decoder, illustrated in the bottom half of Figure 3.2, exactly resembles the encoder, except for an additional encoder-decoder attention layer and an additional layer normalization and residual connection. The transformer decoder's encoder-decoder attention layer requires the keys and values to come from a separate source, while the output of the self-attention layer in the transformer decoder becomes the queries.

Notice that the output of the transformer encoder (decoder) can be inputted into another transformer encoder (decoder). This is known as stacking transformer blocks. Most modern architectures stack multiple transformer blocks.

3.2.3 Training Early Fusion Models

3.2.3.1 Transfer Learning

Deep learning models typically require very large datasets. Thus, one of the principal barriers to using social science data with deep learning methods is labeled data availability. Even if data are readily available, such as social media data, it is still costly to manually annotate the data (Webb Williams et al., 2020). Transfer learning allows researchers to use state-of-the-art deep learning methods with smaller datasets. A formal definition of transfer learning is given in Pan & Yang (2010).

Informally, transfer learning takes a model trained on a source domain and uses that model to improve the learning of a target predictive function in a separate target domain. Transfer learning typically has two steps: pretraining and finetuning. During *pretraining*, a model is trained to solve general tasks over a large, general dataset. For example, ResNet (He et al., 2015) is a deep convolutional neural network trained on ImageNet (Deng et al., 2009), a dataset of 14 million images with each image belonging to one of 21,841 classes. We train the pretrained model with our data of interest during *finetuning* to learn specific annotation tasks instead of training a model from scratch. Using the pretrained model as the starting point allows us to use deep learning methods with much smaller datasets. This approach is illustrated in Figure 3.3. Intuitively, transfer learning works because the model learns general concepts during pretraining. For example, ResNet learns shapes, edges, colors, etc., which are visual concepts not exclusive to the images it was originally trained on.

Transfer learning has been most successfully used to pretrain image models, but natural language processing has also recently used the transfer learning paradigm. Pretrained transformers-based language models learn general linguistic concepts such as word order and word similarities; these models are then finetuned for specific downstream tasks (Terechshenko et al., 2021). The most popular of these pretrained language models is bidirectional encoder representations from transformers, or BERT (Devlin et al., 2018). BERT follows the transfer learning paradigm described in the previous section and consists of two steps: pretraining and finetuning. First, special tokens are appended to the inputs. A `[CLS]` token is appended to the beginning of the sentence, while `[SEP]` is appended at the end of a sentence. The corresponding output vector for `[CLS]` becomes the sentence or document embedding. `[SEP]` is used to separate two sentences if two sentences are inputted together into BERT. BERT is pretrained on two tasks during pretraining: the masked language model (MLM) and next sentence prediction (NSP). Details about these pretraining tasks can be found in Section 3.8.4 in the Supplemental Information. BERT is pretrained on BooksCorpus (800 million words) and English Wikipedia (2,500 million words).

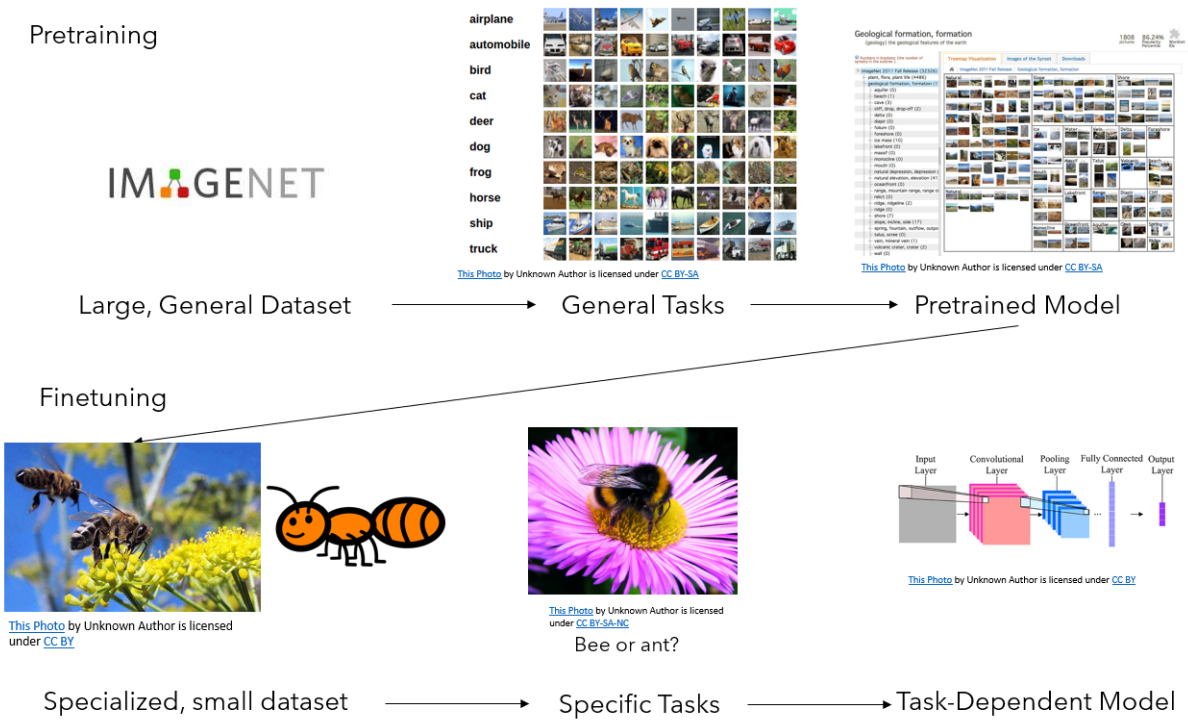


Figure 3.3: An illustration of the pretraining-finetuning pipeline classifying images of bees and ants, found in Chilamkurthy (2017). Even though the training dataset only contains 120 images of bees and ants, the finetuned model is able to correctly classify images of bees and ants with 96% accuracy.

Beyond using the `[SEP]` token between two sentences, BERT also uses token type embeddings in order to distinguish the first and second sentences. Two token type embeddings are learned: t_1 and t_2. t_1 is added to word embeddings that come from the first sentence while t_2 is added to word embeddings that come from the second sentence. Positional embeddings, previously discussed in the context of attention, are also used. See Figure 3.4 for an overview of a BERT transformer encoder block.

Pretraining BERT in this fashion, with no labeled dataset but instead using the MLM and NSP pretraining tasks, allows us to train a model that "understands language" in a general way. After BERT is pretrained, we can finetune the pretrained weights on a downstream task. Because of the transformer encoder's flexibility to model complexities in language, the parallelizability of transformers, and the ability to capture long-range dependencies between words, almost all state-of-the-art benchmark scores for natural language processing tasks are set by transformer-based pretrained language models.

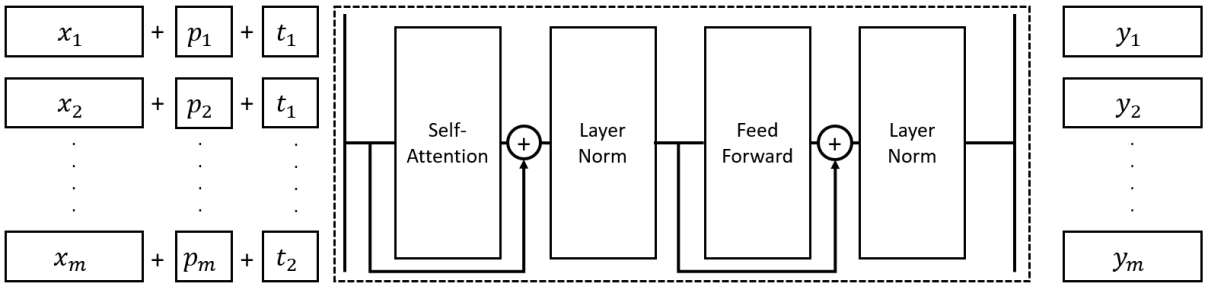


Figure 3.4: The architecture of the first BERT block (Devlin et al., 2018). Subsequent BERT blocks do not use position embeddings or token type embeddings. It exactly resembles the transformer encoder, except that token type embeddings and positional embeddings are learned and added to the input. The token type embeddings **t** indicate which sentence the embedding comes from; the position embeddings **p** take into account word order.

3.2.3.2 Pretraining Tasks for Early Fusion Multimodal Models

Because of the success of the transfer learning paradigm in computer vision and natural language processing, it is natural to develop a similar approach for multimodal models. But it is conceptually more difficult to define what "general" means in the context of multimodal data. Most state-of-the-art multimodal models are pretrained using image annotation datasets, such as Microsoft COCO (Lin et al., 2014) or Conceptual Captions (Sharma et al., 2018). Each observation in these datasets contains an image and one or several associated captions in English. General pretraining tasks similar to MLM and NSP used to pretrain BERT are used to pretrain the multimodal models. For example, VisualBERT uses two pretraining tasks: a masked language modeling with the image, where specific tokens of the text must be predicted using the surrounding text and the image, and sentence-image prediction, where the model must predict if a caption actually corresponds with the image or not (Li et al., 2019).

Pretraining multimodal models with image captioning datasets, however, presents a few issues. First, the relationship between an image and its caption is generally not the image-text relationship in real-world observations that use image and text. For example, the text of multimodal social media posts is generally not simply describing what is happening or what objects are in an image. The text and the image modalities extend or modify the overall message the post is attempting to deliver. Singh et al. (2020) find that many of these pretraining tasks do not improve the model's performance. They find that pretraining on data closer to the domain of the downstream task rather than pretraining on image captioning datasets typically yields better performance. However, such experimentation with different pretraining datasets requires computational resources unavailable to most researchers.

Moreover, even without experimentation, these pretraining tasks are computationally expensive to complete, often requiring the use of several GPUs. Even when pretrained models are available to be downloaded, additional pretraining on the data for the task of interest is required because it allows a model to adapt to a new target domain. State-of-the-art multimodal models are also not developed to accommodate missing modalities. Many models cannot be used with datasets where observations are missing a modality. Others randomly initialize image or text features for observations missing an image or text, respectively.

3.3 MARMOT Details

Figure 3.5 shows an overview of the MARMOT architecture. It contains four main components, further described in more detail below: the pretrained image model, the pretrained image captioner, modality translation, and the pretrained language model. The model is simple, and both training and inference can be accomplished using minimal computational resources.[1] Section 3.8.8 in the Supplemental Information details what hyperparameters need to be selected, learning rate schedulers, optimizers, and training strategies.

3.3.1 Pretrained Image Model

The first step involves inputting the image into a pretrained image model, such as ResNet (He et al., 2015), Inception v3 (Szegedy et al., 2015), or MobileNet v2 (Sandler et al., 2018). In our applications, we use ResNet-50 pretrained using the VirTex approach (Desai & Johnson, 2021). The VirTex approach pretrains a deep convolutional neural network using image captions instead of labeled images. Section 3.8.6 in the Supplemental Information contains more information about the VirTex pretraining approach. We scale down (or up) an arbitrary image $x_{img} \in \mathbb{R}^{3\times H_0\times W_0}$ to $\mathbb{R}^{3\times 224\times 224}$, and then use ResNet-50 pretrained using VirTex to generate a lower resolution activation map $z \in \mathbb{R}^{2048\times 7\times 7}$. Any pretrained image model will work for this step, but exact dimensions may differ.

3.3.2 Pretrained Image Captioner

At the same time, we generate an image caption for every image in our data. We use self-critical sequence training (SCST) to generate an image caption (or multiple image captions)

[1]For example, training and inference can be complete using Google Colab, a free resource that offers the use of one GPU.

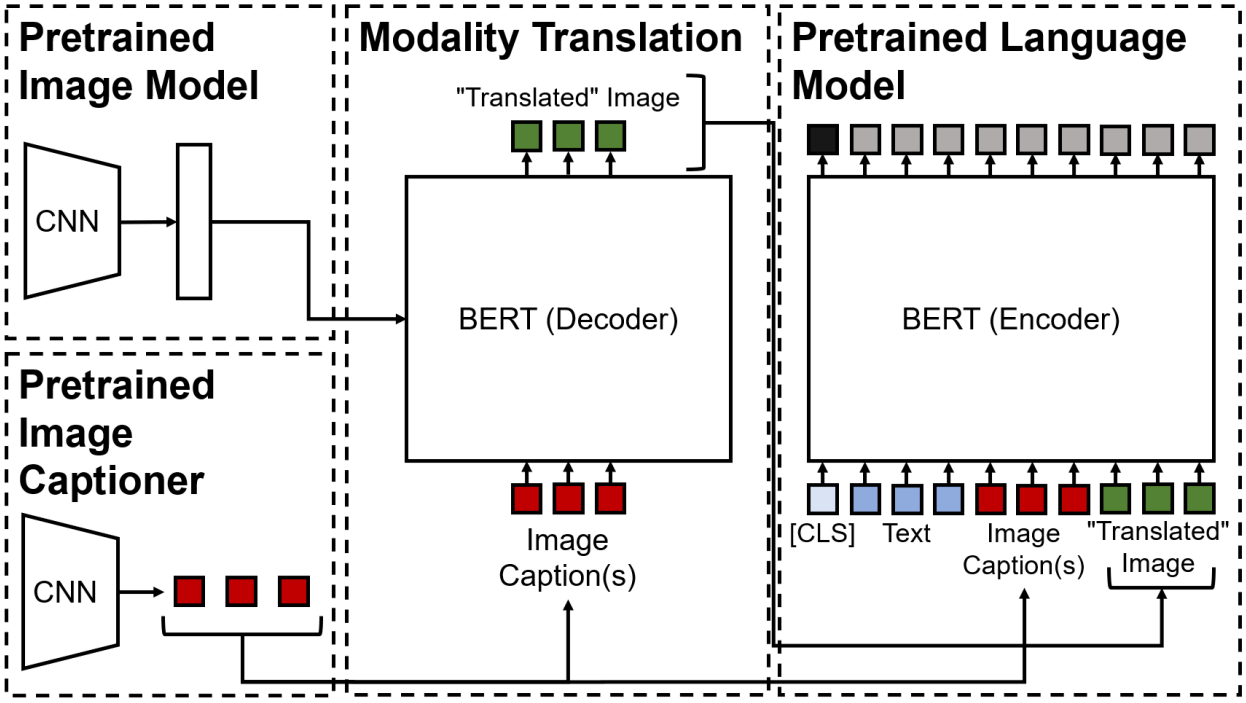


Figure 3.5: An overview of the MARMOT architecture. MARMOT uses a pretrained image model to extract image features from the input image. The model flattens these image features and passes them into the encoder-decoder layer of the transformer decoder. Image captions are generated using a pretrained image captioner. During modality translation, the image captions are the input into the BERT decoder. At the encoder-decoder attention layer of the BERT decoder, the image captions attend to the image. The output of the BERT decoder is called the "translated" image. This "translated" image is then jointly inputted with the text, image captions, and the `[CLS]` token into the BERT encoder. The joint representation is either the first outputted vector corresponding to the `[CLS]` token (in black) or the average of the rest of the output vectors (in gray).

for each image (Rennie et al., 2016), although any pretrained image captioner will suffice.[2] Generally, multiple captions are used because, during inference, pretrained image captioners may produce different captions that focus on different aspects of the image.

3.3.3 Modality Translation

Modality translation captures the spirit of pretraining used in other multimodal models such as VisualBERT (Li et al., 2019) without having to further pretrain the model on an image annotations dataset or the data for our task of interest. It also means not having to experiment to find which pretraining dataset works best (Singh et al., 2020). Recall that the goal of pretraining with an image annotations dataset in multimodal models is

[2]See Section 3.8.7 in the Supplemental Information for more details about SCST.

to adapt the underlying transformer-based pretrained language model (usually BERT) to accept both image and text features and learn patterns between image features and text features. VisualBERT's pretraining tasks reflect these goals: the masked language model with images aims to predict masked out words using the image and the unmasked text, while the sentence-image prediction aims to predict which caption actually belongs to an image (Li et al., 2019). Other models with a pretraining step use similar pretraining tasks.

Instead of pretraining BERT with an image annotations dataset, we provide BERT with explicit image captions generated in the previous step from a pretrained image captioner. A transformer decoder initialized with pretrained BERT weights learns the relationship between the image and the image captions. Inspired by neural translation models (see Figure 3.2), this step aims to "translate" the image features to text features. Simply inputting text and image features directly into a pretrained BERT model without multimodal pretraining is problematic because their input representations have different levels of abstraction (Lu et al., 2019). A learning rate that is too low will better preserve the general language understanding of BERT but learn too little from the image features; a learning rate that is too high will damage the BERT language model's pretrained weights. The image translation step maps the image features to the relevant parts of the text feature subspace to avoid this issue.

To implement image translation, we use a 1×1 convolution over the activation map z to create a new feature map $z' \in \mathbb{R}^{d \times 7 \times 7}$ (if using ResNet-50 or ResNet-152), where $d = 768$ if using $\text{BERT}_{\text{base}}$ or $d = 1024$ if using $\text{BERT}_{\text{large}}$. The decoder expects a sequence of vectors as input, so we collapse the spatial dimensions of z' into one dimension, resulting in a $z'' \in \mathbb{R}^{d \times 49}$ feature map. The image feature map z'' is inputted at the encoder-decoder layer of the BERT decoder. At the encoder-decoder layer, the image caption attends to the image, decoding the 49 d-dimensional vectors in parallel. In essence, this layer re-expresses the word embedding of each word in the image caption as a weighted sum of image features. The output of the BERT decoder is called the "translated" image.

3.3.4 Pretrained Language Model

The text, image captions, and "translated" image feature map are inputted into a transformer initialized with BERT weights. We distinguish text, image captions, and "translated" image features using different token type embeddings. Because only two token type embeddings are pretrained by BERT, we initialize the third token type embedding as the average of the two pretrained token type embeddings and add $N(0, 0.0001)$ noise to each dimension of the embedding. The model uses 0-indexed position embeddings for each segment, meaning it starts counting from position 0 for each segment. Lastly, we append the `[CLS]` token to the

beginning of the sequence, which acts as an embedding of the observation.

The BERT encoder outputs the joint image-text representation. The representation is either the average across all outputted vectors (the gray outputs in Figure 3.5) or the first outputted vector that corresponds to the `[CLS]` token (the black output in Figure 3.5); the choice is a hyperparameter. In our applications, we find the averaging approach typically works slightly better than using the output vector corresponding to the `[CLS]` token.

3.3.5 Missing Modalities

To handle missing modalities, we use attention masks. Recall, from Equations 3.3, 3.4, and 3.5, that the outputted vector $\mathbf{z}_i$ for $i \in \{1, ..., N\}$ from attention is calculated as

$$w'_{ij} = \mathbf{q}_i^T \mathbf{k}_j$$

$$w_{ij} = \text{softmax}(w'_{ij})$$

$$\mathbf{z}_i = \sum_j w_{ij} \mathbf{v}_j$$

We can set the weight w'_{ij} to $-\infty$ for observations of $\mathbf{v}_j$ that need to be excluded from calculating $\mathbf{z}_i$; this effectively sets w_{ij} to 0. This is a mechanism called masking (Vaswani et al., 2017). Masking is typically used for translation tasks when one does not want the attention mechanism to "peek" ahead and batching together texts of unequal lengths.

We take advantage of masking by simply masking out the missing modality in the pre-trained language model step. As a concrete example, a dataset may contain observations that all have text, but some are missing an image. We can associate a dummy image and dummy image caption with observations that are missing images. We can mask out the translated dummy image and dummy image caption when they are inputted into the BERT encoder. Backpropagation updates the parameters in the BERT encoder using only the text and does not update the parameters in the BERT decoder in modality translation, as the attention masks block the flow of gradients.

3.4 Application 1: Election Incidents Reported on Twitter During the 2016 U.S. General Election

3.4.1 Dataset Background

Mebane et al. (2018) collected a dataset of tweets that reported election incidents during

the 2016 U.S. general election. An election incident is an individual's personal experience with voting or some other activity in the election. Tweets came between October 1, 2016 and November 8, 2016. The dataset has a binary outcome variable indicating whether the tweet was a reported incident or not. Among the tweets that were classified as an election incident, Mebane et al. (2018) also include a deeper breakdown about what type of incident it was. The categories are line length/waiting time/polling place overcrowding, polling place event, electoral system, absentee/mail-in/provisional ballot issue, and registration. These categories are then further broken down into subcategories, which are adjectives characterizing the categories. Definitions of the subcategories can be found in Section 3.8.9 in the Supplemental Information; exact definitions of the categories can be found in Mebane et al. (2018). Human coders of the Twitter data examined all modalities to assign labels.

In this section, we focus only on the multilabel classification part of the data processing. Of the total 4,018 tweets labeled with categories and subcategories, 1,741 tweets included at least one image. We used 80% of the dataset for hyperparameter selection and training and set aside the remaining 20% of the dataset as a test set.

3.4.2 Results

Mebane et al. (2018) use a text-only ensemble classifier consisting of logistic regression, multinomial naïve Bayes, and linear support vector machine. Mebane et al. (2018) binarize each category and subcategory, meaning that they treat every category and the subcategories under each category as individual binary classification problems. The ensemble classifier only uses text features from the tweets, ignoring all images. Mebane et al. (2018) also append the date and location of the tweet to the text of the tweet.

We take the same approach to make results comparable to results found in Mebane et al. (2018): we also binarize each category and subcategory and we append the date and location of the tweet to the text of the tweet. MARMOT representations were classified using a two-layer feedforward neural network with a ReLU activation function and we used the cross-entropy loss function. Section 3.8.10 in the Supplemental Information contains information about hyperparameters. Results from the ensemble classifier and MARMOT are in Table 3.1; the results are expressed as F1 scores over the positive class. If a tweet had an image, three image captions were generated using self-critical sequence training.

MARMOT outperforms the ensemble classifier on all categories and subcategories except for "Line Length: No crowd or no line." In that specific subcategory, MARMOT matches the performance of the text-only ensemble classifier. There are improvements in the F1 scores where we would intuitively expect images to play a role, such as in the short and long

	Ensemble	MARMOT	Support	# Pictures
Not an Incident	0.66	**0.72**	1149	330
Line Length	0.91	**0.92**	1045	440
(a) No crowd or no line	0.61	0.61	85	36
(b) Small crowd or short line	0.21	**0.30**	91	31
(c) Large crowd or long line	0.82	**0.88**	869	373
Polling Place Event	0.78	**0.82**	1477	721
(a) Did not function as expected	0.08	**0.49**	86	19
(b) Neutral observation	0.47	**0.63**	481	255
(c) Functioned properly	0.63	**0.68**	910	447
Electoral System	0.63	**0.67**	596	244
(a) Did not function properly	0.05	**0.15**	89	28
(b) No comment on function	0.65	**0.73**	473	211
Absentee / Mail-in Voting Issue	0.87	**0.88**	1702	748
(a) Did not function properly	0.34	**0.53**	121	28
(b) Neutral observation	0.60	**0.71**	613	296
(c) Functioned properly	0.70	**0.77**	968	424
Registration	0.85	**0.88**	475	190
(a) Not able to register	0.17	**0.62**	59	9
(b) Neutral observation	0.84	**0.86**	339	166
(c) Able to register	0.50	**0.69**	77	15

Table 3.1: Binarized classifier performance across the ensemble classifier and MARMOT over the election incidents dataset with 4,018 tweets. Categories are in bold, while subcategories are listed with letters. Results of the ensemble classifier and MARMOT are over a test set that is 20% of all tweets and the results are F1 scores on the positive class. For a definition of the F1 metric, refer to Section 3.8.5 in the Supplemental Information. The total number of observations (across both the training and test sets) that belong to each category and subcategory is noted in the third column. The total number of images (across both the training and test sets) for each category and subcategory is noted in the fourth column. The results of the ensemble classifier come directly from Mebane et al. (2018). Numbers are rounded to two decimal places because the results of the ensemble classifier were originally reported to only two digit places.

line subcategories under the "Line Length" category. There are also improvements in some subcategories that contain few images. For example, the text-only ensemble classifier struggled with the subcategory "Polling Place Event: Did not function as expected." MARMOT performed significantly better, despite the subcategory containing very few images: only 19 of the 86 tweets in this subcategory had an image. In other words, it is not immediately apparent that all improvements are the direct result of including images. MARMOT uses BERT, which consistently outperforms other text classifiers as well (Devlin et al., 2018). We turn to look at model variants to examine this possibility.

3.4.3 Model Variants

We look at two variants of the model: the first uses only the text from each observation with the standard BERT encoder, and the second uses the text and image captions with the BERT encoder but does not use the translated image. The results of the two model variants, along with the full MARMOT model, are in Table 3.2, which details the F1 score over the positive class for each model variant.

We can attribute many of the improvements over the text-only ensemble classifier baseline to BERT. For example, most of the improvements in the category "Line Length: Large crowd or long line" came from BERT. The inclusion of images did offer an improvement in the F1 score in several subcategories—namely, "Not an Incident," "Line Length: No crowd or no line," "Line Length: Small crowd or short line," "Polling Place Event: Did not function as expected," "Polling Place Event: Functioned properly," "Electoral System: Did not function properly," "Absentee / Mail-in Voting Issue: Did not function properly," "Absentee / Mail-in Voting Issue: Neutral observation," and "Registration: Able to register." Therefore, not all performance gains over the text-only ensemble classifier baseline resulted from using a more powerful text representation architecture.

The *lack* of an image may help MARMOT to predict many more tweets correctly. Recall that MARMOT deals with a missing image in an observation by masking out a dummy image. The learned representation is different for observations that only have text versus observations with both an image and text. The differences in these representations can be a pattern differentiating a positive classification from a negative classification. MARMOT may be helpful in both situations where images play a role in an observation's classification and situations where the lack of an image may further clarify an observation's classification.

	BERT, Text Only	BERT, Text and Image Captions	MARMOT
Not an Incident	0.65	0.71	0.72
Line Length	0.92	0.92	0.92
(a) No crowd or no line	0.44	0.44	0.61
(b) Small crowd or short line	0.21	0.19	0.30
(c) Large crowd or long line	0.86	0.87	0.88
Polling Place Event	0.81	0.81	0.82
(a) Did not function as expected	0.36	0.41	0.49
(b) Neutral observation	0.55	0.58	0.63
(c) Functioned properly	0.65	0.65	0.68
Electoral System	0.66	0.65	0.67
(a) Did not function properly	0.11	0.11	0.15
(b) No comment on function	0.72	0.72	0.73
Absentee / Mail-in Voting Issue	0.87	0.87	0.88
(a) Did not function properly	0.42	0.49	0.53
(b) Neutral observation	0.67	0.67	0.71
(c) Functioned properly	0.77	0.77	0.77
Registration	0.89	0.87	0.88
(a) Not able to register	0.50	0.52	0.62
(b) Neutral observation	0.88	0.85	0.86
(c) Able to register	0.65	0.63	0.69

Table 3.2: Results of MARMOT model variants over the election incidents dataset with 4,018 tweets. The first column reports the F1 scores over each binarized category or subcategory from a model using a pretrained BERT encoder with only text input. The second column reports the results from a model using a pretrained BERT encoder with text and image captions as inputs, but does not use the translated image. The third column reports the results from the full MARMOT model.

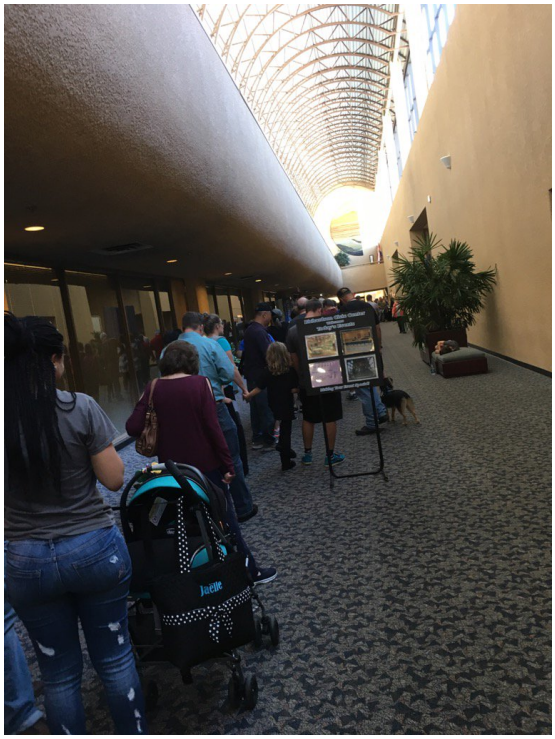
3.4.4 Examples of Successful Classifications

We take a qualitative look at some of the classifier results using the MARMOT representations to understand better the strengths of the MARMOT multimodal representations.

3.4.4.1 Line Length: Large Crowd or Long Line

MARMOT outperformed the text-only ensemble classifier in the large crowd or long line subcategory. Figure 3.6 shows examples of tweets that were correctly classified as long lines at polling places by MARMOT.

"Line to vote in astoria queens."



"The line for first day of early voting, Richardson TX"

Figure 3.6: Two example tweets of long lines at polling stations.

The text-only BERT classifier, however, also correctly classified most of these types of examples. MARMOT only did marginally better in this subcategory in terms of the F1 score. Plausibly, the text-only models can learn to classify any tweet mentioning a line at a polling station as a report of a long line at a polling station. However, we can see that MARMOT is slightly more nuanced than that. Figure 3.7 is an example of a tweet that reports a long line at a polling place. It notes that the polls are open in Georgia, but it did

not refer to Peachtree Corners as a polling place. The image depicts a scene where there is a long line at a polling station, indicated by the "Vote Here" sign. The text-only BERT classifier misclassified this tweet, while MARMOT correctly classified the tweet as depicting a long line at a polling place.

Figure 3.7: "Polls are open in GA. Proud to cast my ballot. Longest line ever seen in Peachtree Corners!"

3.4.4.2 Polling Place Events: Functioned Properly

Images play a role in classifying tweets of polling places functioning correctly, particularly tweets involving people smiling after voting. Figure 3.8 contains two example tweets of polling places functioning correctly. In the first example, the person advocates for voting but does not directly indicate with the text that he successfully voted. The image depicts a person with an "I Voted" sticker on his arm, indicating that he was successful in voting and thus that a polling place functioned correctly. The text-only classifier failed to classify this observation as polling place functioning correctly, while MARMOT correctly classified

this tweet as a tweet indicating a polling place functioning correctly. In the second example, the individual reports that he was honored to vote and included a picture of himself smiling. The text indicates that he voted at a polling station but does not describe his experience; the picture, on the other hand, indicates that he was happy to vote, suggesting that the polling place functioned correctly. MARMOT correctly classifies this tweet as a tweet indicating a polling place functioning correctly; the text-only classifier, however, misclassified this tweet.

"#Voting makes you #Stronger. #Vote The work doesn't stop after today."

"I felt the same way as I arrived to the polling station. As an immigrant, Im honored to be able to vote."

Figure 3.8: Two example tweets of polling places functioning correctly.

3.5 Application 2: Hateful Memes

There is increasing interest in social sciences and computer science around detecting and analyzing hate speech on social media. Siegel et al. (2019) looked at 750 million tweets and did not find an increase in hate speech or white nationalistic language on Twitter. Siegel & Badaan (2020) examine what counter-speech initiatives are most effective at reducing sectarian hate speech online. MacAvaney et al. (2019) and Davidson et al. (2017) examine the machine learning approaches to detecting hate speech on social media.

The Hateful Memes dataset aims to help develop models that more effectively detect multimodal hateful content. Besides being a well-curated dataset for building models that detect multimodal hate speech, the Hateful Memes dataset is also useful for comparing how

MARMOT performs against the state-of-the-art multimodal models because every observation has both text and image. We find that MARMOT shows significant improvements over the results of benchmark state-of-the-art multimodal models on this classification problem, suggesting that MARMOT does not lack performance over other multimodal classifiers even though it does not require pretraining on an image annotations dataset and it can calculate representations for observations missing modalities.

3.5.1 Dataset Background

Facebook Research recently released the Hateful Memes dataset to develop and test multimodal models (Kiela et al., 2020). This dataset was also used in the Hateful Memes challenge.[3] Kiela et al. (2020) define hate as follows: "A direct or indirect attack on people based on characteristics, including ethnicity, race, nationality, immigration status, religion, caste, sex, gender identity, sexual orientation, and disability or disease. We define attack as violent or dehumanizing (comparing people to non-human things, e.g. animals) speech, statements of inferiority, and calls for exclusion or segregation. Mocking hate crime is also considered hate speech."

The problem requires a multimodal model to solve because the image and text may, on their own, not be hateful, but the combination of the two may be hateful. To give an example of an unkind (but not hateful) meme that captures this idea, consider an image of a skunk paired with the text, "Love the way you smell today." Neither the text nor the image is mean on its own. However, the combination of the two modalities makes it an unkind meme.

The dataset contains 10,000 memes. The validation set is 5% of the data, the test set is 10% of the data, and the rest of the data is set aside as training data. Most memes were actually found on Facebook, but they also created a set of benign confounders, which are artificially-created memes based on an actual hateful meme that has been made non-hateful by replacing the image or replacing the text. In total, there are five types of memes in the dataset: multimodal hate, where the text and image on their own are not hateful but are hateful when paired together; unimodal hate, where the text or image (or both) are already hateful on their own; benign text confounders; benign image confounders; and random non-hateful examples. Multimodal hate makes up 40% of the data, unimodal hate makes up 10% of the data, benign text confounders make up 20% of the data, benign image confounders make up 20% of the data, and 10% of the data are random non-hateful. The dataset, however, does *not* specify which observations belong to which categories, meaning

[3]For more information about this challenge, see https://ai.facebook.com/blog/hateful-memes-challenge-and-data-set/.

we could not use the type of meme as part of the classification process.[4] The outcome of interest is whether a meme is hateful or not. The Hateful Memes dataset has 5,000 hateful memes and 5,000 non-hateful memes.

3.5.2 Results

For MARMOT, we used a two-layer feedforward neural network with a ReLU activation function as the classifier and we used the cross-entropy loss function. We generated three captions for each image using self-critical sequence training. For information about hyperparameter selection, see Section 3.8.11 in the Supplemental Information. The accuracy learning curve of MARMOT over the validation set, used to assess potential overfitting, can be found in Section 3.8.12 in the Supplemental Information. Kiela et al. (2020) provide results over the test set for many multimodal models, including ViLBERT, VisualBERT, and MMBT, which are state-of-the-art multimodal classifiers. Results over the test set can be found in Table 3.3. Results over the validation set can be found in Section 3.8.13 in the Supplemental Information.

MARMOT outperforms the benchmark state-of-the-art multimodal classifiers on both accuracy and area under the receiver operating characteristic curve (AUC).[5] MARMOT also outperforms the pretrained variants of ViLBERT and VisualBERT. MARMOT does not trade off performance on classification problems to accommodate potentially missing modalities—it can perform just as well, if not better, than the baseline state-of-the-art multimodal models while still retaining this critical property that makes it useful for social science research. MARMOT's accuracy improves further when MARMOT is used in a deep ensemble, which uses the majority predicted class across 11 separate iterations of MARMOT (Lakshminarayanan et al., 2017). In the Hateful Memes challenge, MARMOT finished in the top 1% of all contestants.[6]

[4]The noted existence of these meme category types was a significant issue during the Hateful Memes challenge. Some contestants designed architectures to predict the category of each meme in the dataset. These category-specific architectures led to extraordinary performances that far exceeded human coder baselines. As these category types would not exist in any real-life setting involving memes, our approach does not attempt to infer the category types of each meme as part of the prediction pipeline.

[5]Intuitively, AUC is the probability that a classifier will rank a randomly chosen positive example over a randomly chosen negative example. For a more detailed set of definitions of the performance metrics, refer to Section 3.8.5 in the Supplemental Information.

[6]Placement in the challenge was based on AUC.

Model	**Accuracy**	**AUC**
Image - Grid	0.5200	0.5263
Image - Region	0.5213	0.5592
Text BERT	0.5920	0.6508
Late Fusion	0.5966	0.6475
Concat BERT	0.5913	0.6579
MMBT - Grid	0.6006	0.6792
MMBT - Region	0.6023	0.7073
ViLBERT	0.6230	0.7045
VisualBERT	0.6320	0.7133
ViLBERT CC	0.6110	0.7003
VisualBERT COCO	0.6473	0.7141
MARMOT	0.6760	**0.7530**
MARMOT - Deep Ensemble	**0.6920**	0.7493

Table 3.3: Accuracy and area under the receiver operating characteristic curve (AUC) performance metrics across the 11 baseline models, MARMOT, and MARMOT in a deep ensemble over the test set of the Hateful Memes dataset. For definitions of these metrics, refer to Section 3.8.5 in the Supplemental Information. "Grid" means that image features derived from ResNet-152 were used as input features. "Region" means that segmented image features are derived using Faster R-CNN (Ren et al., 2015), rather than the entire image, were used as input features. Concat BERT means that BERT embedding features were concatenated with image features. Results are over the test set, which is 1,000 memes. VisualBERT COCO means that VisualBERT was pretrained over the MS COCO dataset (Lin et al., 2014). ViLBERT CC means ViLBERT was pretrained over the Conceptual Captions dataset (Sharma et al., 2018). "Deep Ensemble" means that the final prediction for each observation was the majority predicted class across 11 iterations of MARMOT.

3.5.3 Model Variants

We analyze two variants of the model: the first uses only the text from each observation with BERT and the second uses the text and the image captions with BERT but not the translated image. The results of the two model variants are in Table 3.4.

Model Variant	**Accuracy**	**AUC**
BERT with Text Only	0.5920	0.6508
BERT with Text and Image Captions	0.6510	0.7469
MARMOT	0.6760	0.7530
MARMOT - Deep Ensemble	0.6920	0.7493

Table 3.4: Results of the MARMOT model variants over the test set of the Hateful Memes dataset. The first model variant uses only the text; the result is taken directly from Kiela et al. (2020). The second model variant uses the text and image captions generated during modality translation, but the translated image is not used. The third model variant is the full MARMOT model. The fourth model variant is MARMOT used in a deep ensemble.

The addition of the generated image captions accounts for most of the improvement over using the text alone with a BERT model: there is a 9 point improvement in AUC and a 6 point improvement in accuracy with the image captions. As the captions are text, BERT can easily learn the relationships between the text and the image captions. There is a further improvement when using the full MARMOT model, especially in terms of accuracy. Image translation can capture additional information about the image that is useful for the detection of hateful memes.

3.5.4 Examples of Successful Classifications

To more intuitively understand how MARMOT is classifying the memes as hateful or not, we qualitatively look at a few classification examples where MARMOT was successful. Please note that some of these examples may contain sensitive or offensive content.

The most difficult memes to classify as hateful are the multimodal hate memes. These are the memes where the text alone is not hateful, and the image alone is not hateful, but when put together the meme becomes hateful. Figure 3.9 shows an example of a multimodal hateful meme which MARMOT correctly classified as hateful.

The model may learn that the use of the words “dishwasher” or “sandwich maker” in a memes setting usually implies that the meme is misogynistic. After all, across the training and validation datasets, most of the memes containing the phrase “dishwasher” or “sandwich maker” are hateful. To further demonstrate evidence that the model is learning relationships between the image and the text, we first look at an example, shown in Figure 3.10, consisting

Figure 3.9: An example of a multimodal hateful meme. This meme was correctly classified by MARMOT as hateful. ©Getty Images.

of three memes. The memes on the left and in the center have the same image but different text (benign text confounder). The memes on the left and the right have different images but the same text (benign image confounder). MARMOT correctly classifies the first meme on the left as hateful and correctly classifies the meme in the center and on the right as non-hateful.[7]

Figure 3.10: An example of a multimodal hateful meme with a benign text confounder counterpart and benign image confounder counterpart. MARMOT correctly classifies the meme on the left as hateful and the meme in the center and on the right as non-hateful. ©Getty Images.

To take this analysis one step further, we create visualizations of the attention weights of the BERT encoder used in the last stage of MARMOT.[8] To be clear, these visualizations do

[7]Specifically, the probability of each meme being hateful, using MARMOT, is 0.929, 0.0008, and 0.013, respectively.

[8]Because of computational constraints, these visualizations could not be created for tweets of election

not causally show why MARMOT correctly classified a specific observation; in other words, we are not trying to learn about intent or meaning behind how the MARMOT representations are constructed. Instead, these visualizations show that MARMOT can learn important associations between the image and text. Figure 3.11 shows the attention weights of the 12th attention head of the 10th transformer layer in the BERT encoder. The line weight reflects the attention weight between the attending tokens and the attended tokens. In the hateful example (the meme on the left in Figure 3.10), the token "russian" from the text of the meme shares a high attention weight with the token "cow" from the image caption.[9] In the non-hateful benign text confounder example (the meme in the center in Figure 3.10), the token "i" from the text of the meme shares a lower attention weight with the token "cow" from the image caption compared to the hateful example. In the non-hateful benign image confounder example (the meme on the right in Figure 3.10), there is essentially no attention weight learned between the token "russian" and the image caption. The attention weights suggest that MARMOT learns different relationships among the images and text between the hateful and non-hateful examples, even when the text or image is the same.

We look at another example, shown in Figure 3.12 where the text is the same, but the images are different. MARMOT, again, correctly classifies the meme on the left as hateful and correctly classifies the meme on the right as non-hateful.[10]

To take a closer look at what the model learns between the image and the text, we again create visualizations of the attention weights of the BERT encoder used in the last stage of MARMOT. Figure 3.13 shows the attention weights of the 12th attention head of the 10th layer in the BERT encoder. Again, the line weight reflects the attention weight between the attending tokens and the attended tokens. In the hateful example, the token "dish" shares a high attention weight with the token "woman" from the image caption. In the non-hateful example, the token "dish" shares a high attention weight with the token "man" from the image caption. Again, the attention weights suggest that MARMOT learns different relationships among the images and text between the non-hateful and hateful examples, even when the text is the same.

3.6 Conclusion and Future Directions

Labeling is usually required to identify posts of interest for most content analyses of social media posts. Human coders usually consider image and text if both are available. However,

incidents during the 2016 U.S. general election.

[9]Self-critical sequence training misidentifies the animal in the picture as a cow.

[10]Specifically, the probability of each meme being hateful, using MARMOT, is 0.987 and 0.225, respectively.

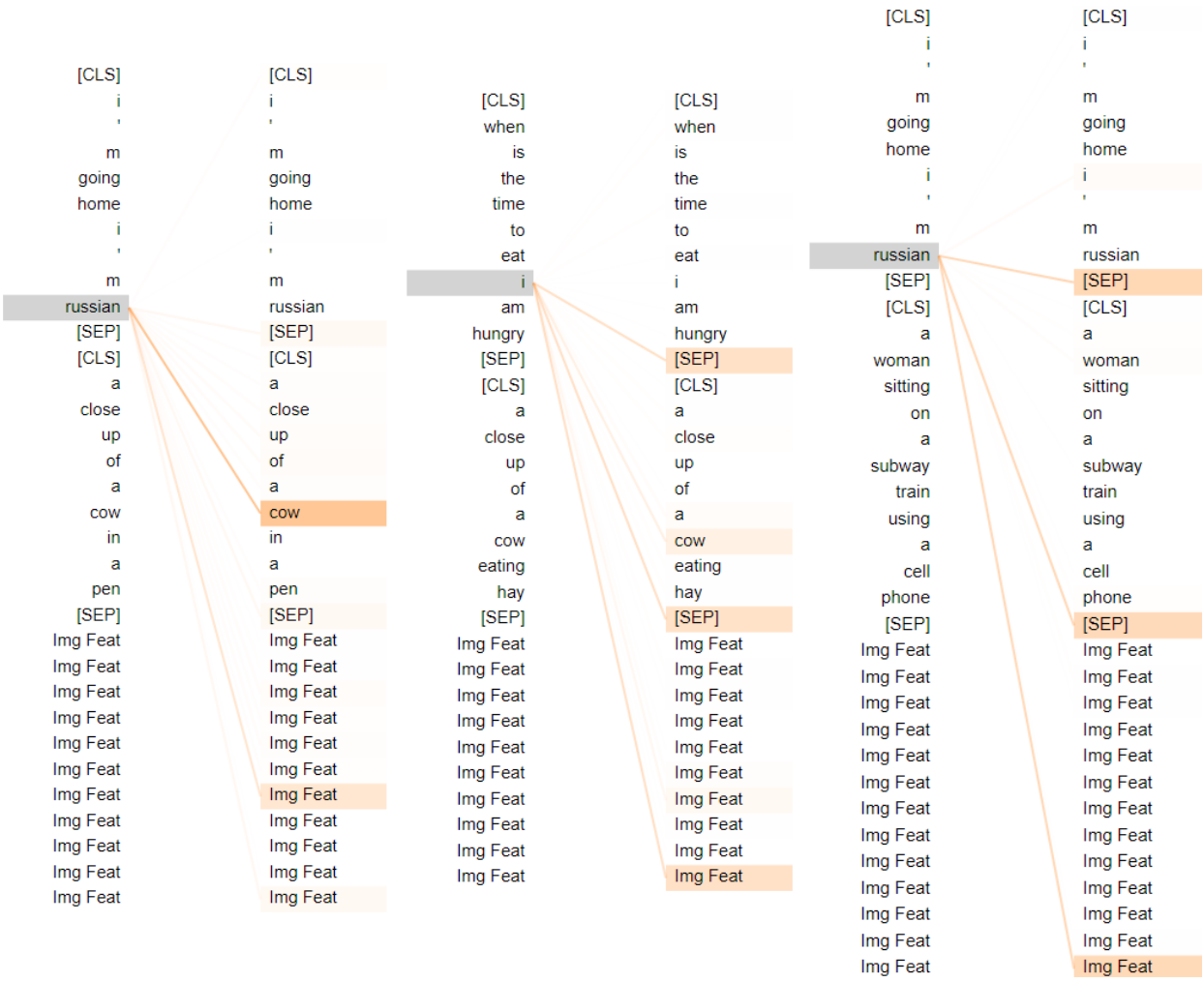


Figure 3.11: A visualization of the attention weights from the 12th attention head of the 10th layer of the BERT encoder from the last step of MARMOT for the three memes from Figure 3.10. The visualizations on the left, center, and right correspond with the left, center, and right memes, respectively, from Figure 3.10. The line weight indicates the attention weight. Visualizations were created using the package described in Vig (2019). The token "Img Feat" indicates an inputted image feature, which does not translate to a word token.

Figure 3.12: An example of a multimodal hateful meme with a benign image confounder counterpart. MARMOT correctly classifies the meme on the left as hateful and the meme on the right as non-hateful. ©Getty Images.

automated machine approaches are almost always unimodal, focusing exclusively on either the text or images. This can create potential biases in downstream analyses of the machine classified data. Ways of representing text or image as quantitative features are well-known in the computer science literature, but multimodal representations—joint representations of image and text—are still a budding field in computer science. Many state-of-the-art multimodal models require that every observation have both image and text and require extensive pretraining on image annotation datasets. The computational costs and the reality that many datasets of interest for social scientists contain observations with missing modalities render these state-of-the-art multimodal models essentially unusable.

We present a new method that calculates joint image-text representations called multimodal representations using modality translation, or MARMOT, to solve both issues. It eschews pretraining, meaning that training and inference can be completed with minimal computational resources. It also leverages off-the-shelf pretrained models: good performance results can be achieved on relatively small datasets. Lastly, it can calculate representations for observations missing modalities.

Specifically, we first note that the pretraining over image annotation datasets used in models such as VisualBERT (Li et al., 2019) is done to adapt the underlying BERT model to accept both image and text features, rather than only the text features it was initially pretrained on, and to relate image features with text features. Additional pretraining with the data of interest is done to adapt further the model to the target domain.

To capture the same spirit of this process without undergoing the computationally expensive pretraining procedure, we develop a process called modality translation. First, we generate image captions directly using a pretrained image captioner. To learn patterns

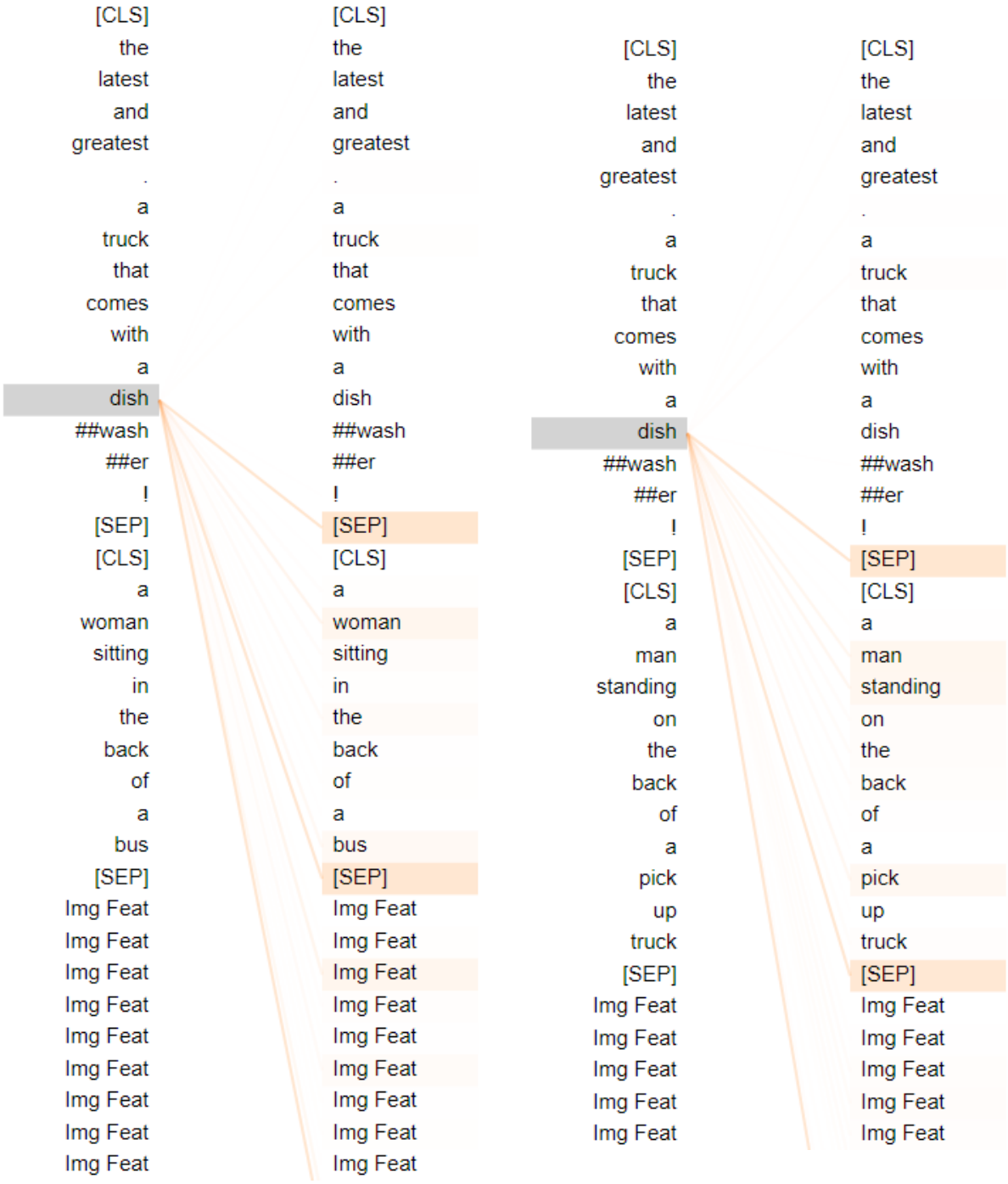


Figure 3.13: A visualization of the attention weights from the 12th attention head of the 10th layer of the BERT encoder from the last step of MARMOT for the two memes from Figure 3.12. The visualizations on the left and right correspond with the hateful and non-hateful memes, respectively, from Figure 3.12. The line weight indicates the attention weight. Visualizations were created using the package described in Vig (2019). The token "Img Feat" indicates an inputted image feature, which does not translate to a word token.

between the image and its image caption, we use a transformer decoder initialized with pretrained BERT weights. The image captions are inputted into the BERT decoder, and the image features, derived from a pretrained image model, are inputted at the encoder-decoder attention layer. This process calculates what we call the translated image.

After obtaining the image caption and the translated image through modality translation, we jointly input the text of the observation, the text of the image caption, and the translated image as one sequence into the transformer encoder initialized with pretrained BERT weights. The three parts of the sequence are differentiated using different token type embeddings. The joint representation is either the outputted vector corresponding to the `[CLS]` token or the average across all vectors of the outputted sequence.

We apply MARMOT to two classification tasks: classifying tweets of election incidents during the 2016 U.S. general election (Mebane et al., 2018) and identifying hateful memes (Kiela et al., 2020). In the election incidents dataset, all observations have text, but only some have images. MARMOT outperforms the ensemble classifier found in Mebane et al. (2018) in 19 of 20 categories in multilabel classifications of tweets (and equals the performance in the last category). However, with the election incidents dataset, we cannot use other state-of-the-art multimodal classifiers because some observations do not contain an image. We turn to the Hateful Memes dataset, where all observations have both text and image. MARMOT improves upon the benchmark state-of-the-art multimodal models in terms of accuracy and area under the receiver operating characteristic curve (AUC), even outperforming pretrained multimodal classifiers. MARMOT improves the best result set by VisualBERT in terms of accuracy from 0.6473 to 0.6760 and in terms of AUC from 0.7141 to 0.7530. Using MARMOT in a deep ensemble further improves accuracy to 0.6920. Qualitatively looking at examples from the test set, MARMOT can correctly classify multimodal hate memes—the memes where the text alone and the image alone are not hateful but combined are hateful. Visualizations of the attention weights used in the BERT encoder, the final stage of MARMOT, further suggest that the architecture is learning important associations between the text and image when making predictions. Thus, MARMOT does not trade off performance on classification problems to accommodate missing modalities. It performs just as well if not better than benchmark state-of-the-art multimodal models while having this key property that makes it useful for political science, communications, and social media research.

There are still many future directions for this project. Numerous methodological details still require experimentation, such as using newer pretrained language models, using image regions derived from a pretrained image segmentation model such as Faster R-CNN (Ren et al., 2015) instead of image features from ResNet, using different training strategies, and

using MARMOT in conjunction with other pretrained vision-and-language models if all observations have both image and text. We also aim to extend MARMOT to non-social media applications, such as multimodal elements in newspapers and other forms of media.

We have also not used MARMOT representations in any substantive applications, such as using the predictions in a regression framework. In a regression framework, MARMOT can be potentially used in two ways. First, the predictions from using MARMOT representations can be used as a covariate. Second, the predictions from using MARMOT representations can be used as an outcome variable. Using the predictions using MARMOT representations as either the outcome variable or covariate without any adjustments can lead to bias or uncontrolled variance. Fong & Tyler (2020) discuss how to adjust machine predictions when used as covariates, and Wang et al. (2020) discuss how to adjust machine predictions when used as the outcome variable. The two frameworks apply to machine prediction pipelines generally, meaning such frameworks can be jointly used with MARMOT to make its predictions useful in substantive applications.

3.7 References

Ba, Jimmy Lei, Kiros, Jamie Ryan, & Hinton, Geoffrey E. (2016). Layer normalization.

Bahdanau, Dzmitry, Cho, Kyunghyun, & Bengio, Yoshua (2014). Neural machine translation by jointly learning to align and translate.

Barberá, Pablo, Casas, Andreu, Nagler, Jonathan, Egan, Patrick J., Bonneau, Richard, Jost, John T., & Tucker, Joshua A. (2019). Who leads? who follows? measuring issue attention and agenda setting by legislators and the mass public using social media data. *American Political Science Review*, *113*(4), 883–901.

Blei, David M., Ng, Andrew Y., & Jordan, Michael I. (2003). Latent dirichlet allocation. *J. Mach. Learn. Res.*, *3*(null), 993–1022.

Bloem, Peter (2019). Transformers from scratch.

Casas, Andreu & Webb Williams, Nora (2019). Images that matter: Online protests and the mobilizing role of pictures. *Political Research Quarterly*, *72*(2), 360–375.

Chang, Charles & Masterson, Michael (2020). Using word order in political text classification with long short-term memory models. *Political Analysis*, *28*(3), 395–411.

Chilamkurthy, Sasank (2017). *Transfer Learning for Computer Vision Tutorial*.

Davidson, Thomas, Warmsley, Dana, Macy, Michael, & Weber, Ingmar (2017). Automated hate speech detection and the problem of offensive language.

Deng, Jia, Dong, Wei, Socher, Richard, Li, Li-Jia, Kai Li, & Li Fei-Fei (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, (pp. 248–255).

Desai, Karan & Johnson, Justin (2021). Virtex: Learning visual representations from textual annotations.

Devlin, Jacob, Chang, Ming-Wei, Lee, Kenton, & Toutanova, Kristina (2018). BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, *abs/1810.04805*. URL: http://arxiv.org/abs/1810.04805.

Fong, Christian & Tyler, Matthew (2020). Machine learning predictions as regression covariates. *Political Analysis*, 1–18.

Hastie, Trevor, Tibshirani, Robert, & Friedman, Jerome (2000). *Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer.

He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, & Sun, Jian (2015). Deep residual learning for image recognition.

Hochreiter, Sepp & Schmidhuber, Jürgen (1997). Long short-term memory. *Neural Comput.*, *9*(8), 1735–1780.

Huang, Gao, Li, Yixuan, Pleiss, Geoff, Liu, Zhuang, Hopcroft, John E., & Weinberger, Kilian Q. (2017). Snapshot ensembles: Train 1, get m for free.

Kiela, Douwe, Bhooshan, Suvrat, Firooz, Hamed, & Testuggine, Davide (2019). Supervised multimodal bitransformers for classifying images and text.

Kiela, Douwe, Firooz, Hamed, Mohan, Aravind, Goswami, Vedanuj, Singh, Amanpreet, Ringshia, Pratik, & Testuggine, Davide (2020). The hateful memes challenge: Detecting hate speech in multimodal memes.

Kingma, Diederik P. & Ba, Jimmy (2014). Adam: A method for stochastic optimization.

Lakshminarayanan, Balaji, Pritzel, Alexander, & Blundell, Charles (2017). Simple and scalable predictive uncertainty estimation using deep ensembles.

Le, Quoc & Mikolov, Tomas (2014). Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, (pp. II–1188–II–1196). JMLR.org.

Li, Liunian Harold, Yatskar, Mark, Yin, Da, Hsieh, Cho-Jui, & Chang, Kai-Wei (2019). Visualbert: A simple and performant baseline for vision and language.

Lin, Tsung-Yi, Maire, Michael, Belongie, Serge, Bourdev, Lubomir, Girshick, Ross, Hays, James, Perona, Pietro, Ramanan, Deva, Zitnick, C. Lawrence, & Dollár, Piotr (2014). Microsoft coco: Common objects in context.

Liu, Kuan, Li, Yanen, Xu, Ning, & Natarajan, Prem (2018). Learn to combine modalities in multimodal deep learning.

Lu, Jiasen, Batra, Dhruv, Parikh, Devi, & Lee, Stefan (2019). Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks.

MacAvaney, Sean, Yao, Hao-Ren, Yang, Eugene, Russell, Katina, Goharian, Nazli, & Frieder, Ophir (2019). Hate speech detection: Challenges and solutions. *PLOS ONE*, *14*(8), 1–16.

Mebane, Jr., Walter R., Wu, Patrick Y., Woods, Logan, Klaver, Joseph, Pineda, Alejandro, & Miller, Blake (2018). Observing election incidents in the united states via twitter: Does who observes matter? Working Paper.

Mikolov, Tomas, Sutskever, Ilya, Chen, Kai, Corrado, Greg, & Dean, Jeffrey (2013). Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, (pp. 3111–3119)., Red Hook, NY, USA. Curran Associates Inc.

Mogadala, Aditya (2015). Polylingual multimodal learning.

Pan, Jennifer & Siegel, Alexandra A. (2020). How saudi crackdowns fail to silence online dissent. *American Political Science Review*, *114*(1), 109–125.

Pan, Sinno Jian & Yang, Qiang (2010). A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, *22*(10), 1345–1359.

Pennington, Jeffrey, Socher, Richard, & Manning, Christopher D. (2014). Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, (pp. 1532–1543).

Ren, Shaoqing, He, Kaiming, Girshick, Ross, & Sun, Jian (2015). Faster r-cnn: Towards real-time object detection with region proposal networks.

Rennie, Steven J., Marcheret, Etienne, Mroueh, Youssef, Ross, Jarret, & Goel, Vaibhava (2016). Self-critical sequence training for image captioning.

Rumelhart, David E., Hinton, Geoffrey E., & Williams, Ronald J. (1986). Learning representations by back-propagating errors. *Nature*, *323*(6088), 533–536.

Rush, Alexander (2018). The annotated transformer.

Sandler, Mark, Howard, Andrew, Zhu, Menglong, Zhmoginov, Andrey, & Chen, Liang-Chieh (2018). Mobilenetv2: Inverted residuals and linear bottlenecks.

Sharma, Piyush, Ding, Nan, Goodman, Sebastian, & Soricut, Radu (2018). Conceptual captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, (pp. 2556–2565)., Melbourne, Australia. Association for Computational Linguistics.

Siegel, Alexandra A. & Badaan, Vivienne (2020). #no2sectarianism: Experimental approaches to reducing sectarian hate speech online. *American Political Science Review*, *114*(3), 837–855.

Siegel, Alexandra A., Nikitin, Evgenii, Barberá, Pablo, Sterling, Joanna, Pullen, Bethany, Bonneau, Richard, Nagler, Jonathan, & Tucker, Joshua (2019). Trumping hate on twitter? online hate speech and white nationalist rhetoric in the 2016 us election campaign and its aftermath. Working Paper.

Singh, Amanpreet, Goswami, Vedanuj, & Parikh, Devi (2020). Are we pretraining it right? digging deeper into visio-linguistic pretraining.

Su, Weijie, Zhu, Xizhou, Cao, Yue, Li, Bin, Lu, Lewei, Wei, Furu, & Dai, Jifeng (2019). Vl-bert: Pre-training of generic visual-linguistic representations.

Sun, Chen, Myers, Austin, Vondrick, Carl, Murphy, Kevin, & Schmid, Cordelia (2019). Videobert: A joint model for video and language representation learning.

Szegedy, Christian, Vanhoucke, Vincent, Ioffe, Sergey, Shlens, Jonathon, & Wojna, Zbigniew (2015). Rethinking the inception architecture for computer vision.

Tan, Hao & Bansal, Mohit (2019). Lxmert: Learning cross-modality encoder representations from transformers.

Terechshenko, Zhanna, Linder, Fridolin, Padmakumar, Vishakh, Liu, Michael, Nagler, Jonathan, Tucker, Joshua A., & Bonneau, Richard (2021). A comparison of methods in political science text classification: Transfer learning language models for politics. Working Paper.

Tian, Lexiao, Zheng, Dequan, & Zhu, Conghui (2013). Image classification based on the combination of text features and visual features. *International Journal of Intelligent Systems*, *28*(3), 242–256.

Vaswani, Ashish, Shazeer, Noam, Parmar, Niki, Uszkoreit, Jakob, Jones, Llion, Gomez, Aidan N., Kaiser, Lukasz, & Polosukhin, Illia (2017). Attention is all you need.

Vig, Jesse (2019). A multiscale visualization of attention in the transformer model. *arXiv preprint arXiv:1906.05714*.

Wang, Siruo, McCormick, Tyler H., & Leek, Jeffrey T. (2020). Methods for correcting inference based on outcomes predicted by machine learning. *Proceedings of the National Academy of Sciences*, *117*(48), 30266–30275.

Wang, Weiyao, Tran, Du, & Feiszli, Matt (2019). What makes training multi-modal classification networks hard?

Webb Williams, Nora, Casas, Andreu, & Wilkerson, John D. (2020). *Images as Data for Social Science Research: An Introduction to Convolutional Neural Nets for Image Classification*. Elements in Quantitative and Computational Methods for the Social Sciences. Cambridge University Press.

Wolf, Thomas, Debut, Lysandre, Sanh, Victor, Chaumond, Julien, Delangue, Clement, Moi, Anthony, Cistac, Pierric, Rault, Tim, Louf, Rémi, Funtowicz, Morgan, & Brew, Jamie (2019). Huggingface's transformers: State-of-the-art natural language processing.

Wu, Yonghui, Schuster, Mike, Chen, Zhifeng, Le, Quoc V., Norouzi, Mohammad, Macherey, Wolfgang, Krikun, Maxim, Cao, Yuan, Gao, Qin, Macherey, Klaus, Klingner, Jeff, Shah, Apurva, Johnson, Melvin, Liu, Xiaobing, Łukasz Kaiser, Gouws, Stephan, Kato, Yoshikiyo, Kudo, Taku, Kazawa, Hideto, Stevens, Keith, Kurian, George, Patil, Nishant, Wang, Wei, Young, Cliff, Smith, Jason, Riesa, Jason, Rudnick, Alex, Vinyals, Oriol, Corrado, Greg, Hughes, Macduff, & Dean, Jeffrey (2016). Google's neural machine translation system: Bridging the gap between human and machine translation.

Xu, Kelvin, Ba, Jimmy, Kiros, Ryan, Cho, Kyunghyun, Courville, Aaron, Salakhutdinov, Ruslan, Zemel, Richard, & Bengio, Yoshua (2015). Show, attend and tell: Neural image caption generation with visual attention.

Zahavy, Tom, Magnani, Alessandro, Krishnan, Abhinandan, & Mannor, Shie (2016). Is a picture worth a thousand words? a deep multi-modal fusion architecture for product classification in e-commerce.

Zhang, Han & Pan, Jennifer (2019). CASM: A deep-learning approach for identifying collective action events with text and image data from social media. *Sociological Methodology*, *49*(1), 1–57.

3.8 Supplemental Information

3.8.1 Feedforward Neural Networks

Define a single-layer feedforward neural network as $s = XW_1$, where X is the design matrix of dimension $N \times d$ and W_1 is $d \times C$, where N is the total number of observations, d is the total number of input features, and C is either 1 when using neural networks for regression problems or C is the total number of classes that an observation can belong to. Including a column of 1's in the design matrix adds a bias term. This produces a score matrix s of dimension $N \times C$. In a classification problem, each row contains the raw scores of an observation belonging to each class c. The argmax of each row is the predicted class.

We can plausibly give this network more representational power by using a second weight function, W_2, of dimension $k \times C$, and redefining W_1 as a $d \times k$ matrix, where k is a dimension that we can choose and is often referred to as the number of hidden neurons. This is known as a two-layer neural network. Notice that we cannot simply define the two-layer neural network as $s = (XW_1)W_2$, because this simply collapses back into a single-layer neural network: $s = (XW_1)W_2 = XW$ where $W = W_1W_2$ and has dimensions $d \times C$. Instead, we introduce a nonlinear function, f, called the activation function. This nonlinear function is applied pointwise. We can then properly define the two-layer neural network as $s = f(XW_1)W_2$. Popular options for f include the inverse tan function, the sigmoid function, $\sigma(x) = \frac{1}{1+e^{-x}}$, and the ReLU function, $ReLU(x) = max(0, x)$. Because of its simplicity and its empirically-determined robustness, the ReLU function is the most popular activation function.

A three-layer neural network would be similarly defined: redefine W_1 as a $d \times k_1$ matrix, redefine W_2 as a $k_1 \times k_2$ matrix, and define W_3 as a $k_2 \times C$ matrix; here, k_1 and k_2 are the number of hidden neurons. Then, $s = f(f(XW_1)W_2)W_3$. s is ultimately still an $N \times C$ matrix, with prediction carried out in the same way as the single-layer neural network.

The ultimate goal of the model is to have weights that minimize prediction errors. To do this, we need a measure of how well the model predicts and a way to update the weights of the network such that it reduces prediction errors. A loss function quantifies how well the model performs by comparing predictions with ground truth data. Popular loss functions include the cross-entropy loss function for classification problems and the mean squared error for regression problems. We can then use backpropagation (Rumelhart et al., 1986) to calculate the gradient of the loss function: we calculate the gradient of the loss function with respect to each weight one layer at a time, iterating from the last layer to the first layer. Weights are then updated using a gradient descent step.

3.8.2 Residual Connections

Theoretically, a deeper neural network should be able to perform just as well as a shallower neural network. A deeper network can imitate a shallower network by copying over the layers of the shallower network and setting the additional layers to the identity mapping. But He et al. (2015) observe that deeper networks often perform worse than shallower networks. They argue that as a neural network increases in layers it is harder for information from shallower layers to propagate to deeper layers. To solve what they call the degradation problem, they propose a very simple solution: after a set of layers $F(\mathbf{x})$, we sum the initial input $\mathbf{x}$ and the learned mapping $F(\mathbf{x})$. In other words, this set of layers outputs $F(\mathbf{x}) + \mathbf{x}$ instead of $H(\mathbf{x})$. This allows information from shallower layers to propagate forward more easily. It is called a residual connection because the layers learn the residual, or information not immediately learned from $\mathbf{x}$ directly. Although simple in concept, residual connections allowed much deeper networks to be trained.

3.8.3 Layer Normalization

Gradients with respect to weights in one layer are dependent on the outputs of the previous layer. This can cause the distribution of the parameters of one layer to shift in a way that affects the quality of learning for deeper layers. A normalization procedure, such as layer normalization, can reduce covariate shift (Ba et al., 2016). Layer normalization normalizes the inputs across the features. We can recalculate the inputs at each layer as follows:

$$\mu_i = \frac{1}{K}\sum_{k=1}^{K} x_{ik}$$

$$\sigma_i^2 = \frac{1}{K}\sum_{k=1}^{K}(x_{ik} - \mu_i)^2$$

$$\hat{x_{ik}} = \frac{x_{ik} - \mu_i}{\sqrt{\sigma_i^2 + \varepsilon}}$$

where x_{ik} is the kth feature of the ith sample and K is the total number of features. The last step is to scale and shift $\hat{x}_i$ by γ and β, respectively, which are learnable parameters: $\mathrm{LN}_{\gamma,\beta}(x_i) = \gamma\hat{x}_i + \beta$.

3.8.4 Additional Details About BERT

There are two variants of BERT. $\text{BERT}_{\text{base}}$ is 12 transformer encoder blocks stacked, while $\text{BERT}_{\text{large}}$ is 24 transformer encoder blocks stacked. The former uses embeddings with 768 dimensions, while the latter uses embeddings with 1,024 dimensions. $\text{BERT}_{\text{base}}$ is 110 million parameters, while $\text{BERT}_{\text{large}}$ is 340 million parameters. BERT takes as input a sequence of WordPiece embeddings (Wu et al., 2016).

In MLM, 15% of words are masked out and BERT must predict what the masked out word is. In NSP, pairs of sentences are inputted. Half of these pairs' second sentence is the sentence that follows the first sentence in the original corpus; the other half contains sentences randomly paired together. BERT must predict if the second sentence actually follows the first sentence in the original corpus. This prediction is made using the corresponding output vector for the `[CLS]` token.

Popular word embedding methods such as word2vec or GloVe embeddings (Pennington et al., 2014) are context-free, meaning that each word is assigned a fixed embedding irrespective of its context. For example, the word2vec embedding for the word "trump" would be the same in the sentences "I support Donald Trump" and "She played the trump card," despite the fact that "trump" has different meanings in each sentence. In other words, word2vec or GloVe word embeddings cannot map multiple vectors for polysemous words in a self-supervised fashion. BERT embeddings, on the other hand, are context-dependent: the BERT embedding for a given word is different for each context. The self-attention mechanism within the transformer encoder outputs a word embedding that, in essence, is a weighted average of all the other word embeddings of words in the document. The word embedding for "trump" in each sentence would be different.

3.8.5 Definition of Evaluation Metrics

3.8.5.1 Accuracy, Precision, Recall, and F1 Score

In binary classification problems, predictions are either for the positive or negative class. We can evaluate our predictions against the ground truth using the terms true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). False positives are known as Type I errors, while false negatives are known as Type II errors. With the predictions sorted into one of these four categories, we can calculate accuracy, precision, recall, and the F1 score. The metric most important to consider varies from problem to problem, although the F1 score is often used as an overall metric of performance. Precision, recall, and F1 are defined individually over each of the two categories. Macro and micro F1 are defined by combining the F1 metrics across the two classes. Accuracy is defined across the two classes.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

$$\text{Precision}_0 = \frac{TN}{TN + FN}$$

$$\text{Precision}_1 = \frac{TP}{TP + FP}$$

$$\text{Recall}_0 = \frac{TN}{TN + FP}$$

$$\text{Recall}_1 = \frac{TP}{TP + FN}$$

$$\text{F1}_0 = 2 \cdot \frac{\text{Precision}_0 \text{Recall}_0}{\text{Precision}_0 + \text{Recall}_0}$$

$$\text{F1}_1 = 2 \cdot \frac{\text{Precision}_1 \text{Recall}_1}{\text{Precision}_1 + \text{Recall}_1}$$

Denoting $N = N_0 + N_1$, where N_0 is the number of observations that belong to the negative class and N_1 is the number of observations that belong to the positive class,

$$\text{Macro F1} = \frac{\text{F1}_0 + \text{F1}_1}{2}$$

$$\text{Micro F1} = \frac{N_0}{N}\text{F1}_0 + \frac{N_1}{N}\text{F1}_1$$

3.8.5.2 Area Under the Receiver Operating Characteristic Curve (AUC)

The receiver operating characteristic curve (ROC curve) plots the true positive rate (or recall) against the false positive rate at various threshold settings. The true positive rate, or TPR, is defined as $\frac{TP}{TP+FN}$ while the false positive rate, or FPR, is defined as $\frac{FP}{FN+TP}$. When we make a classification, notice that the model does not simply return a 1 or 0; instead, it returns a probability that an observation belongs to the positive class. For example, we usually classify any inputted observation that returns a probability greater than or equal to 0.5 as the positive class. But this threshold is arbitrary. If we increase the threshold, the true positive rate would drop but so would the false positive rate. If we decreased the threshold, the true positive rate would rise but so would the false positive rate. See Figure 3.14 for an example of an ROC curve. An observation under the diagonal line signals worse-

than-random; an observation on the line signals random guessing; an observation at (0,1) indicates a perfect classifier—it has a perfect TPR and the FPR is 0.

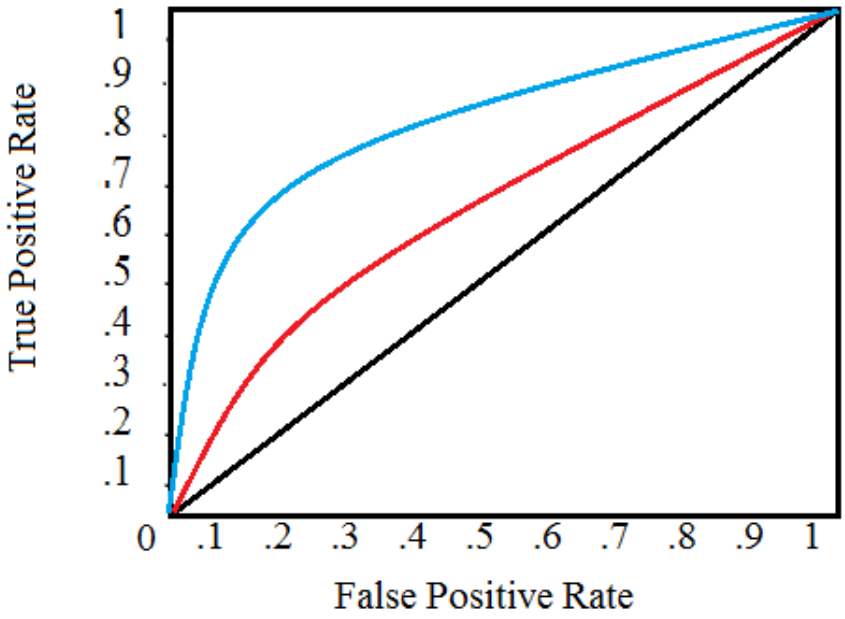


Figure 3.14: Example of an ROC curve. Figure taken from https://www.statisticshowto.com/receiver-operating-characteristic-roc-curve/.

A way of assessing the ROC curve is to measure the area under the curve, or AUC. A perfect AUC would be 1, corresponding to the 90 degree curve that consists of a line segment from (0,0) to (0,1) and a line segment from (0,1) to (1,1), indicating a perfect classifier. More interestingly, the AUC can be shown to be equivalent to the Mann-Whitney U statistic (Hastie et al., 2000). The AUC is equal to the probability that a classifier will rank a randomly chosen positive instance higher than a randomly chosen negative instance. Thus, the AUC can be interpreted as how well the classifier discriminates between the two classes. For this reason, it is often the preferred metric over other metrics, like accuracy or F1. Its major drawback is interpretability.

3.8.6 Additional Details about VirTex

Desai & Johnson (2021) propose pretraining a deep convolutional neural network (ConvNet) using images and image captions. This differs from the typical approach of pretraining ConvNets, which typically uses a large, labeled image dataset such as ImageNet (Deng et al., 2009). Their goal is to learn high quality image representations while using much fewer images. To do this, they jointly train a ConvNet and a transformer (Vaswani et al.,

2017) using image-caption pairs. They use ResNet-50 as the ConvNet. The visual backbone extracts image features, and then the transformer predicts the caption. The model is then trained end-to-end from scratch. After this pretraining process, the ConvNet can be used for downstream visual recognition tasks. See Figure 3.15 for an overview of the pretraining setup.

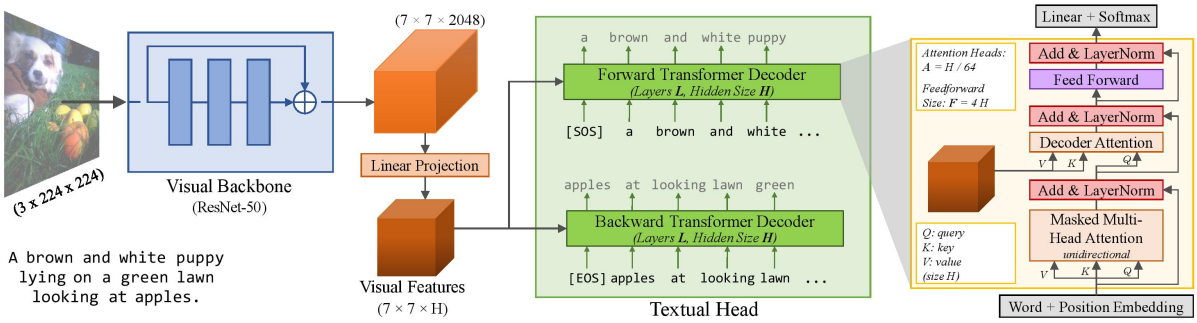


Figure 3.15: The VirTex pretraining setup, as described in Desai & Johnson (2021). This figure is taken directly from Desai & Johnson (2021).

3.8.7 Additional Details about Self-Critical Sequence Training

We use self-critical sequence training (Rennie et al., 2016) to calculate the image captions. Deep generative models typically use a technique known as teacher-forcing, which maximizes the likelihood of the next ground-truth word given the previous ground-truth words. This creates a mismatch between training and testing—during testing, the previous generated words are used instead of the previous ground-truth labels. Rennie et al. (2016) approach the image captioning problem with a reinforcement learning framework. The recurrent models, the LSTMs, are the "agents" that interacts with an external "environment" consisting of words and image features. The parameters of the network define a policy p_θ. When the end-of-sentence token is generated, or EOS, the agent is given a "reward" that is computed by evaluating generated caption with the ground-truth caption using some kind of a metric, such as CIDEr. This model is pretrained on Microsoft COCO (Lin et al., 2014). We do not further train this model; instead, we simply put the pretrained model in inference mode and generated a caption (or multiple captions) per image.

3.8.8 Details on Training MARMOT

To train MARMOT, the following hyperparameter choices need to be made:

- Using either $\text{BERT}_{\text{base}}$ or $\text{BERT}_{\text{large}}$; refer to the Section 3.8.4 in the Supplemental Information for more information about $\text{BERT}_{\text{base}}$ and $\text{BERT}_{\text{large}}$. The primary difference is the number of encoder layers used to pretrain each model.
- The number of training epochs
- The learning rate
- The learning rate schedule
- The batch size
- The number of captions to use per image
- Whether to freeze certain parts of the architecture
- The details of the fully-connected classifier

In our applications, we use $\text{BERT}_{\text{base}}$, primarily because of computational constraints. We select the training epochs from $\{3, 4\}$, the learning rate from $\{2 \times 10^{-5}, 3 \times 10^{-5}, 5 \times 10^{-5}\}$, and the batch size from $\{16, 32\}$. The limited ranges of hyperparameters make grid searches feasible. These are the same values suggested in Devlin et al. (2018). We use the cosine learning rate schedule (Huang et al., 2017). We find that using 3 image captions generally work well, with more captions typically worsening performance. Using the weighted Adam optimizer (Kingma & Ba, 2014), as suggested in Devlin et al. (2018), generally works well.

Whether we freeze certain parts of the architecture depends on the application and grid search feasibility. Wang et al. (2019) notes that part of the difficulty of training multimodal models is that the image and text features are learned by the model at different rates. One strategy to alleviate this issue is to freeze and unfreeze certain parts of the model at different stages of training (Kiela et al., 2019). When a part of the model is frozen, the parameters are not updated after the backwards pass. In the first few iterations, both the transformer decoder of modality translation and the pretrained language model are frozen, allowing the model to only have the ability to update the weights of the 1×1 convolution over the image features and the pretrained image network. Then, the transformer decoder of modality translation is unfrozen but the pretrained language model remains frozen. Lastly, the pretrained model is unfrozen and the model is trained end-to-end.

Finetuning is also quite sensitive to the learning rate schedule, which is how the learning rate changes as training progresses. We design a three-stage learning rate scheduler when freezing certain parts of the architecture. The first 10% of the total training iterations are dedicated to a warmup period, where the learning rate rises from 0 to the initial set learning rate in a linear fashion. Then for the epochs where the transformer decoder of

modality translation or pretrained language model are frozen the learning rate is fixed at the initial set learning rate. When finetuning of the entire model occurs after the pretrained language model is unfrozen, the learning rate decreases following the values of a cosine function between the initial set learning rate to zero. Freezing the transformer decoder of modality translation for 2 epochs and the pretrained language model for 4 epochs generally worked well, and experimentation showed that more epochs where either part were frozen did not yield improvements in performance.

The model is implemented in Python using PyTorch and HuggingFace's `transformers` library (Wolf et al., 2019).

3.8.9 Application 1: Definitions of Subcategories

The sub-bullet points indicate the subcategories that belong to a given category. More detailed definitions for each category can be found in Mebane et al. (2018).

- Line length, waiting time, polling place overcrowding
  - There was no crowd or line at the polling place
  - There was a small crowd, short line, or wait
  - The polling place was crowded or there was a long line or wait (20 minutes or longer)
- Polling place event
  - The polling place did not function as expected or information is incorrect
  - The tweet describes the polling place without noting whether it or an aspect functioned correctly or incorrectly
  - The polling place did function correctly or information is correct
- Electoral system
  - The electoral system did not function appropriately
  - The tweet makes a neutral statement about the electoral system without an indication of if it functioned appropriately
- Absentee, mail-in, or provisional ballot issue
  - The absentee, mail-in, or provisional ballot system did not function appropriately
  - The tweet makes a neutral observation or statement about the absentee, mail-in, or provisional ballot system without noting it having functioned correctly or incorrectly

  - The absentee, mail-in, or provisional ballot system functioned properly

- Registration
  - The tweet indicates that an individual was not able to register to vote
  - The tweet makes a neutral observation about the voter registration process without noting if the individual in question registered or not
  - The tweet notes that the individual was able to vote

3.8.10 Application 1: Hyperparameters

To create a validation dataset, we randomly held out 20% of the training set for each category and subcategory. We used a grid search to find the learning rate and the number of training epochs that optimized the F1 score over the class. We did not grid search over the batch size because of computational constaints. Table 3.5 details the hyperparameters for the categories and subcategories.

Category	**Batch Size**	**Learning Rate**	**Epochs**
Not an Incident	16	5×10^{-5}	4
Line Length	16	2×10^{-5}	4
(a) No crowd	16	3×10^{-5}	4
(b) Small crowd	16	3×10^{-5}	4
(c) Large crowd	16	5×10^{-5}	4
Polling Place Event	16	3×10^{-5}	4
(a) Did not function as expected	16	2×10^{-5}	4
(b) Neutral observation	16	5×10^{-5}	4
(c) Functioned properly	16	2×10^{-5}	3
Electoral System	16	3×10^{-5}	4
(a) Did not function properly	16	3×10^{-5}	3
(b) No comment on function	16	2×10^{-5}	4
Absentee or Early Voting Issue	16	5×10^{-5}	4
(a) Did not function properly	16	5×10^{-5}	4
(b) Neutral observation	16	5×10^{-5}	4
(c) Functioned properly	16	5×10^{-5}	4
Registration	16	2×10^{-5}	4
(a) Not able to register	16	2×10^{-5}	4
(b) Neutral observation	16	5×10^{-5}	4
(c) Able to register	16	3×10^{-5}	4

Table 3.5: The selected hyperparameters for each category and subcategory for the tweets about election incidents during the 2016 U.S. general election.

We used $\text{BERT}_{\text{base}}$. We did not freeze any parts of MARMOT. We used weighted Adam (Kingma & Ba, 2014) with a cosine learning schedule. We did not use gradient clipping.

The ε, β_1, and β_2 for Adam are set to 1×10^{-8}, 0.9, and 0.98 respectively, following the parameters set by Kiela et al. (2020).

3.8.11 Application 2: Hyperparameters

Hyperparameters were selected using the dev set provided by the Hateful Memes dataset (Kiela et al., 2020). We used a grid search to find the batch size, learning rate, and number of training epochs that optimized area under the receiver operating characteristic curve (AUC) over the dev set. The hyperparameters selected were: batch size of 32, learning rate of 5×10^{-5}, and 8 training epochs. We used $\text{BERT}_{\text{base}}$.

The parameters of both the transformer decoder of modality translation and the pretrained language model were frozen for 2 epochs, and the parameters of just the pretrained language model were frozen for 2 more epochs after. After the 4th epoch, the entire model is finetuned end to end. We used weighted Adam (Kingma & Ba, 2014) with the learning rate schedule detailed in Section 3.8.8 in the Supplemental Information. We did not use gradient clipping. The ε, β_1, and β_2 for Adam are set to 1×10^{-8}, 0.9, and 0.98 respectively, following the parameters set by Kiela et al. (2020).

3.8.12 Application 2: Accuracy Learning Curve During Training over the Hateful Memes Dataset

To assess whether MARMOT overfits the training data, we plot the accuracy learning curve of the validation set during training. Figure 3.16 shows that the curve monotonically increases during training. Because the parameters of both the transformer decoder of modality translation and the pretrained language model were frozen for 2 epochs and the parameters of just the pretrained language model was frozen for 2 more epochs after, the accuracy does not increase in the first four epochs of training.

3.8.13 Application 2: Results Over the Validation Set of the Hateful Memes Dataset

Table 3.6 contains the accuracy and area under the receiver operating characteristic curve (AUC) performance metrics across the 11 baseline models and MARMOT over the validation dataset. Results for the 11 baseline models come from Kiela et al. (2020). It is important to note that hyperparameters were optimized using the validation dataset. The results are largely in line with the results over the test set.

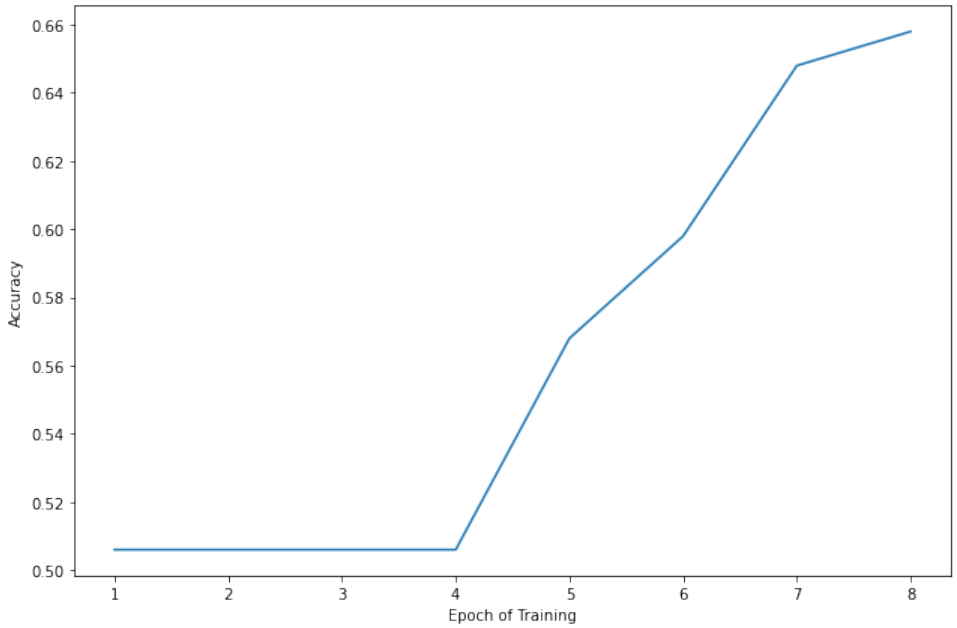


Figure 3.16: Accuracy learning curve of the validation set during training over the Hateful Memes dataset.

Model	**Accuracy**	**AUC**
Image - Grid	0.5273	0.5879
Image - Region	0.5266	0.5798
Text BERT	0.5826	0.6465
Late Fusion	0.6153	0.6597
Concat BERT	0.5860	0.6525
MMBT - Grid	0.5820	0.6857
MMBT - Region	0.5873	0.7173
ViLBERT	0.6220	0.7113
VisualBERT	0.6210	0.7060
ViLBERT CC	0.6140	0.7060
VisualBERT COCO	0.6506	0.7397
MARMOT	0.6580	0.7587

Table 3.6: Accuracy and area under the receiver operating characteristic curve (AUC) performance metrics across the 11 baseline models and MARMOT over the validation set of the Hateful Memes dataset.

CHAPTER 4

Studying Language Usage Evolution Using Pretrained and Non-Pretrained Embeddings

Abstract

Changing usage of language in politics indicate shifts in how specific issues are discussed and what is politically salient at a given time. Natural language processing researchers have used distributed word embeddings to study the evolution of particular words over time (see, e.g., Hamilton et al., 2016b). Typically, separate word embedding spaces are trained using the text from each period of interest. Distributed word embedding approaches offer an advantage over topic models because the researcher can examine how usages of specific words evolve. However, the corpora that political scientists usually work with are much smaller than the extensive corpora used in natural language processing research. Splitting up the corpus into even smaller corpora leads to poorly trained embeddings. This paper proposes a framework, based on the theory developed in Arora et al. (2018), that uses both pretrained and non-pretrained embeddings to learn time-specific word embeddings. I call this the pretrained-augmented embeddings (PAE) framework. In the first application, I apply the PAE framework to a corpus of *New York Times* text data spanning several decades. The PAE framework matches human judgments of how specific words evolve in their usage more closely than existing methods. In the second application, I apply the PAE framework to a corpus of tweets published during the COVID-19 pandemic about masking. I show that the PAE framework automatically detects discussions about specific events during the COVID-19 pandemic vis-à-vis the keyword of interest.

Author's Note

This work was supported in part by a fellowship from the Michigan Institute for Computational Discovery & Engineering (MICDE).

4.1 Introduction

Studying how language is used is a central part of analyzing politics. For example, political scientists have studied how the language used to frame certain issues shapes public opinion (e.g., Hopkins, 2018; Aslett et al., 2020). Others have looked at how the language used by the political non-elite shapes the language used by political elites (e.g., Barberá et al., 2019). Scholars have also looked at how language usage shifts over time. These changes are of interest because they reflect shifts in what political issues are considered important. For example, scholars have also studied how the language in policies has changed over time (e.g., Park et al., 2020; Bagozzi & Berliner, 2018). Works on topic models have also looked at how political attention on certain topics changes over time (e.g., Quinn et al., 2010). Political methodologists have also developed frameworks using automated text-as-data methods, such as word embeddings, to study how words change over time and how words are used differently between partisan or interest groups (Li et al., 2017; Rodman, 2020; Rodriguez et al., 2021).

This paper proposes a method of studying how a word's or words' usages evolve over time for smaller corpora and rare words or phrases. The method learns time-sensitive embeddings that use both pretrained and non-pretrained components. It builds off the theoretical framework proposed in Arora et al. (2018) and the application of this framework detailed in Khodak et al. (2018). The proposed framework first learns the relationship between words and their contexts through a linear transformation using word embeddings. It then applies this linear transformation to all of a word of interest's contexts in a specific time period to learn a time-specific embedding. However, with smaller corpora, the context vectors, which I learn using doc2vec, and the word embeddings, also learned through doc2vec using skip-gram, may be poorly fit (Le & Mikolov, 2014). Pretrained embeddings, which are embeddings trained on large, general text datasets such as Google News or Wikipedia, are typically much higher quality embeddings (Mikolov et al., 2013). However, they often do not capture nuances of highly specialized corpora or corpora using informal language, such as tweets.

The pretrained-augmented embeddings (PAE) framework solves this issue using both pretrained embeddings and non-pretrained embeddings. To expand context vectors, I concatenate doc2vec document embeddings with BERT contextual embeddings (Devlin et al., 2019). To expand the word embeddings, I calculate a weighted sum of the pretrained and non-pretrained embeddings. This weighted sum works such that the most frequently occurring words resemble their corresponding non-pretrained embedding, and less frequently occurring words resemble their corresponding pretrained embedding. I then use the pretrained-augmented embeddings to learn the linear transformation. Lastly, I calculate the time-

sensitive embedding for a word of interest using the pretrained-augmented context vectors of the word of interest in a specific time period and multiplying this by the learned linear transformation matrix.

This paper makes three methodological contributions. First, this paper is the first to use document embeddings to calculate time-specific embeddings. Second, this paper is one of the first papers to meaningfully combine static word embeddings and contextual word embeddings derived from pretrained language models such as BERT. To the best of my knowledge, only one other paper, Alghanmi et al. (2020), has combined word2vec embeddings with BERT embeddings. Third, it builds on literature that studies how political scientists can meaningfully apply an automated text-as-data method to smaller corpora. It can also augment existing methods that study language usage, such as methods that automatically identify issue framing (e.g., Sagi et al., 2013; Tsur et al., 2015), to additionally study how language usage changes over time. It requires no human coding to use.

The paper proceeds as follows. I first review the literature on lexical semantic change and language evolution, including meaning change versus word usage change. I then discuss in detail the proposed approach that uses pretrained and non-pretrained embeddings. I then apply the approach to Rodman (2020), a paper that studies the relationship between the word "equality" and five major social categories in *New York Times* articles across time. The primary challenge of this dataset is its small corpus size. There are only $3,105$ articles. For each article, the dataset provides only the headline, an abstract, and the first paragraph. Each time period can have as little as 80 of these observations. The PAE framework improves the best correlation between the human-verified associations and the machine-calculated associations from 0.611 to 0.864. I then apply the approach to tweets about masking during the COVID-19 pandemic between January 2020 and July 2020. The PAE framework detects shifts in the discussion surrounding masking, moving from discussions about China, to discussions about masking as a way to prevent the spread of the coronavirus, to discussions about the efficacy and the merits of masking.

4.2 Studying Language Evolution

This section briefly reviews the previous literature on modeling meaning and word usage. I first provide a non-technical review of the literature on word embeddings and pretrained language models and how they have facilitated the growth in research on computational lexical semantic change. I then discuss the difference between meaning change versus word usage change and argue that almost all computational methods have actually studied the latter rather than the former. However, under the distributional semantics hypothesis, both

are functionally equivalent.

4.2.1 Computationally Modeling Semantics and Word Usage

The popular computational models for meaning fall into three categories: topic models, distributed word embeddings (known as *static embeddings*), and pretrained language models (known as *contextual embeddings*). Topic models split words into semantic areas that roughly approximate senses. A static embedding is a single vector representation for a word and all its semantic information. A contextual embedding is a representation that changes with every instance of a given word. Some works have used topic models to study lexical semantic change, but most works in lexical semantic change have concentrated on the second category. A growing literature uses contextual embeddings to study lexical semantic change; however, most of these methods are post hoc clustering methods that contend that clusters represent a word's usage in similar ways.

4.2.1.1 Topic Models

Topic models are a set of general statistical models that aim to discover latent topics of a corpus. The most well-known topic model is latent Dirichlet allocation, which treats each document as a random mixture over latent topics and each topic as a probability distribution over tokens (Blei et al., 2003). Topic models generally generate groups of keywords; researchers then assign a topic based on the keywords present. For example, a set containing the keywords "healthcare," "mandate," "premiums," and "deductible" may indicate a health insurance topic. For a more technical introduction to topic models, see Blei (2012).

Several topic models explicitly study the evolution of topics over time; most are a modification of LDA. Blei & Lafferty (2006), Wang & McCallum (2006), and Wang et al. (2008) propose LDA-style topic models that associate each topic with a probability distribution over timestamps. Generally, applications of their proposed methods are over long texts such as over the abstracts of *Science* journal articles (Blei & Lafferty, 2006). Quinn et al. (2010) develop a topic model for legislative speeches where every document is assigned only one topic. They use this to study how political attention to certain topics shift over time. Roberts et al. (2013) propose Structural Topic Models, a topic model that incorporates document-level covariates. They argue that one of these covariates can be time and look at news wires from China. They find a spike in the "Taiwan" topic during the 2000 and 2004 presidential elections in Taiwan.

However, these time-sensitive topic models struggle to study how a specific word's usage changes over time. In other words, the topic models do not deal with semantics and studying

how individual words relate to one another beyond topic assignment. It is plausible to argue that these models study word usage change over time if every document in a corpus contains the given word. However, because the researcher has no control over the topics of interest, the topic model may focus on a topic that is not related to the word of interest. Because there is no control over what topics or words to look at, most works in computational lexical semantic change have not used topic models.

4.2.1.2 Distributed Word Embeddings (Static Embeddings)

Most works in lexical semantic change have focused on using distributed word embeddings. Distributed word embedding methods refer to a set of computational techniques that map words to vectors. The underlying assumption of these methods is the distributional semantics hypothesis, which posits that a word's meaning comes from how it is used (Firth, 1957; Wittgenstein, 2009). In other words, distributed word embeddings assume that the word's context gives it its meaning. For example, "cat" and "dog" are more similar in meaning than "dog" and "apple" because people use many similar words in the context of "cat" and "dog."

These embeddings are called "static" embeddings because every word is assigned to one word embedding, even if a word is used in different senses. The inability to deal with polysemy is one of static word embeddings' primary drawbacks. Several approaches have been developed to overcome this problem, with most solutions modifying the word embeddings after they are calculated (e.g., Mu et al., 2016; Sun et al., 2017; Arora et al., 2018).

The most popular distributed word embedding methods are word2vec (Mikolov et al., 2013) and GloVe (Pennington et al., 2014). The former produces word embeddings using a shallow neural network, and the latter produces word embeddings using matrix factorization techniques. Both produce embeddings that have similar properties. In particular, word2vec uses a shallow two-layer neural network to predict words around a given word (or uses a given word to predict its context words) (Mikolov et al., 2013). The weights of the neural network form the word vectors. The distance between word vectors, usually measured using cosine similarity,[1] indicates semantic similarity. The underlying idea is intuitive: if two words are similar in meaning, then they should have similar context words under the distributional hypothesis. If the neural network predicts similar context words for a pair of words, then their weights should be approximately the same.

Because word embeddings encode semantic similarity between words, they have been the most popular vehicle for studying lexical semantic change. These works focus on calculating

[1]Cosine similarity is defined, for vectors A and B, as $\cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|}$, where $\cos(\theta)$ is the cosine similarity.

time-specific word embeddings. Most works in this domain first fit a word embedding space over each period; then, the spaces are aligned using some kind of a transformation (see, e.g., Kim et al., 2014; Kulkarni et al., 2015; Hamilton et al., 2016b). Other methods explicitly incorporate time into learning the embeddings. For example, Rudolph & Blei (2018) learn representations that are a sequential random variable; they assume that context vectors are the same across time. Rosenfeld & Erk (2018) train word embeddings using both a word component (resembling the skip-gram negative sampling model of word2vec) and a continuous time component.

These approaches have yielded interesting substantive results. Hamilton et al. (2016a) use word embeddings and two novel computational measures to determine if word meaning changes are occurring because of cultural shifts or more regular processes of semantic change, such as grammaticalization or subjectification. Garg et al. (2018) find that changes in word embeddings correlate with demographic and occupation shifts over time.

The drawback with these approaches is that they tend to either require a large number of documents in the corpus or make strong assumptions, such as assuming that the context vectors remain the same throughout time. For example, Hamilton et al. (2016b) use the Corpus of Historical American English (COHA) as their smallest corpus. COHA contains more than 475 million words from texts written between the 1820s and the 2010s. Word embedding methods are data-hungry, and splitting up a corpus into corpora by time slices further worsens this issue with smaller corpora often found in the social sciences.

4.2.1.3 Pretrained Language Models (Contextual Embeddings)

Pretrained language models emerged as a solution for smaller corpus sizes. These models are pretrained on a set of general tasks. For example, bidirectional encoder representations from transformers, or BERT, are among the most popular pretrained language models. BERT calculates representations using the transformer. The transformer is a deep learning architecture that utilizes the attention mechanism, which, in the case of text data, upweights and downweights certain parts of the text (Vaswani et al., 2017). The model is pretrained on two tasks: the masked language model (MLM) and next sentence prediction (NSP). In the former task, the model must predict a masked out word using the surrounding words. In the latter task, the model receives as input two sentences and must predict whether the two sentences follow one another in the actual text (Devlin et al., 2019). BERT is pretrained using text from English Wikipedia and a large corpus of novels. BERT learns both embeddings for the document and the individual words. When used for classification or prediction, BERT is typically finetuned. For a more technical introduction to transformers and pretrained language models, see Bloem (2019), Rush (2018), and Wu & Mebane (2021).

Pretrained language models learn so-called "contextual" embeddings because the embeddings learned for each word change depending on the context. For example, the word embedding for "bank" would be different in a sentence referring to the financial institution compared to the word embedding for the same word in a sentence referring to the land along the side of a river. Static word embeddings would assign the same embedding for both usages of the word.

Because the word embedding changes based on its context, recent lexical semantic changes have aimed to utilize these pretrained language models. Most works have used BERT because of its popularity and ease of use through pre-existing packages (Wolf et al., 2020). It is not clear, however, how to compare BERT embeddings (Reimers & Gurevych, 2019). It is also not clear how to compare words with one another, given that there is a set of contextual embeddings for every word. Martinc et al. (2020) average across each word's set of contextual embeddings to create a word embedding for each unique word in the corpus. Obtaining averages for each word allows them to calculate cosine similarities between words. Their paper, however, does not compare this approach to word2vec.

Most works cluster BERT embeddings to study word meaning change. These papers generally apply some clustering approach to a word's set of contextual embeddings. They then use some kind of metric to quantify semantic change based on changing clusters over time (see, e.g., Giulianelli et al., 2020; Martinc et al., 2020). The assumption is that the contextual embeddings of a word that cluster together indicate similar word usages. However, clustering methods tend to be post hoc, and it is not clear how finetuning BERT changes results. It is also not clear how stable clusters are with different clustering methods.

4.2.2 Meaning Change vs. Word Usage Change

The most used computational lexical semantic change approaches rely on the distributional hypothesis (Hengchen et al., 2021). In other words, these works assume that a changing context for a given word indicates semantic shift. Changing contexts, however, do not reflect changes at all layers of meaning (Blank & Koch, 1999). For example, the phrase "President Trump" refers to the United States president from January 2017 to January 2021. However, how the news and social media discuss him changes dramatically in various contexts, from the Women's March to his various scandals to his response to the coronavirus pandemic. Although the contextual neighborhood around the phrase "President Trump" changes with time, the meaning associated with the word both stays constant on one layer (his occupation of a formal government position) and changes at the same time (how perception of his presidency shifts vis-á-vis his actions and current cultural, social, and political events).

Therefore, most works in computational lexical semantic change are explicitly studying how word usage shifts over time. The claim that the dominant meaning of a word changes usually comes from looking at a word's context across a massive corpus, such as COHA. If there appears to be a "systematic" shift in a word's context, the claim is that this indicates a change in a word's dominant meaning. What qualifies as "systematic," however, differs from work to work and is often unclear (Hengchen et al., 2021). Most works also do not differ between word meaning shifts due to cultural or political shifts versus word meaning shifts over time from regular processes of semantic change.

To avoid this issue of what "meaning" precisely means and how to best study "meaning" shifts, I instead focus on how a word's usage evolves over time without claiming that this is a definitive shift in a word's meaning or meanings. I am also less concerned with shifts over time from regular processes of semantic change and instead focus on how a word's usage evolves due to cultural, social, or political shifts. Throughout this paper, "meaning change" or "lexical semantic change" refers to meaning change in the distributional hypothesis sense, acknowledging that this may or may not capture any definitive change in the layers of meaning of the word of interest.

4.3 The Pretrained-Augmented Embeddings Framework

This section details the proposed approach. Algorithms 4.1 and 4.2 fully describe the proposed approach. The section will first review the theoretical background of the proposed framework. Then, I describe the proposed method.

4.3.1 Theoretical Background

The proposed approach builds on the theory and methods developed in Arora et al. (2018) and Khodak et al. (2018). Arora et al. (2018) argue that a polysemous word's corresponding word embedding is a weighted sum of its various senses.[2] Arora et al. (2018) assume that a corpus is generated using a Gaussian walk model. A discourse vector c is drawn from a Gaussian distribution with mean 0 and covariance Σ. A window of n words $w_1, w_2, ..., w_n$ is

[2]It is important to note that Arora et al. (2018)'s theoretical framework dealing with polysemous word embeddings is not the only theoretical framework dealing with this issue of static word embeddings. For an overview of various sense embedding methods, see Camacho-Collados & Pilehvar (2018).

generated from c by

$$P\left[w_1, w_2, ..., w_n|c\right] = \prod_{i=1}^{n} P\left[w_i|c\right]$$

$$P\left[w_i|c\right] = \exp\left(c \cdot v_{w_i}\right)/Z_c$$

where $Z_c = \sum_w \exp(c \cdot v_w)$ is the partition function. Assuming that the v_w are random vectors, the partition function can be approximated using $Z_c \approx Z \exp\left(\|c\|^2\right)$ for some constant Z.

Using this Gaussian random walk model, Arora et al. (2018) first show that there exists a linear relationship between the vector of a word and the vectors of the words in its contexts. Specifically, they argue that, for s denoting the random variable of a window of n words, there exists a linear transformation $\mathbf{A}$ such that

$$v_w \approx \mathbf{A}\mathbb{E}\left[\frac{1}{n}\sum_{w_i \in s} v_{w_i} | w \in s\right]$$

They use this theorem to prove that, under the random walk model, $\|v_w - \bar{v}_w\|_2 \to 0$ as the corpus length tends to infinity, where $\bar{v}_w \approx \alpha v_{s_1} + \beta v_{s_2}$ for $\alpha = \frac{f_1}{f_1+f_2}$ and $\beta = \frac{f_2}{f_1+f_2}$ where f_1 and f_2 are the numbers of occurrences of s_1 and s_2 in the corpus. In other words, the word embedding of word w is the weighted sum of the word embeddings of its various senses, where the weights are the frequency of usages of each sense.

Khodak et al. (2018) use this theoretical model to fit word embeddings for words that either infrequently appear or did not initially appear in the corpus the word embeddings were fit over. To calculate these word embeddings, they learn the linear transformation that best recovers existing word vectors $\mathbf{v}_w$ using the average context of a given word. Each context is the average of the word embeddings corresponding to the words in a given word's context. A fixed-size window determines a word's context. More precisely, the relationship between the existing word vectors $\mathbf{v}_w$ with its context vectors is as follows:

$$\mathbf{v}_w \approx \mathbf{A}\left(\frac{1}{|\mathcal{C}_w|}\sum_{c \in \mathcal{C}_w}\frac{1}{|c|}\sum_{w' \in c}\mathbf{v}_{w'}\right)$$

where $\mathcal{C}_w$ is the subcorpus of all documents containing the word w, and $|c|$ is the window size determining the word's context. Under this setup, the linear transformation $\mathbf{A}$ is learned using linear regression. After learning $\mathbf{A}$, word embeddings for out-of-vocabulary or rare words can be calculated within the semantic space of the already existing word embeddings using the linear transformation $\mathbf{A}$. Khodak et al. (2018) find that this approach yields state-of-the-art results on several baselines.

4.3.2 Proposed Approach

The proposed approach deals explicitly with the issue of time. For a word embedding space trained using a corpus with time range T, where $t \in T$ denotes a time period, we can imagine that a word embedding is a weighted sum of its meanings from each time period. This is a direct application of Arora et al. (2018)'s argument that a word embedding is a weighted sum of its various meanings. Then, assuming without loss of generality a corpus over two time periods, the word embedding for word w can be expressed as follows:

$$v_w = \alpha v_{w,t} + \beta v_{w,t+1}$$

We can use the method developed in Khodak et al. (2018) to estimate the time-specific word embeddings.

$$v_{w,t} = \mathbf{A}\left(\frac{1}{|\mathcal{C}_{w,t}|}\sum_{c\in\mathcal{C}_{w,t}}\frac{1}{|c|}\sum_{w'\in c}\mathbf{v}_{w',t}\right)$$

However, we cannot estimate this because this relies on time-specific word embeddings of the context words. Using the word embeddings trained using the entire corpus assumes that the word embeddings of contextual words remain constant through time. We could potentially get around this issue by training a word embedding space for each time period separately, but this may trim down an already small corpus into even smaller chunks. Such an approach would produce poor quality word embeddings for each time period.

To get around this issue of calculating the time-specific average context, I use the document embedding calculated using doc2vec (Le & Mikolov, 2014). Doc2vec is a simple extension of word2vec. Distributed bag-of-words version of paragraph vectors (PV-DBOW) uses the document identifier to predict all the words in that specific document; distributed memory version of paragraph vectors (PV-DM) selects a given word and uses the document identifier along with the given word's context to predict that word. In both versions, the document identifier is used to predict words in a specific document. Both approaches can also generate word embeddings in the same vector space. Although document vectors are not explicitly time-sensitive, they predict the words in a document from a specific time.

Because of the short length of documents and the relatively smaller corpus sizes, document embeddings may not be high quality embeddings. I turn to bidirectional encoder representations from transformers, or BERT, to generate pretrained document embeddings (Devlin et al., 2019). BERT is a language model pretrained using English Wikipedia and a large set of novels. Technical introductions to transformers can be found at Rush (2018) and Bloem (2019). I use BERT to generate a document embedding. The context vector for

one context of a given word is the concatenation of the context's doc2vec embedding with the context's BERT embedding.

I use both doc2vec and BERT embeddings because they complement each other. Doc2vec explicitly learns the semantics of a given document, which is what we are ultimately interested in. However, because of corpus size constraints, these document embeddings may not be very high quality. BERT, on the other hand, is capturing far more than semantic information. Many works in "BERTology" argue that BERT learns much more than only semantic information in the document (Rogers et al., 2020). Several works have shown that BERT's attention mechanism learns the semantic information, grammatical structure, and syntactical relationships across words in the document (Clark et al., 2019; Hewitt & Manning, 2019; Coenen et al., 2019; Tenney et al., 2019). However, the exact nature of the relationship between embeddings derived from distributional hypothesis-based approaches and embeddings from deep pretrained language models is still not fully understood. Alghanmi et al. (2020) have shown that combining the static and contextual embeddings improve text classification of social media posts. Further work is required to fully understand the relationship between distributional hypothesis-based document embeddings and document embeddings from deep pretrained language models.

This same issue applies to doc2vec's word embeddings. Words that infrequently appear most are likely to be low quality embeddings. Pretrained word embeddings exist, such as pretrained embeddings over the Google News dataset (Mikolov et al., 2013). However, these pretrained embeddings do not capture the unique senses of specific words in the corpus of interest. For example, corpora related to social media or political floor speeches often feature words used in non-ordinary senses. Instead of using only the non-pretrained and noisy embeddings or only the pretrained and much less noisy embeddings, I use a convex combination of the two. The number of documents that the word appears in determines the weights. This regularization step shrinks the word vector $\mathbf{v}_w$ towards its pretrained vector. A word that appears more frequently will have a word embedding that resembles the non-pretrained word embedding, while a word that appears infrequently will have a word embedding that resembles the pretrained word embedding. Algorithm 4.1 details this process on line 10.

Algorithms 4.1 and 4.2 detail the pretrained-augmented embeddings (PAE) framework. Algorithm 4.1 details how I use the average doc2vec/BERT contexts and the pretrained/non-pretrained word embeddings to estimate the linear transformation matrix $\mathbf{A}$. It requires a pretrained word embedding space $\mathbf{V}_{\text{pretrained}}$, such as the pretrained word2vec embedding space using Google News (Mikolov et al., 2013). Algorithm 4.2 details how I use this learned linear transformation matrix to estimate the time-specific embeddings. The algorithms make

three contributions. First, it proposes using document vectors to avoid assuming that the word vectors of context words remain the same throughout all time periods. Second, it proposes using both pretrained and non-pretrained components in both the contextual embeddings and the word embeddings, improving the quality of representations when documents are short or when the corpus size can be quite small. Third, this framework works well with small corpora.

Algorithm 4.1: Obtain the linear transformation matrix $\mathbf{A}$

Data: vocabulary $\mathcal{V}$, corpus $\mathcal{C}_{\mathcal{V}}$, pretrained word embedding space $\mathbf{V}_{\text{pretrained}}$
Result: linear transformation matrix $\mathbf{A}$

Use doc2vec with skipgram to generate a joint document-word embedding vector space with dimension d using $\mathcal{C}_{\mathcal{V}}$ and $\mathcal{V}$. Denote the set of document vectors as $\mathbf{D}_{d2v}$ and the set of word embeddings as $\mathbf{V}_{d2v}$
Use BERT to generate document vector space $\mathbf{D}_{BERT}$ using $\mathcal{C}_{\mathcal{V}}$
$F_{\mathcal{V}} \leftarrow [\,]$
for $w \in \mathcal{V}$ **do**
    Calculate the number of documents w occurs in, f_w
    $F_{\mathcal{V}}$.append(f_w)
end
for $w \in \mathcal{V}$ **do**
    $\mathbf{v}_w \leftarrow \frac{\log(f_w)}{\log(\max(F_{\mathcal{V}}))}\mathbf{v}_{w,d2v} + \left(1 - \frac{\log(f_w)}{\log(\max(F_{\mathcal{V}}))}\right)\mathbf{v}_{w,pretrained}$
    let $\mathcal{C}_w \subseteq \mathcal{C}_{\mathcal{V}}$ be the subcorpus of documents containing at least one instance of w
    $\mathbf{u}_w \leftarrow \frac{1}{f_w}\sum_{c \in \mathcal{C}_w} \text{concat}\,(\mathbf{d}_{c,d2v}, \mathbf{d}_{c,BERT})$
end
$\mathbf{A} \leftarrow \arg\min_{\mathbf{A}} \sum_{w \in \mathcal{V}} \|\mathbf{v}_w - \mathbf{A}\mathbf{u}_w\|_2^2$

Algorithm 4.2: Obtain time-specific word embeddings for word w

Data: word w, corpus of documents from time range T containing word w $\mathcal{C}_w$, F_T, number of documents word w occurs in by time f_T, $\mathbf{D}_{d2v}$, $\mathbf{D}_{BERT}$, $\mathbf{A}$
Result: time-specific word embeddings for word w for each time $t \in T$

$\mathbf{V}_T \leftarrow [\,]$
for $t \in T$ **do**
    let $\mathcal{C}_{w,t} \subseteq \mathcal{C}_w$ be the subcorpus of documents from time t
    $\mathbf{u}_{w,t} \leftarrow \frac{1}{f_{w,t}}\sum_{c \in \mathcal{C}_{w,t}} \text{concat}\,(\mathbf{d}_{c,d2v}, \mathbf{d}_{c,BERT})$
    $\hat{\mathbf{v}}_{w,t} \leftarrow \mathbf{A}\mathbf{u}_{\mathbf{w,t}}$
    $\mathbf{V}_T$.append($\hat{\mathbf{v}}_{w,t}$)
end

4.3.3 Practical Considerations

In practice, some minor adjustments can be made to the above algorithms. During the regression step, one may weigh each point by some non-decreasing function α based on the word's corpus count; this is the same modification made in Khodak et al. (2018). Specifically,

$$\mathbf{A} = \arg\min_{\mathbf{A}} \sum_{w \in \mathcal{V}} \alpha(f_w) \|\mathbf{v}_w - \mathbf{A}\mathbf{u}_w\|_2^2$$

This nondecreasing function can be $\alpha(f_w) = \log(f_w)$ or $\alpha(f_w) = 1_{f_w \geq \tau}$ where τ is some threshold count.

In the applications, results do not significantly change when using ridge regression or LASSO regression. A nonlinear transformation may work best, given that the relationship between the pretrained and non-pretrained components is almost certainly not linear. However, in practice, a nonlinear transformation—like a two-layer neural network—involves extensive hyperparameter tuning. In the following applications, a nonlinear transformation, even after extensive hyperparameter tuning, could not achieve the same performance as a linear transformation.

Whether to finetune the BERT model is another open question and one that requires further study. Finetuning the BERT model to predict the time period that a document came from or finetuning the model using a contrastive learning approach (Le-Khac et al., 2020) could potentially improve the BERT embeddings. In practice, finetuning BERT using various contrastive approaches did not improve results; using BERT out of the box seems to be the best approach. Further research is required to determine any finetuning tasks for BERT that could improve the BERT embeddings in the PAE framework.

The layer of BERT where the BERT embeddings come from can also affect the quality of the PAE embeddings. Many works have shown that lexical semantic information is best encoded in the earlier layers of BERT (Bommasani et al., 2020; Tenney et al., 2019; Liu et al., 2019). Throughout this paper, I use the embeddings from the fourth layer of BERT. Further research is required to see which layer works best for the PAE framework, and it is most likely different for each application.

It is also not known when a document is too long in length to be considered a word's context. The applications discussed in this paper deal exclusively with smaller corpora and short text per observation. With lengthier documents, one may need to infer a new document vector of the phrase with a specific window around the word of interest. At what point the document is too long is left to future study.

4.4 Application 1: *New York Times* Articles with "Equality" in Headlines

4.4.1 Dataset

I use the PAE framework with the data from Rodman (2020). Rodman (2020) tests four time-sensitive implementations of word2vec against a gold standard that was developed from a dataset of *New York Times* articles from 1855 to 2016 that had the word "equality" in their headlines. For each article, the text consists of the headline, the first paragraph, and the abstract. Rodman (2020) splits the corpus into seven 25-year time slices. There is a total of 3,105 articles, and the number of articles in each chronological slice ranged from 80 to 660 articles.

The goal of Rodman (2020)'s project was to understand "who, in a given article, is seeking, achieving, needing, or being denied equality—in other words, which group was associated with the quality discussed in a given article" (Rodman, 2020). The author and two undergraduate coders manually labeled articles into fifteen mutually exclusive and exhaustive topic codes. An iterative coding method yielded 400 articles that served as a training set for a supervised topic model. For each era, the author used the full training set to model each era separately; bootstrapping was then used in each era to produce means for the document proportions for each topic. Five of the fifteen topics were then selected because these were the topics that could be easily approximated using a single word. Specifically, the topics selected were "gender," "international relations," "Germany," "race," and "African American." For each topic, the author constructed a set of constituent keywords replaced by a single collapsed word. Table 4.1 describes the constituent words and the collapsed word for each topic. These five categories collectively form the "gold standard" dataset. The author assumed that the proportion of texts in each topic "tracks the strength of the semantic relationship in the corpus between equality and the topic."

The author then uses four chronological word2vec models to see how well the cosine similarities between the word "equality" and the five test words representing each topic correlate with the gold standard dataset. The naïve time model resampled documents from the corpus and modeled the resampled corpus from each time period until the bootstrapped mean around the cosine similarity scores for the five test words stabilized. The overlapping time model uses a small proportion of the previous time period's documents to smooth the embeddings learned between two periods. Each era was bootstrapped as per the naïve time model. The chronologically trained model initializes each time period's word embedding space using the previous era's word embedding space, with the first space initialized with the

Topic	Collapsed Word	Constituent Words
Gender	"gender"	gender(s), woman, woman's, women, women's, female, suffragette(s), sexes, sex
International Relations	"treaty"	treaty, treaties, pact(s)
Germany	"german"	german(s), germany
Race	"race"	race, races, racial
African American	"african_american"	freedman, freedmen, blacks, african american(s), mulatto(es), negro, negroes, colored

Table 4.1: The constituent words and the respective collapsed word for each topic of interest in Rodman (2020).

word embedding space trained over the entire corpus, and each time slice was bootstrapped per the naïve time model. Lastly, the aligned model separately modeled each corpus slice. The word vector spaces in adjacent slices were aligned in sequence using a Procrustes matrix alignment, per Hamilton et al. (2016b). The author finds that the chronologically trained model has the highest correlation with the gold standard dataset, with a correlation of 0.611.

4.4.1.1 Proportions Using Keywords

Rodman (2020)'s dataset is one of the few datasets in the literature on lexical semantic change and word usage change with human-labeled annotations of changes over time (Kutuzov et al., 2018). Rodman (2020)'s dataset also uses actual articles from the *New York Times*, which makes it a more useful dataset for studying language evolution than the artificially created datasets that are popular in this literature (Kutuzov et al., 2018; Hengchen et al., 2021).

The assumption that the proportion of texts in each topic tracks the strength of the semantic relationship in the corpus between equality and a given topic proves problematic when evaluating how well a model performs. For each of the five test terms, I calculate the proportion of *New York Times* articles from each period that contain a specific test term. I then calculate the correlation, the sum of absolute deviance, and the sum of squared deviance using this approach. Table 4.2 shows the average correlation, the sum of absolute deviance, and the sum of squared deviance across all of the five topics.

Simply calculating the proportion of documents containing a given keyword is enough to dramatically outperform the word embedding methods proposed in Rodman (2020) while being much less computationally expensive to calculate and much more conceptually straightforward. The downside, of course, is that there is no word embedding space to explore relationships between other words, see which words have the closest meaning to a given word,

	Deviance	Squared Deviance	Correlation
Chronologically Trained Model	22.689	21.860	0.611
Proportions of Keywords	14.771	11.899	0.829

Table 4.2: The proportions of keywords by time period compared to the gold standard model. Results of the "Chronologically Trained Model," the best performing word2vec model from Rodman (2020), is included for comparison.

or how words move closer or farther apart from each other in meaning (in the distributional hypothesis sense). The proportions approach may also fail if the corpus is too small. For those reasons, the word2vec approaches are still useful. However, seeing how the proportion of documents containing a given keyword compares with the human-labeled dataset is an essential baseline that any word usage change method should be measured against.

4.4.2 Preprocessing the Data and Hyperparameter Choices

The text from each article had its punctuation stripped and stopwords removed. This preprocessing differs slightly from Rodman (2020), who also stripped punctuation but did not remove stopwords. Rodman (2020) also split all documents into sentences; in other words, her approach treated every sentence as a separate document. I kept each article (headline, first paragraph, and abstract) as one document. I also substituted a set of constituent words associated with each topic of interest with a single collapsed word, as Rodman (2020) does. Table 4.1 contains the full list of constituent word substitutions made in the corpus. Lastly, I removed the word "equality" from the dataset, as all documents contained at least one instance of this word. Removing a word that occurs very commonly can improve the quality of word and document embeddings as it prevents the neural network used to fit the word and document embeddings from predicting the common word in every context.

I used the PV-DBOW algorithm of doc2vec with skip-gram to learn the document and word embeddings. This produces a vector space that contains both the word and document embeddings. PV-DBOW generally produced more stable results than PV-DM and was faster to train over the corpus. I used a vector size of 300, a window size of 5, and a learning rate of 0.025 decaying to 0.0001. I also fit word vectors for words that occurred at least 12 times in the corpus. 100 epochs were used to fit the word and document embeddings; epochs between 50 and 200 generally produced similar results.

For the pretrained word embeddings, I used the word embeddings pretrained on the Google News dataset, which contains about 100 billion words (Mikolov et al., 2013). The model contains 300-dimensional word vectors for 3 million words or short phrases (such as "new_york"). For pretrained context embeddings, I used the pretrained BERT model with

no further pretraining. I used the BERT base model, which consists of context vectors of 768 dimensions.

I calculated the linear transformation matrix $\mathbf{A}$ using Algorithm 4.1, and calculated time-specific embeddings for words using Algorithm 4.2. I did not weigh each point by an $\alpha(f_w)$ function. To construct the time-specific word embedding for "equality," I averaged all PAE context embeddings by each time period and multiplied this by $\mathbf{A}$. The word "equality" was removed, but every document contained this word by construction.

4.4.3 Results

Table 4.3 contains the full set of results.

	Deviance	**Squared Deviance**	**Correlation**
Chronologically Trained Model	22.689	21.860	0.611
Proportions of Keywords	14.771	11.899	0.829
PAE Framework	13.557	9.418	0.864

Table 4.3: Results comparing the best diachronic word2vec model, the chronologically trained model, from Rodman (2020) with PAE averaged across 50 iterations. Also included are the results from the proportions of keywords, as explained in Section 4.4.1.1.

I used the average of 50 separate initializations of doc2vec to calculate the sum of absolute deviance, the sum of squared deviance, and the average correlation between the cosine similarities of the time-specific embedding for "equality" and the time-specific embeddings of the five test terms across the seven 25-year eras. I use an average because different initializations of doc2vec can lead to different results. Because I use BERT out of the box, there is no issue with initializations or sampling. Across the 50 separate initializations, the lowest average correlation across the five topics was 0.819 and the highest average correlation across the five topics was 0.905.

As the results indicate, the results using the PAE framework far outperform the best model in Rodman (2020). It also outperforms taking the proportions of documents containing keywords for each era, as explained in Section 4.4.1.1. These results of the PAE framework suggest that it finds relationships between equality and the five major social categories similar to human-annotated relationships. Figure 4.1 plots the z-score normalized model outputs from the PAE framework and the gold standard dataset. As the graphs indicate, the PAE framework follows the gold standard quite closely in all categories of interest.

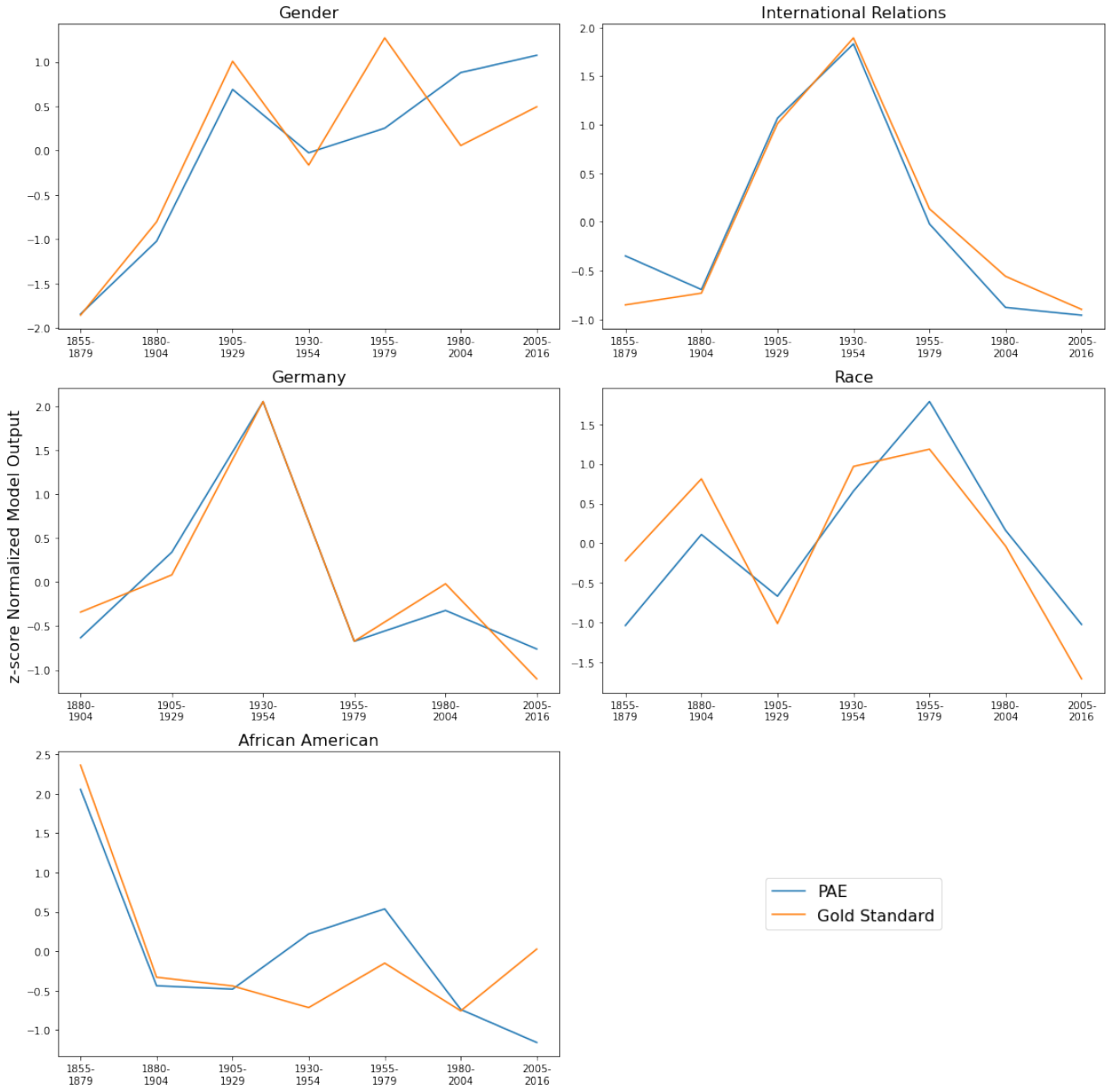


Figure 4.1: PAE framework correlations between time-specific embeddings compared with the gold standard results, using z-score normalized model outputs. Results are averaged across 50 iterations.

4.4.4 Framework Variations

I compare the performance of the PAE framework with various modifications. I compare the PAE framework with the following variations:

- Using the weighted sum of pretrained and non-pretrained word embeddings for $\mathbf{v_w}$ and only using doc2vec embeddings for the context embeddings $\mathbf{u_w}$ (PAE WE on Doc2Vec Context, Variation (1))
- Using the weighted sum of pretrained and non-pretrained word embeddings for $\mathbf{v_w}$ and only using BERT embeddings for the context embeddings $\mathbf{u_w}$ (PAE WE on BERT Context, Variation (2))
- Using only non-pretrained word embeddings for $\mathbf{v_w}$ and using doc2vec embeddings concatenated with BERT embeddings for the context embeddings $\mathbf{u_w}$ (WE on PAE Context, Variation (3))
- Using only non-pretrained word embeddings for $\mathbf{v_w}$ and using only doc2vec embeddings for the context embeddings $\mathbf{u_w}$ (WE on Doc2vec Context, Variation (4))
- Using the weighted sum of pretrained and non-pretrained word2vec embeddings for $\mathbf{v_w}$ and using the average of word2vec embeddings concatenated with BERT embeddings for the context embeddings $\mathbf{u_w}$ (PAE W2V WE on PAE W2V Context, Variation (5)); this variation combines the PAE framework with the approach used in Khodak et al. (2018) and it assumes that context vectors are the same in every time period

For each framework variation, I learn a linear transformation between the context vectors and the word vectors. I then obtain time-specific word embeddings using Algorithm 4.2. Table 4.4 contains the results across these framework variations.

	Deviance	**Sq'd Deviance**	**Correlation**
(1) PAE WE on Doc2Vec Context	19.149	17.255	0.752
(2) PAE WE on BERT Context	20.808	21.708	0.687
(3) WE on PAE Context	17.246	16.275	0.765
(4) WE on Doc2Vec Context	19.975	19.262	0.723
(5) PAE W2V WE on PAE W2V Context	19.277	16.912	0.755
PAE WE on PAE Context	13.557	9.418	0.864

Table 4.4: The sum of absolute deviance, the sum of squared deviance, and the average correlation across the five topics of interest. Results are averaged over 50 iterations. Although the variations outperform the chronologically trained model, they do not outperform the proportions of keywords approach nor the PAE Framework.

I calculated the results using variations of the PAE framework using the same 50 iterations used to calculate the results in Table 4.3. These results suggest that the combination of pretrained and non-pretrained components leads to the exceptionally high average correlation and low sum of deviance and sum of squared deviance.

In variation (5), I used 50 iterations of word2vec instead of doc2vec. I used the same hyperparameters as the doc2vec models and used skip-gram. I used a window size of 5 as the context, and averaged the word embeddings within the context. I use these contexts to learn the linear transformation. This variation demonstrates the importance of using document embeddings as a keyword's context rather than averaging the words surrounding the keyword when analyzing word usage over time.

4.4.5 Most Similar Words by Time

I also look at the words most closely associated with each of the five keywords. To calculate the most similar words by time, I first calculate the time-sensitive embedding for the keyword for each time period. I then calculate the time-sensitive embeddings for all unique words in each time period. I then compare each keyword's time-sensitive embedding in a given period with all the other time-sensitive embeddings. Words are more closely associated (have similar usage) if they have a higher cosine similarity.

The words closest to the word "gender" can be found in Table 4.5, the words closest to the word "treaty" can be found in Table 4.6, the words closest to "German" can be found in Table 4.7, the words closest to "race" can be found in Table 4.8, and the words closest to "African American" can be found in Table 4.9.

1855-1879	hour, speech, ohio, issues, american, states
1880-1904	political, number, clubs, meeting, large, right
1905-1929	men, party, national, mrs, equal, today
1930-1954	rights, men, equal, today, mrs, new
1955-1979	equal, new, men, today, washington, govt
1980-2004	men, editor, rights, american, new, article
2005-2016	marriage, new, court, couples, state, supreme

Table 4.5: Closest words to "gender" by era.

These closest words intuitively capture the political events of each era. For the words closest to "gender," we can see a movement from discussions about women's rights to marriage equality. It is also able to capture events such as World War I and World War II, as shown in the closest words to "treaty" and "German." Lastly, it is also able to capture the struggle for racial equality in the United States, as discussions move from concepts of

1855-1879	religious, debate, austria, minister, day, regard
1880-1904	diaz, country, castle, horse, clock, commerce
1905-1929	compact, chinese, china, states, washington, united
1930-1954	arms, german, versailles, reich, london, french
1955-1979	sign, apprentice, largest, steel, avert, major
1980-2004	trade, prices, narrow, international, foreign, powerful
2005-2016	document, language, ago, anti, page, meeting

Table 4.6: Closest words to "treaty" by era.

1880-1904	return, policy, czechs, demands, vienna, austria
1905-1929	berlin, allies, peace, kaiser, france, security
1930-1954	reich, berlin, arms, french, london, france
1955-1979	west, european, proposal, wilson, discussions, good
1980-2004	institute, draft, means, army, need, feb, editorial
2005-2016	schools, academic, world's, quality, seeking, best

Table 4.7: Closest words to "German" by era.

1855-1879	freedom, orleans, south, business, african_american, new
1880-1904	social, african_american, sheepshead, bay, stakes, horse
1905-1929	african_american, says, japan, league, japanese, pres
1930-1954	african_american, crime, discrimination, york, human, bias
1955-1979	discrimination, bias, today, new, rights, urges
1980-2004	black, university, article, stand, says, professor
2005-2016	castro, segregation, nation, applaud, south, economic

Table 4.8: Closest words to "race" by era.

1855-1879	south, louisiana, georgia, convention, rights, lincoln
1880-1904	south, race, social, says, law, black
1905-1929	social, race, state, meeting, georgia, whites
1930-1954	race, segregation, white, whites, court, harlem
1955-1979	whites, white, washington, black, equal, june
1980-2004	advancement, urges, thomas, lincoln, gays, statement
2005-2016	castro, applaud, age, collins, shift, join

Table 4.9: Closest words to "African American" by era.

"freedom" and the "south" to "segregation" to "advancement." It is even able to capture "Thomas," referencing Clarence Thomas and his confirmation to the United States Supreme Court in 1991.

4.5 Application 2: COVID Mask Tweets

4.5.1 Data

I apply the PAE framework to tweets about masking during the COVID-19 pandemic. Tweets were posted between January 1, 2020 to July 1, 2020. I derive the masking tweets from the COVID-CORE dataset (Lu & Mei, 2020). The COVID-CORE dataset was created from Twitter Decahose data, which represents a 10% sample of all tweets. The tweets making up the COVID-CORE dataset were chosen based on a set of base keywords. Section 4.8.1 in the Supplemental Information contains a list of these base keywords. To extract all tweets mentioning masking, I searched for any tweets with the phrase "mask," ignoring all characters or spaces that came before or after the word and ignoring case. I then kept only English tweets (using the `lang` field) and only kept tweets that were not retweets or replies. I chose not to look at replies because this disproportionately included a small number of users involved in internet arguments. In total, there are 131,709 tweets. I used the same hyperparameters used in the previous application detailed in Section 4.4. Because some words occur across all documents, due to the way the COVID-CORE dataset was constructed, I removed a set of words from the corpus; this is detailed in Section 4.8.2 in the Supplemental Information. Lastly, I chose to look at words on a weekly basis. In other words, tweets are separated by which week they were posted.

4.5.2 Analyzing the Language Usage Evolution of Three Words

I look at the language usage evolution of three words: "mask," "Trump," and "flu." The word "mask" is the central word of interest in this particular dataset. I look at the language usage evolution of "Trump" because President Trump was a central focus of discussion on Twitter during the COVID-19 pandemic. Lastly, I look at the word "flu" because many comparisons were drawn between the flu and COVID-19. I construct the time-sensitive embedding for all unique words in each time period. To study the language usage evolution, I first look at the closest words to each word w in terms of the cosine similarity on a weekly basis. I then project the word embedding for each week down to one dimension using principal component analysis (Pearson, 1901). The use of PCA, although imperfect, is a common way of approximately visualizing the change of a word's evolution over time (see, e.g., Rudolph & Blei, 2018; Giulianelli et al., 2020).

4.5.2.1 Language Usage Evolution of "Mask"

Because I removed the word "mask" from each tweet (because all tweets contain some variant of this word), I could not directly construct the time-sensitive embedding for this particular word. Instead, I take the average of all PAE document embeddings for every tweet each week and then multiply it by the linear transformation matrix. I then find the closest words, using cosine similarity, to these document embeddings. The closest words are listed in Table 4.10.

Jan. 19 - Jan. 25	china, wuhan, surgical, protect, news
Jan. 26 - Feb. 1	china, surgical, outbreak, wuhan, protect
Feb. 2 - Feb. 8	china, medical, outbreak, amid, surgical
Feb. 9 - Feb. 15	china, coronavirusoutbreak, amid, outbreak, wuhan
Feb. 16 - Feb. 22	china, amid, outbreak, masked, surgical
Feb. 23 - Feb. 29	china, protect, need, surgical, health
Mar. 1 - Mar. 7	protect, hand, medical, stop, buying
Mar. 8 - Mar. 14	gloves, hand, protect, medical, stop
Mar. 15 - Mar. 21	gloves, medical, need, use, protect
Mar. 22 - Mar. 28	medical, available, specific, brandon, dee
Mar. 29 - Apr. 4	spread, trump, public, use, china
Apr. 5 - Apr. 11	spread, gloves, help, public, fight
Apr. 12 - Apr. 18	spread, fight, ppe, gloves, use
Apr. 19 - Apr. 25	fight, spread, use, public, protect
Apr. 26 - May 2	spread, trump, public, use, pence
May 3 - May 9	trump, spread, making, free, public
May 10 - May 16	trump, spread, public, social_distancing, gloves
May 17 - May 23	spread, public, trump, gloves, cloth
May 24 - May 30	spread, trump, social_distancing, going, public
May 31 - Jun. 6	spread, protests, protect, going, social_distancing
Jun. 7 - Jun. 13	spread, public, study, use, cases
Jun. 13 - Jun. 20	spread, trump, public, cases, going
Jun. 21 - Jun. 27	trump, public, spread, going, cases
Jun. 28 - Jul. 1	spread, trump, public, cases, going

Table 4.10: The closest word to "mask" by week. Because "mask" was removed from each tweet (every tweet had some variant of this word), I took the average of all PAE document embeddings for every tweet for each week and then multiplied it by the linear transformation matrix. I then find the closest words, using cosine similarity, to these document embeddings. All tweets are from 2020.

The results make sense given our retrospective understanding of the events related to masking and the coronavirus from January 1, 2020 to July 1, 2020. Initially, most discussions about masking are concerning China. The conversation evolves to the topic of protection

from the coronavirus. Towards the end of this time period, the discussion has turned political, as the word "Trump" is consistently one of the most related terms to the tweets.

Figure 4.2 shows the one-dimensional projection of the time-specific pretrained-augmented embeddings of the word "mask" by week. The y-axis can be roughly interpreted as how the word is used, although the actual scale is uninterpretable. The figure suggests that the usage of "masks" is constantly evolving in this time period.

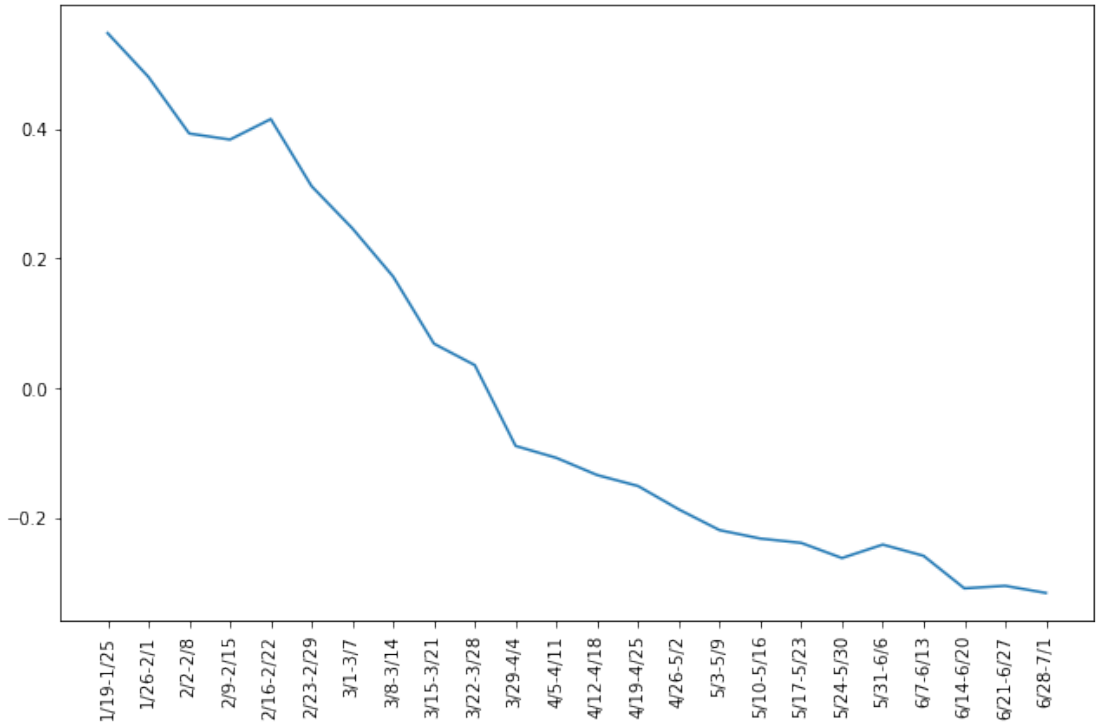


Figure 4.2: The language evolution of the word "mask" as a one-dimensional projection of the time-specific pretrained-augmented embeddings. The y-axis can be roughly interpreted as how the word is used, although the actual scale is uninterpretable.

4.5.2.2 Language Usage Evolution of "Trump"

I next analyze the language usage evolution of the word "Trump." Table 4.11 contains the list of the closest words to the word "Trump" by week. The PAE framework detects specific presidential events, such as the Honeywell factory tour, the Ford plant tour in Michigan, and Trump's decision to hold a presidential campaign rally in Tulsa amid a pandemic. The usage of the word "Trump" evolves with the weeks with respect to the current events related to the COVID-19 pandemic.

Figure 4.3 shows the one-dimensional projection of the time-specific pretrained-augmented embeddings of the word "Trump" by week. What this suggests is that the usage of the word

Feb. 2 - Feb. 8	facial, extreme, chief, new_york, street
Feb. 9 - Feb. 15	issue, medical, lack, supply, close
Feb. 16 - Feb. 22	beijing, realdonaldtrump, order, nmask, dust
Feb. 23 - Feb. 29	response, yorker, pence, china, eyes
Mar. 1 - Mar. 7	boost, fuel, company, billion, plans
Mar. 8 - Mar. 14	donald, palin, sarah, twitter, china
Mar. 15 - Mar. 21	donald, getting, trumpliespeopledie, throwing, supplies
Mar. 22 - Mar. 28	donald, president, blame, touted, ventilators
Mar. 29 - Apr. 4	donald, voluntary, president, americans, administration
Apr. 5 - Apr. 11	administration, donald, deal, updates, failure
Apr. 12 - Apr. 18	failings, turns, stark, crisis, guardian
Apr. 19 - Apr. 25	donald, jim, jordan, proposed, plot
Apr. 26 - May 2	pence, mike, mayo, clinic, ignored
May 3 - May 9	factory, honeywell, donald, president, tour
May 10 - May 16	donald, white_house, president, american, testing
May 17 - May 23	ford, plant, michigan, factory, donald
May 24 - May 30	biden, fool, donald, mocking, calls
May 31 - Jun. 6	swabs, factory, destroys, refusing, batch
Jun. 7 - Jun. 13	america, rally, rallies, administration, deprived
Jun. 13 - Jun. 20	rally, donald, americans, tulsa, supporters
Jun. 21 - Jun. 27	donald, supporters, rally, gop, americans
Jun. 28 - Jul. 1	donald, allies, refuses, worsens, desert

Table 4.11: The closest words to "Trump" by week. All tweets are from 2020.

"Trump" shifted quite a bit at the beginning of the pandemic but stayed relatively constant as time went on.

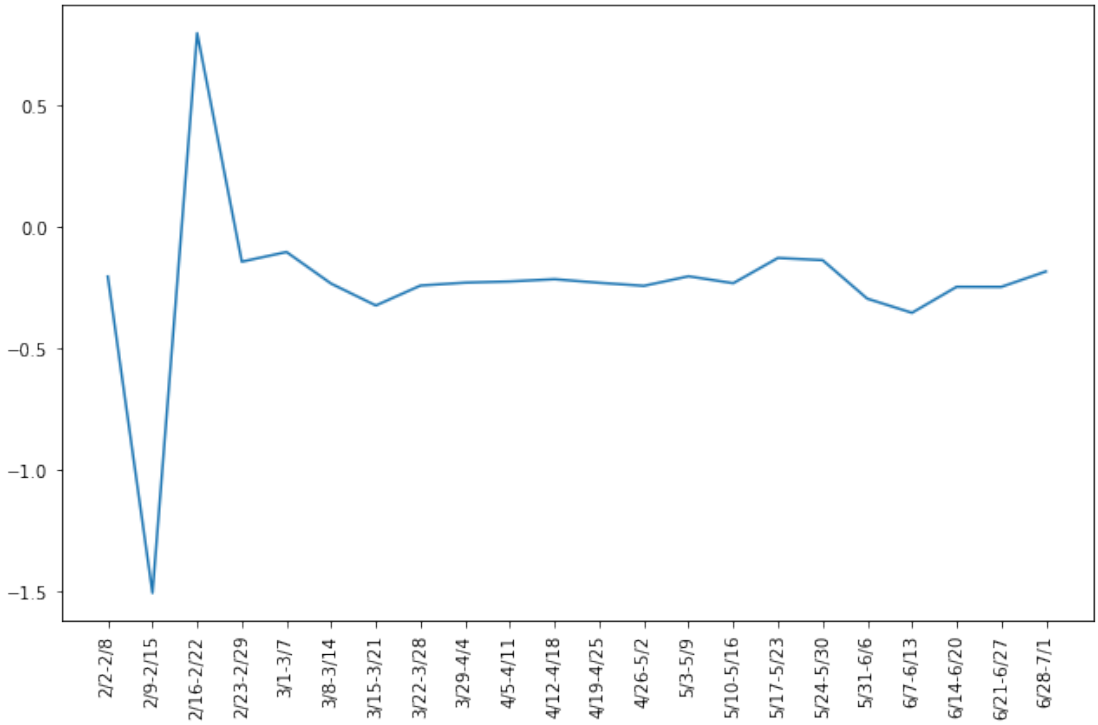


Figure 4.3: The language evolution of the word "Trump" as a one-dimensional projection of the time-specific pretrained-augmented embeddings.

4.5.2.3 Language Usage Evolution of "Flu"

I then analyze the language usage of the word "flu" by week. Table 4.12 lists the closest words. Initial discussions about the flu seemed to involve discussions about staying healthy and China. As time went on, the word took on a more conspiratorial undertone: the word "flu" was used in discussions about overpopulation, riots, and fake news. The word is also used in the context of "kung," referring to "kung flu," a term that President Trump used to describe the coronavirus during a rally on June 20 and again on June 24. The PAE framework captures the expected evolution of the usage of the word "flu."

Figure 4.4 shows the one-dimensional projection of the time-specific pretrained-augmented embeddings of the word "flu" by week. This suggests that the usage of the word "flu" shifted quite a bit at the beginning of the pandemic. There were, however, some shifts towards the end of the dataset's time span. This may have been the result of President Trump using the term "flu" in his rallies.

Jan. 19 - Jan. 25	year, away, wash, hands, forget
Jan. 26 - Feb. 1	beat, pollution, future, help, btw
Feb. 2 - Feb. 8	coronaoutbreak, die, beat, corononavirus, pollution
Feb. 9 - Feb. 15	chinaflu, wuhanflu, valve, coronaoutbreak, respirator
Feb. 16 - Feb. 22	chinaflu, wuhanflu, coronaviruswuhan, coronaoutbreak, respirator
Feb. 23 - Feb. 29	coronaviruswuhan, coronavirususa, chinaflu, coronaoutbreak, wuhanflu
Mar. 1 - Mar. 7	cold, coronavirususa, common, coronavid, sick
Mar. 8 - Mar. 14	coronaupdates, cold, vaccine, coronavirususa, update
Mar. 15 - Mar. 21	negative, phone, testing, calls, capitalism
Mar. 22 - Mar. 28	swine, replenish, obama, advised, admin
Mar. 29 - Apr. 4	illegal, parts, changed, america, spanish
Apr. 5 - Apr. 11	spanish, illegal, changed, parts, fakenews
Apr. 12 - Apr. 18	influenza, spanish, diseases, germs, masksunited
Apr. 19 - Apr. 25	league, spanish, francisco, suffered, timkmak
Apr. 26 - May 2	spanish, viruses, history, kills, wise
May 3 - May 9	spanish, caused, identical, history, years
May 10 - May 16	spanish, surprisingly, vintage, teach, relevant
May 17 - May 23	camps, admitting, halls, pure, dining
May 24 - May 30	fuel, usdtrillion, debtpushers, fuels, overpopulation
May 31 - Jun. 6	months, learned, shot, hat, distance
Jun. 7 - Jun. 13	spanish, players, president, league, faith
Jun. 13 - Jun. 20	spanish, cold, kung, protectothers, influenza
Jun. 21 - Jun. 27	kung, cold, season, spanish, fall
Jun. 28 - Jul. 1	swine, pigs, seasonal, strain, flumask

Table 4.12: The closest words to “flu” by week. All tweets are from 2020.

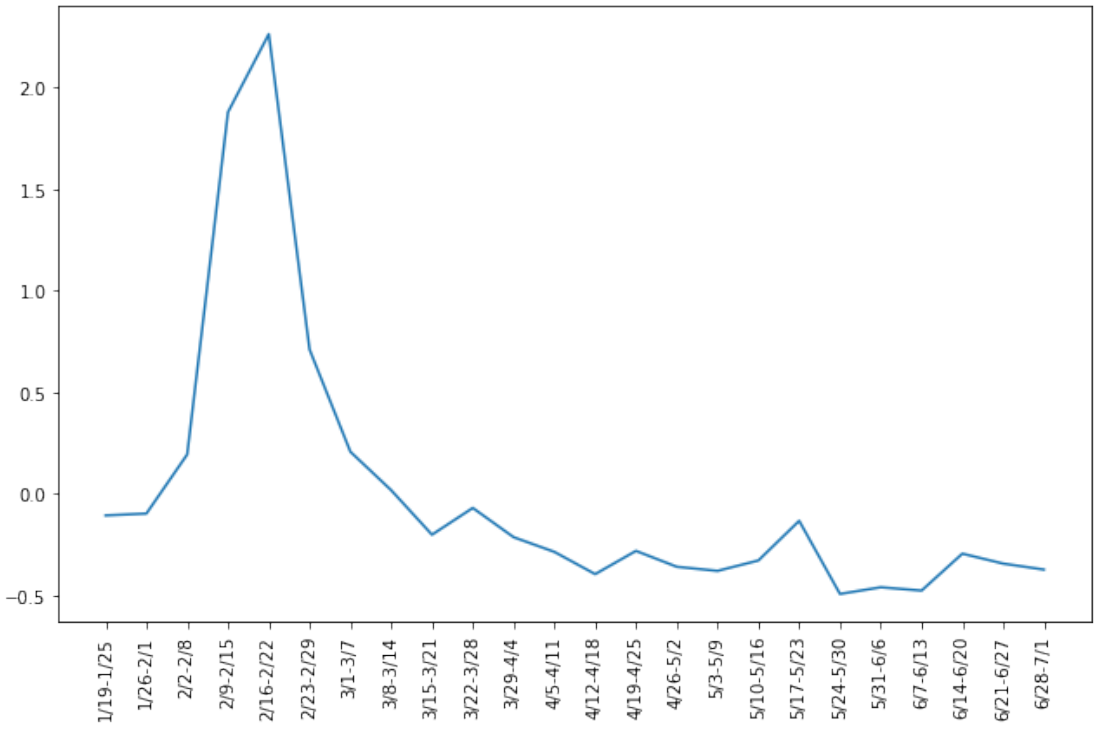


Figure 4.4: The language evolution of the word "flu" as a one-dimensional projection of the time-specific pretrained-augmented embeddings.

4.5.3 Analyzing the Language Usage Evolution of Three Words, Divided by Partisanship

Language usage evolution may differ based on the partisanship of the person tweeting. To study this, I first obtained the partisan association of each Twitter user using their user bios. The details of this method are described in Wu et al. (2019). Informally, obtaining partisan associations involves mapping the bios of Twitter users to document vectors and the unique words that make up all the bios to word vectors. This is done using doc2vec (Le & Mikolov, 2014). Because the document vectors and word vectors are in the same embedding space, they are comparable to one another. To calculate the partisan associations, Wu et al. (2019) take the cosine similarity between document vectors and partisan subspaces, which are created using the differences of the corresponding word embeddings of pairs of partisan keywords or the difference between two groups of averaged word embeddings corresponding to partisan keywords. The idea here is that even though not all users may use partisan keywords in their bios (in fact, oftentimes very few do), Twitter users will list hobbies, activities, and interests with partisan inflections (see, e.g.,. Hetherington & Weiler, 2018; Haieshutter-Rice et al., 2021).

To calculate partisan associations for each Twitter user, I first selected a set of keywords associated with Republicans and Democrats. Because many users negatively use terms related to Republicans or Democrats (e.g., "I dislike Democrats"), I chose terms that would most likely be hashtags or standalone slogans. These are terms not typically used in a negative fashion. Specifically, I chose the terms "MAGA" and "KAG" as Republican keywords, and I chose the terms "resist" and "TheResistance" as Democratic keywords.

To create the partisan subspace, I first averaged the corresponding word embeddings of the Republican keywords, and I averaged the corresponding word embeddings of the Democratic keywords. I then subtracted the average Democratic word embedding from the average Republican word embedding. I then took the cosine similarity between the partisan subspace and each user's bio. In total, 116,259 users had a user bio. I deemed any user who had a partisan association score greater than 0 a "Republican-leaning" user. Similarly, I deemed any user who had a partisan association score less than 0 a "Democratic-leaning" user. 42,909 users were Democratic-leaning, while 73,350 users were Republican-leaning.

I construct time-specific and partisan association-specific words using the linear transformation matrix $\mathbf{A}$ from the analysis in Section 4.5.2. To do this, I average the pretrained-augmented contextual embeddings of each word by both time and party. I then matrix multiply this average by $\mathbf{A}$. In other words, I calculated two sets of embeddings—one for Democratic-leaning users, one for Republican-leaning users—for each week.

4.5.3.1 Analyzing the Language Usage Evolution of "Mask," Divided by Partisanship

The same caveats from Section 4.5.2.1 hold here concerning the fact that I removed the word "mask" from all contexts. All calculations are the same, except I obtain a time and partisan association-sensitive embedding. Table 4.13 lists the top five most related words by week broken down by partisan association.

The list of closest words by week suggests that both Republican-leaning and Democratic-leaning users discussed the masking and the COVID-19 pandemic in mostly similar terms. Both Democratic-leaning and Republican-leaning Twitter users discussed "China" with respect to masking. As time goes on, the discussion of masks turns political, as "Trump" becomes one of the most related words in most weeks. Notably, only Democratic-leaning users use the hashtag "wearamask" in their discussions of masking.

Table 4.13 suggests that discussions about masking were, perhaps surprisingly, quite similar between Democratic-leaning and Republican-leaning Twitter users. While discussions with the word "mask" may use similar words, the nuances of the discussion may differ. I project the time and partisan association-specific embeddings for "mask" down to four

Jan. 19 - Jan. 25 (D)	china, wuhan, surgical, going, hands
Jan. 19 - Jan. 25 (R)	china, surgical, wuhan, protect, medical
Jan. 26 - Feb. 1 (D)	china, chinese, wuhan, protect, outbreak
Jan. 26 - Feb. 1 (R)	surgical, china, outbreak, wuhan, amid
Feb. 2 - Feb. 8 (D)	china, surgical, protect, medical, avoid
Feb. 2 - Feb. 8 (R)	china, outbreak, amid, medical, chinese
Feb. 9 - Feb. 15 (D)	coronavirusoutbreak, china, shortage, coronaviruschina, fears
Feb. 9 - Feb. 15 (R)	china, amid, outbreak, chinese, coronavirusoutbreak
Feb. 16 - Feb. 22 (D)	china, amid, coronavirusoutbreak, wuhan, masked
Feb. 16 - Feb. 22 (R)	china, amid, japan, outbreak, masked
Feb. 23 - Feb. 29 (D)	need, buying, surgical, stop, china
Feb. 23 - Feb. 29 (R)	china, protect, health, buy, coronavirusoutbreak
Mar. 1 - Mar. 7 (D)	buying, protect, stop, need, hands
Mar. 1 - Mar. 7 (R)	hand, medical, coronavirusoutbreak, stock, panic
Mar. 8 - Mar. 14 (D)	hand, gloves, china, world, test
Mar. 8 - Mar. 14 (R)	stop, protect, gloves, medical, hand
Mar. 15 - Mar. 21 (D)	gloves, need, medical, use, hospitals
Mar. 15 - Mar. 21 (R)	gloves, medical, protect, help, need
Mar. 22 - Mar. 28 (D)	medical, available, specific, gloves, ppe
Mar. 22 - Mar. 28 (R)	medical, available, specific, brandon, logan
Mar. 29 - Apr. 4 (D)	trump, spread, public, use, protect
Mar. 29 - Apr. 4 (R)	spread, trump, use, china, public
Apr. 5 - Apr. 11 (D)	spread, gloves, help, cloth, protect
Apr. 5 - Apr. 11 (R)	spread, fight, gloves, public, amid
Apr. 12 - Apr. 18 (D)	spread, gloves, ppe, fight, cloth
Apr. 12 - Apr. 18 (R)	fight, spread, ppe, use, protect
Apr. 19 - Apr. 25 (D)	spread, protect, use, fight, lockdown
Apr. 19 - Apr. 25 (R)	fight, spread, use, making, public
Apr. 26 - May 2 (D)	trump, pence, spread, mike, clinic
Apr. 26 - May 2 (R)	spread, public, use, health, lockdown
May 3 - May 9 (D)	trump, spread, making, public, safe
May 3 - May 9 (R)	spread, trump, free, making, public
May 10 - May 16 (D)	trump, spread, public, gloves, social_distancing
May 10 - May 16 (R)	trump, spread, public, social_distancing, cloth
May 17 - May 23 (D)	trump, spread, public, gloves, protect
May 17 - May 23 (R)	spread, public, trump, cloth, going
May 24 - May 30 (D)	trump, spread, going, public, social_distancing
May 24 - May 30 (R)	spread, trump, social_distancing, going, public
May 31 - Jun. 6 (D)	spread, social_distancing, protest, protests, going
May 31 - Jun. 6 (R)	spread, protect, protests, going, gloves
Jun. 7 - Jun. 13 (D)	public, spread, use, study, cases
Jun. 7 - Jun. 13 (R)	spread, study, use, cases, public
Jun. 14 - Jun. 20 (D)	trump, spread, cases, public, going
Jun. 14 - Jun. 20 (R)	spread, trump, public, cases, social_distancing
Jun. 21 - Jun. 27 (D)	trump, going, spread, cases, public
Jun. 21 - Jun. 27 (R)	trump, public, spread, cases, going
Jun. 28 - Jul. 1 (D)	spread, trump, public, wearamask, going
Jun. 28 - Jul. 1 (R)	spread, trump, public, cases, going

Table 4.13: Closest words by week to "mask," by partisan associations. "D" indicates Democratic-leaning (a partisan association score less than 0), while "R" indicates Republican-leaning (a partisan association score greater than 0).

dimensions using PCA to examine this possibility. Figure 4.5 shows the evolution of the four dimensions by week. The first two dimensions are very similar across the Democratic-leaning and Republican-leaning users, while the third and fourth dimensions differ more. Further research is required to understand what these dimensions mean, but it suggests that usage of the words does differ between Democratic and Republican-leaning users despite the similarity in closest words by week.

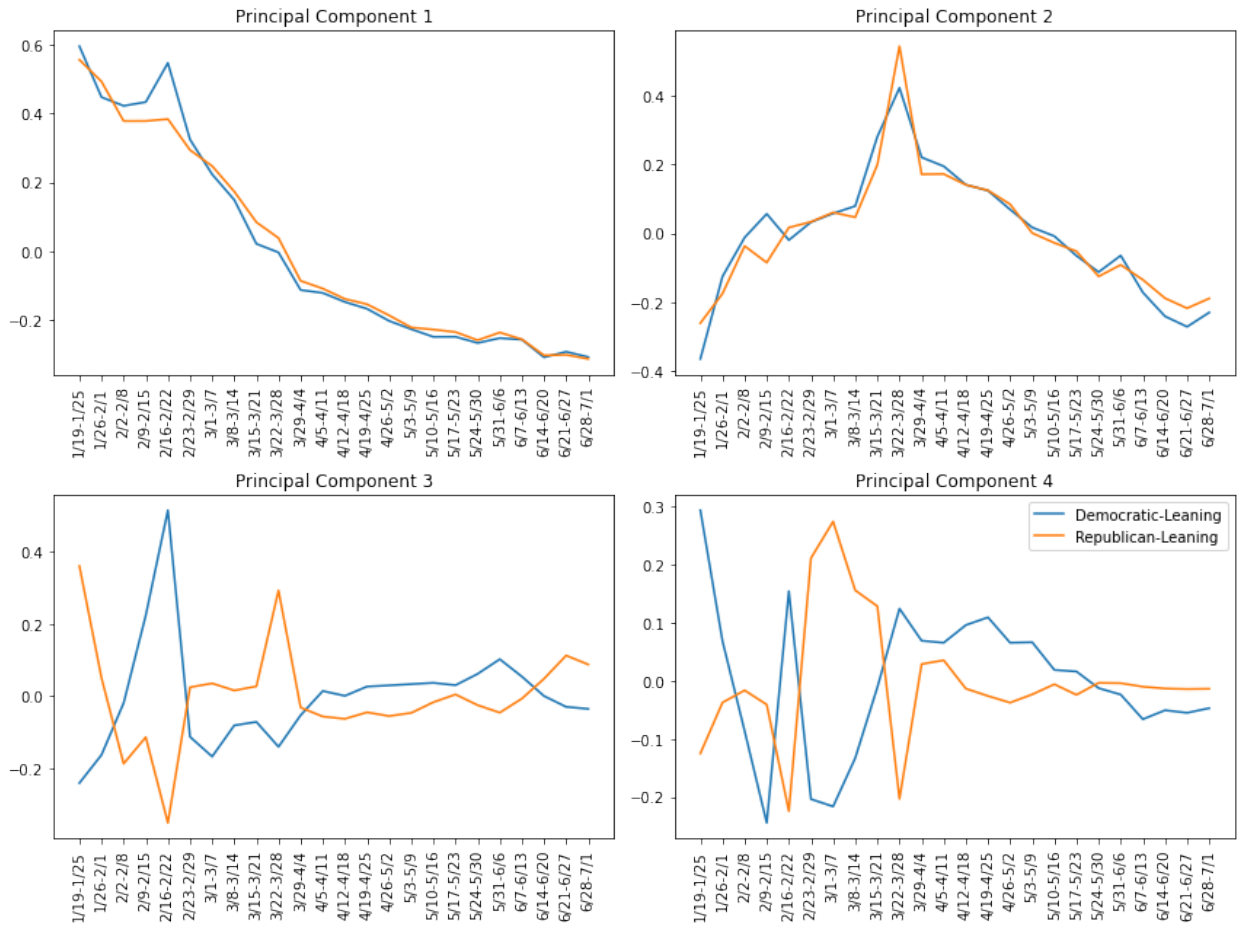


Figure 4.5: The top 4 principal components of the time and partisan association-specific word embeddings for "mask."

4.5.3.2 Analyzing the Language Usage Evolution of "Trump," Divided by Partisanship

I next look at the closest words to "Trump" among tweets discussing masking and the COVID-19 pandemic, divided by partisan association. Table 4.14 lists the top 5 most related words by week. Interestingly, most of the closest time-specific word embeddings to the word

embedding for "Trump" are essentially the same across Democratic-leaning and Republican-leaning Twitter users. Both sides discuss the same news stories, such as the *Rolling Stone* article about the failure of Trump's administration to deliver N95 masks to frontline workers during the week of June 7 to June 13. They also discuss similar events, such as President Trump's visits to the Honeywell factory and the Ford plant.

Feb. 23 - Feb. 29 (D)	americans, response, president, eyes, pence
Feb. 23 - Feb. 29 (R)	messages, mixed, francisco, needed, san
Mar. 1 - Mar. 7 (D)	plans, company, smartnews, millions, combat
Mar. 1 - Mar. 7 (R)	company, calls, boost, combat, plans
Mar. 8 - Mar. 14 (D)	realdonaldtrump, china, close, border, funding
Mar. 8 - Mar. 14 (R)	palin, sarah, singer, retweeetplease, newspicks
Mar. 15 - Mar. 21 (D)	throwing, families, short, tests, ventilators
Mar. 15 - Mar. 21 (R)	donald, getting, fauci, good, hospitals
Mar. 22 - Mar. 28 (D)	doctors, ventilators, donald, hoarding, president
Mar. 22 - Mar. 28 (R)	donald, president, door, petition, sign
Mar. 29 - Apr. 4 (D)	voluntary, americans, donald, administration, cdc
Mar. 29 - Apr. 4 (R)	donald, voluntary, president, americans, won
Apr. 5 - Apr. 11 (D)	updates, failure, warning, issues, knew
Apr. 5 - Apr. 11 (R)	deal, administration, donald, reaches, million
Apr. 12 - Apr. 18 (D)	stark, failings, turns, crisis, guardian
Apr. 12 - Apr. 18 (R)	failings, turns, stark, crisis, dishonest
Apr. 19 - Apr. 25 (D)	proposed, jim, attack, jordan, trumppressconference
Apr. 19 - Apr. 25 (R)	donald, oversight, jim, plot, jordan
Apr. 26 - May 2 (D)	pence, mayo, clinic, mike, stance
Apr. 26 - May 2 (R)	pence, ignored, mike, warnings, refused
May 3 - May 9 (D)	factory, donald, honeywell, tour, valet
May 3 - May 9 (R)	factory, honeywell, president, cnbc, arizona
May 10 - May 16 (D)	white_house, testing, pennsylvania, donald, smartnews
May 10 - May 16 (R)	donald, president, white_house, american, west
May 17 - May 23 (D)	ford, plant, michigan, visit, donald
May 17 - May 23 (R)	ford, plant, michigan, factory, cnbc
May 24 - May 30 (D)	biden, donald, fool, mocking, reporter
May 24 - May 30 (R)	biden, fool, calls, mocking, joe
May 31 - Jun. 6 (D)	swabs, factory, destroys, refusing, entire
May 31 - Jun. 6 (R)	swabs, factory, batch, destroys, refusing
Jun. 7 - Jun. 13 (D)	rally, rallies, maskless, supporters, factory
Jun. 7 - Jun. 13 (R)	america, negligence, deprived, administration, rollingstone
Jun. 14 - Jun. 20 (D)	rally, americans, donald, tulsa, supporters
Jun. 14 - Jun. 20 (R)	rally, tulsa, americans, disapproval, signal
Jun. 21 - Jun. 27 (D)	gop, rally, supporters, arizona, donald
Jun. 21 - Jun. 27 (R)	donald, supporters, rally, president, americans
Jun. 28 - Jul. 1 (D)	donald, refuses, allies, desert, worsens
Jun. 28 - Jul. 1 (R)	donald, allies, refuses, worsens, desert

Table 4.14: Closest words by week to "Trump," by partisan associations. "D" indicates Democratic-leaning (a partisan association score less than 0), while "R" indicates Republican-leaning (a partisan association score greater than 0).

Again, this list suggests that the usage of the word "Trump" was similar across users. To

examine this more closely, Figure 4.6 shows the time and partisan association-specific word embeddings projected down to four dimensions.

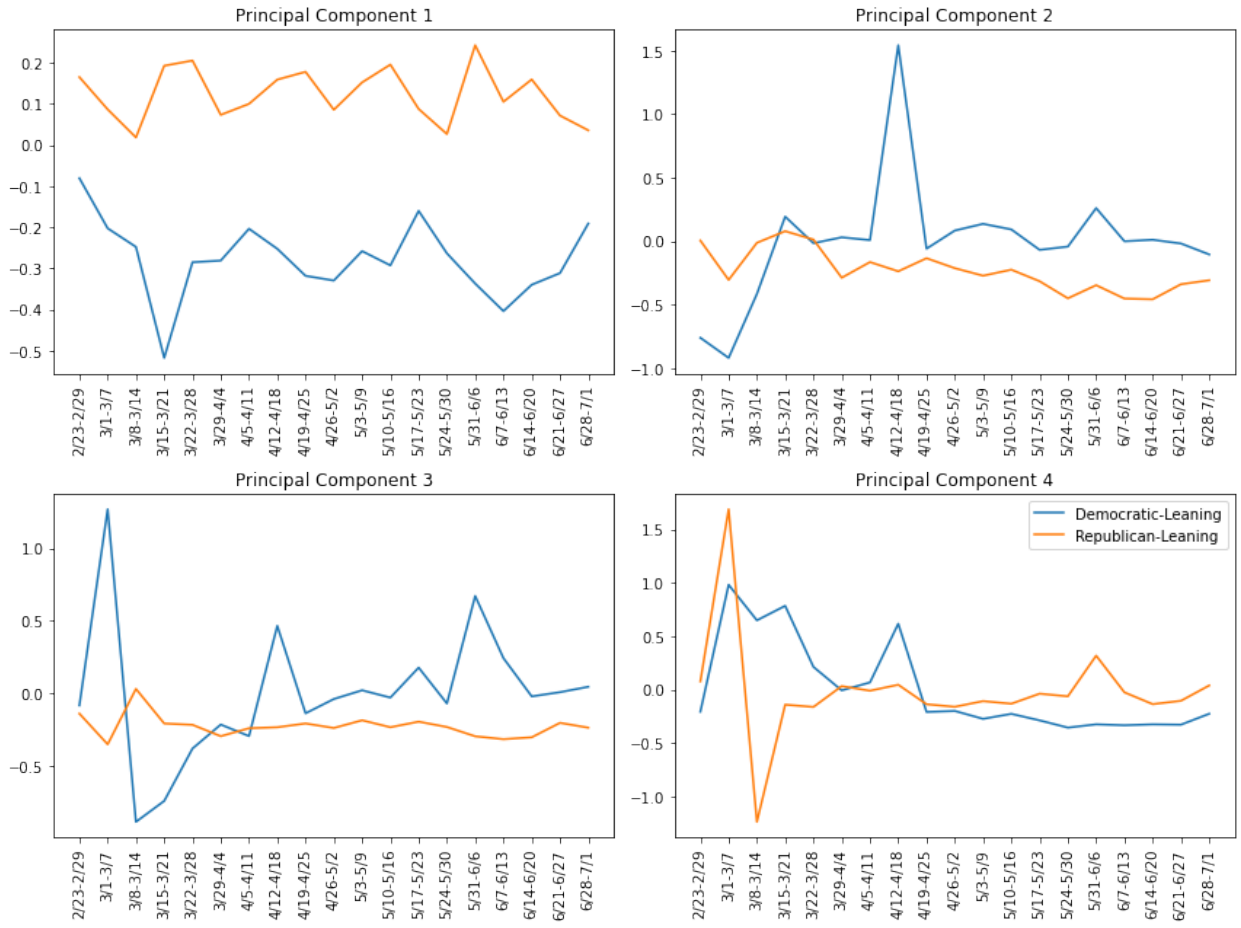


Figure 4.6: The top 4 principal components of the time and partisan association-specific word embeddings for "Trump."

The Democratic-leaning and Republican-leaning time-specific embeddings differ in all four components. The first dimension suggests that the usage of the word "Trump" was mostly consistent within each group but remained distinct across the two groups from February 23 to July 1. Again, further research is needed to give an interpretation to each dimension. This does hint that the nuances of the discussion around the word "Trump" differ by partisan association, even if the closest words to "Trump" by week are similar between Democratic-leaning and Republican-leaning Twitter users.

4.5.3.3 Analyzing the Language Usage Evolution of "Flu," Divided by Partisanship

I then look at the closest words to "flu," divided by partisanship leaning. The top 5 words by week can be found in Table 4.15. The usages of the word "flu" were broadly similar across Republicans and Democrats at the beginning of the pandemic. As the pandemic went on, Republican-leaning users were the first to compare the coronavirus to the 1918 influenza pandemic. Eventually, both sides began to use "flu" in the context of the 1918 pandemic. Republican-leaning users were also the only ones to use "flu" with "kung," indicating usage of the phrase "kung flu." It was around this time that then-President Trump first used the term in one of his rallies.

Again, I project the time and partisan association-specific embeddings for "flu" down to four dimensions. Figure 4.7 shows these four dimensions by week. The first component is approximately the same across both Democratic-leaning and Republican-leaning users. The second component suggests that this dimension of language usage was stabler for Republican-leaning users than Democratic-leaning users. The third and fourth components also differed across the time-specific embeddings. The interpretation of these components is, again, still an open question. It does suggest that the usages of the word "flu" across Democratic-leaning users and Republican-leaning users differ, even if the closest words by week seem broadly similar.

4.6 Conclusion and Future Directions

I propose a framework for studying the usage of words over time. The framework is built to accommodate the shorter, more specialized texts typically found in political science. The method builds on the theory of Arora et al. (2018) and the application of Khodak et al. (2018). Instead of using the word embeddings of a given word's context, which would implicitly assume that the contextual vectors are the same throughout all time periods, I propose using document vectors from doc2vec instead (Le & Mikolov, 2014). However, political science texts are typically short, so the document and word vectors calculated by doc2vec may be poorly fit. To correct this, I propose augmenting the embeddings using corresponding pretrained components. I use BERT embeddings (Devlin et al., 2019) to augment the document embeddings, and I use pretrained word2vec embeddings from the Google News dataset (Mikolov et al., 2013) to augment the word embeddings. I then use these pretrained-augmented embeddings to learn the linear transformation between document embeddings (a given word's context) and a given word's embedding. After learning this linear trans-

Jan. 19 - Jan. 25 (D)	away, start, protect, coronaoutbreak, stay
Jan. 19 - Jan. 25 (R)	year, wash, hands, wipes, feel
Jan. 26 - Feb. 1 (D)	beat, pollution, future, help, got
Jan. 26 - Feb. 1 (R)	canada, usa, amazon, btw, seriously
Feb. 2 - Feb. 8 (D)	pets, hands, time, stop, wash
Feb. 2 - Feb. 8 (R)	coronaviruswuhan, beat, pollution, coronaoutbreak, kit
Feb. 9 - Feb. 15 (D)	coronaviruswuhan, coronavirusoutbreak, protection, symptoms, surgical
Feb. 9 - Feb. 15 (R)	respirator, dust, looking, coronaoutbreak, check
Feb. 16 - Feb. 22 (D)	chinaflu, coronaviruswuhan, coronaoutbreak, wuhanflu, respirator
Feb. 16 - Feb. 22 (R)	wuhanflu, chinaflu, particulate, pack, coronavirususa
Feb. 23 - Feb. 29 (D)	coronavirususa, coronaviruswuhan, wuhan, coronaoutbreak, filter
Feb. 23 - Feb. 29 (R)	covidusa, coronaviruswuhan, coronavirususa, coronaoutbreak, wuhanflu
Mar. 1 - Mar. 7 (D)	coronavirususa, covidus, wuhanvirus, respirator, coronavid
Mar. 1 - Mar. 7 (R)	common, water, lots, getting, cold
Mar. 8 - Mar. 14 (D)	folks, filter, nose, coughing, antivirus
Mar. 8 - Mar. 14 (R)	coronaupdates, cold, vaccine, update, coronavirususa
Mar. 15 - Mar. 21 (D)	thinking, testing, employees, influenza, population
Mar. 15 - Mar. 21 (R)	phone, negative, calls, heart, thread
Mar. 22 - Mar. 28 (D)	silly, swab, makeshift, rule, regular
Mar. 22 - Mar. 28 (R)	swine, replenish, obama, proof, president
Mar. 29 - Apr. 4 (D)	illegal, changed, parts, america, nail
Mar. 29 - Apr. 4 (R)	illegal, parts, changed, spanish, america
Apr. 5 - Apr. 11 (D)	changed, illegal, parts, spanish, entry
Apr. 5 - Apr. 11 (R)	spanish, influenza, tht, fakenews, cats
Apr. 12 - Apr. 18 (D)	influenza, supplychain, masksunited, spanish, covidpandemic
Apr. 12 - Apr. 18 (R)	germs, viruses, deadly, wouldn, politicians
Apr. 19 - Apr. 25 (D)	spanish, league, interesting, francisco, anti
Apr. 19 - Apr. 25 (R)	francisco, suffered, league, thread, timkmak
Apr. 26 - May 2 (D)	kills, killed, viruses, vulnerable, maybe
Apr. 26 - May 2 (R)	spanish, wise, history, teach, season
May 3 - May 9 (D)	history, spanish, teach, worse, bad
May 3 - May 9 (R)	spanish, caused, identical, years, germany
May 10 - May 16 (D)	spanish, teach, history, compulsory, vintage
May 10 - May 16 (R)	spanish, boy, cute, kids, hot
May 17 - May 23 (D)	americans, wore, history, horror, tech
May 17 - May 23 (R)	camps, admitting, dining, halls, pure
May 24 - May 30 (D)	overpopulation, fuels, usdtrillion, fuel, wipeout
May 24 - May 30 (R)	vendetta, likening, bosses, father, spanish
May 31 - Jun. 6 (D)	fall, months, died, twice, kills
May 31 - Jun. 6 (R)	hat, haze, epidemic, corrupt, finally
Jun. 7 - Jun. 13 (D)	spanish, treatments, thinks, joebiden, illness
Jun. 7 - Jun. 13 (R)	faith, president, spanish, germs, abc
Jun. 14 - Jun. 20 (D)	spanish, influenza, cold, hearing, wearamask
Jun. 14 - Jun. 20 (R)	causes, bacteria, proof, cold, believe
Jun. 21 - Jun. 27 (D)	season, fall, winter, cold, considering
Jun. 21 - Jun. 27 (R)	kung, cold, spanish, season, obama
Jun. 28 - Jul. 1 (D)	potential, swine, horrible, control, threat
Jun. 28 - Jul. 1 (R)	swine, stayhealthy, flumask, history, maskers

Table 4.15: Closest words by week to "flu," by partisan associations. "D" indicates Democratic-leaning (a partisan association score less than 0), while "R" indicates Republican-leaning (a partisan association score greater than 0).

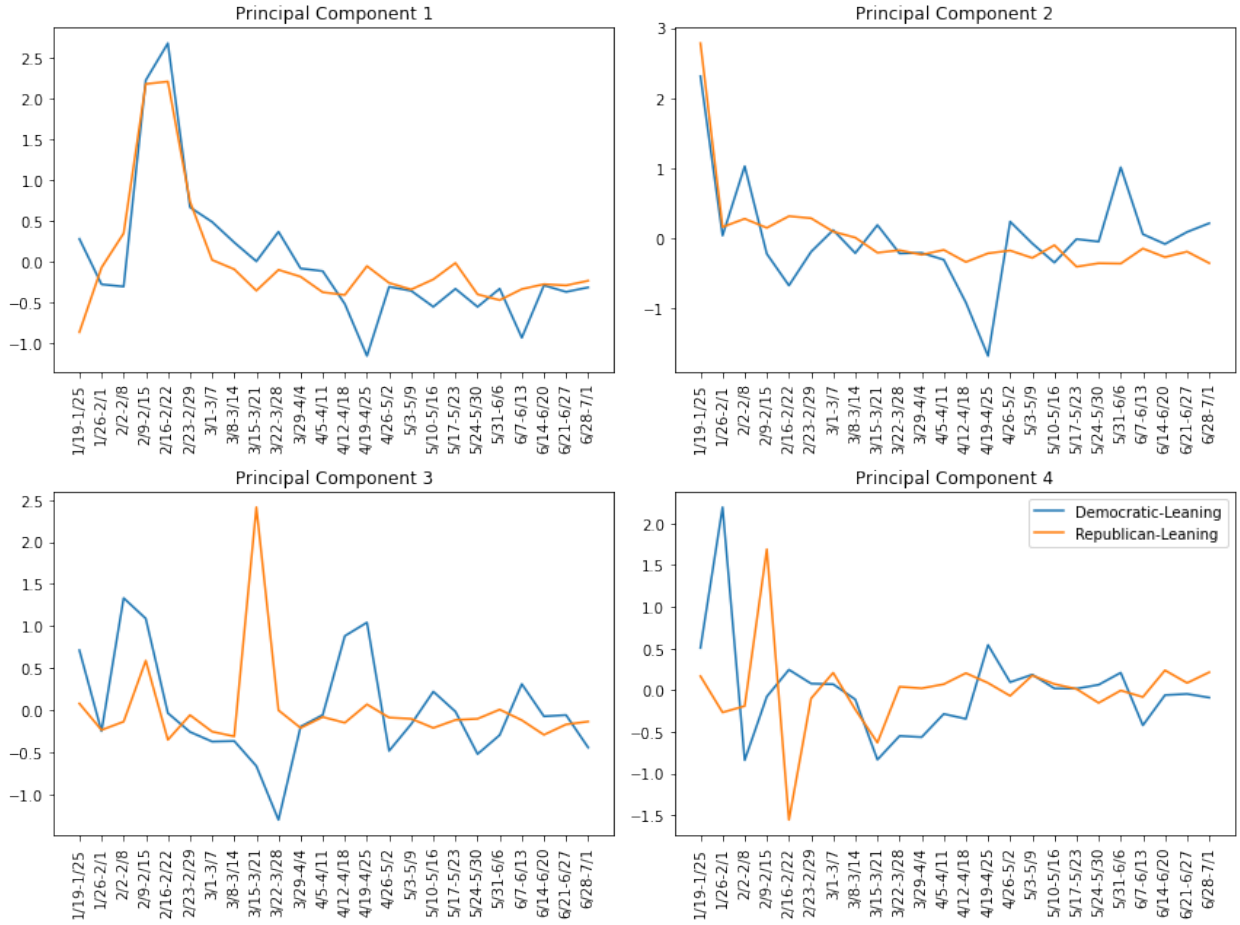


Figure 4.7: The top 4 principal components of the time and partisan association-specific word embeddings for "flu."

formation, I apply this linear transformation to obtain the time-specific embedding for a given word using a given word's context in a specific time period. I call this framework the pretrained-augmented embeddings (PAE) framework.

The contribution of this project is threefold:

1. This is the first work to present empirical evidence that using doc2vec document embeddings can produce time-specific embeddings with better properties than averaged word2vec embeddings.

2. This work is one of the first works, to the best of my knowledge, that meaningfully combines static distributed embeddings with contextual pretrained language model embeddings.

3. The PAE framework can be effectively used with small, specialized text corpora often found in political science.

There are many extensions of this project. In future work, I plan to apply this framework with the method of calculating partisan associations using user profiles to understand how partisan associations shift over time (Wu et al., 2019). Methodologically, it is not clear how well longer documents work with the use of document embeddings. In such cases, the researcher may have to split up documents into smaller documents. In other words, how long a document is until the context is considered "too big" is still an open research question. Theoretically, it is still unclear why combining pretrained elements with non-pretrained elements works better than using either component independently. It is also likely that the relationship between the pretrained-augmented contextual embedding and the pretrained-augmented word embedding is nonlinear, especially given that the BERT embeddings are not necessarily related to each other linearly. How to best learn this nonlinear relationship between the context embeddings and the word embeddings is also an open research question.

4.7 References

Alghanmi, Israa, Espinosa Anke, Luis, & Schockaert, Steven (2020). Combining BERT with static word embeddings for categorizing social media. In *Proceedings of the Sixth Workshop on Noisy User-generated Text (W-NUT 2020)*, (pp. 28–33)., Online. Association for Computational Linguistics.

Arora, Sanjeev, Li, Yuanzhi, Liang, Yingyu, Ma, Tengyu, & Risteski, Andrej (2018). Linear algebraic structure of word senses, with applications to polysemy.

Aslett, Kevin, Webb Williams, Nora, Casas, Andreu, Zuidema, Wesley, & Wilkerson, John (2020). What was the problem in parkland? using social media to measure the effectiveness of issue frames. *Policy Studies Journal*.

Bagozzi, Benjamin E. & Berliner, Daniel (2018). The politics of scrutiny in human rights monitoring: Evidence from structural topic models of us state department human rights reports. *Political Science Research and Methods*, *6*(4), 661–677.

Barberá, Pablo, Casas, Andreu, Nagler, Jonathan, Egan, Patrick J., Bonneau, Richard, Jost, John T., & Tucker, Joshua A. (2019). Who leads? who follows? measuring issue attention and agenda setting by legislators and the mass public using social media data. *American Political Science Review*, *113*(4), 883–901.

Blank, Andreas & Koch, Peter (1999). *Historical Semantics and Cognition*. Cognitive linguistics research; 13. Mouton de Gruyter.

Blei, David M. (2012). Probabilistic topic models. *Commun. ACM*, *55*(4), 77–84.

Blei, David M. & Lafferty, John D. (2006). Dynamic topic models. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, (pp. 113–120)., New York, NY, USA. Association for Computing Machinery.

Blei, David M., Ng, Andrew Y., & Jordan, Michael I. (2003). Latent dirichlet allocation. *J. Mach. Learn. Res.*, *3*(null), 993–1022.

Bloem, Peter (2019). Transformers from scratch. http://www.peterbloem.nl/blog/transformers.

Bommasani, Rishi, Davis, Kelly, & Cardie, Claire (2020). Interpreting Pretrained Contextualized Representations via Reductions to Static Embeddings. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, (pp. 4758–4781)., Online. Association for Computational Linguistics.

Camacho-Collados, Jose & Pilehvar, Mohammad Taher (2018). From word to sense embeddings: A survey on vector representations of meaning. *J. Artif. Int. Res.*, *63*(1), 743–788.

Clark, Kevin, Khandelwal, Urvashi, Levy, Omer, & Manning, Christopher D. (2019). What does bert look at? an analysis of bert's attention.

Coenen, Andy, Reif, Emily, Yuan, Ann, Kim, Been, Pearce, Adam, Viégas, Fernanda, & Wattenberg, Martin (2019). Visualizing and measuring the geometry of bert.

Devlin, Jacob, Chang, Ming-Wei, Lee, Kenton, & Toutanova, Kristina (2019). Bert: Pre-training of deep bidirectional transformers for language understanding.

Firth, John Rupert (1957). *Studies in Linguistic Analysis*. Blackwell.

Garg, Nikhil, Schiebinger, Londa, Jurafsky, Dan, & Zou, James (2018). Word embeddings quantify 100 years of gender and ethnic stereotypes. *Proceedings of the National Academy of Sciences*, *115*(16), E3635–E3644.

Giulianelli, Mario, Del Tredici, Marco, & Fernández, Raquel (2020). Analysing lexical semantic change with contextualised word representations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, (pp. 3960–3973)., Online. Association for Computational Linguistics.

Haieshutter-Rice, Dan, Neuner, Fabian G., & Soroka, Stuart (2021). Cued by culture: Political imagery and partisan evaluations. *Political Behavior*.

Hamilton, William L., Leskovec, Jure, & Jurafsky, Dan (2016a). Cultural shift or linguistic drift? comparing two computational measures of semantic change. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, (pp. 2116–2121)., Austin, Texas. Association for Computational Linguistics.

Hamilton, William L., Leskovec, Jure, & Jurafsky, Dan (2016b). Diachronic word embeddings reveal statistical laws of semantic change. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, (pp. 1489–1501)., Berlin, Germany. Association for Computational Linguistics.

Hengchen, Simon, Tahmasebi, Nina, Schlechtweg, Dominik, & Dubossarsky, Haim (2021). Challenges for computational lexical semantic change.

Hetherington, Marc & Weiler, Jonathan (2018). *Prius or Pickup?: How the Answers to Four Simple Questions Explain America's Great Divide*. Boston: Houghton Mifflin Harcourt.

Hewitt, John & Manning, Christopher D. (2019). A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, (pp. 4129–4138)., Minneapolis, Minnesota. Association for Computational Linguistics.

Hopkins, Daniel J. (2018). The exaggerated life of death panels? the limited but real influence of elite rhetoric in the 2009–2010 health care debate. *Political Behavior*, *40*, 681–709.

Khodak, Mikhail, Saunshi, Nikunj, Liang, Yingyu, Ma, Tengyu, Stewart, Brandon, & Arora, Sanjeev (2018). A la carte embedding: Cheap but effective induction of semantic feature vectors.

Kim, Yoon, Chiu, Yi-I, Hanaki, Kentaro, Hegde, Darshan, & Petrov, Slav (2014). Temporal analysis of language through neural language models. In *Proceedings of the ACL 2014 Workshop on Language Technologies and Computational Social Science*, (pp. 61–65)., Baltimore, MD, USA. Association for Computational Linguistics.

Kulkarni, Vivek, Al-Rfou, Rami, Perozzi, Bryan, & Skiena, Steven (2015). Statistically significant detection of linguistic change. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15, (pp. 625–635)., Republic and Canton of Geneva, CHE. International World Wide Web Conferences Steering Committee.

Kutuzov, Andrey, Øvrelid, Lilja, Szymanski, Terrence, & Velldal, Erik (2018). Diachronic word embeddings and semantic shifts: a survey. In *Proceedings of the 27th International Conference on Computational Linguistics*, (pp. 1384–1397)., Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Le, Quoc V. & Mikolov, Tomas (2014). Distributed representations of sentences and documents.

Le-Khac, Phuc H., Healy, Graham, & Smeaton, Alan F. (2020). Contrastive representation learning: A framework and review. *IEEE Access*, *8*, 193907–193934.

Li, Ping, Schloss, Benjamin, & Follmer, D. Jake (2017). Speaking two "languages" in america: A semantic space analysis of how presidential candidates and their supporters represent abstract political concepts differently. *Behavior Research Methods*, *49*, 1668–1685.

Liu, Nelson F., Gardner, Matt, Belinkov, Yonatan, Peters, Matthew E., & Smith, Noah A. (2019). Linguistic knowledge and transferability of contextual representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, (pp. 1073–1094)., Minneapolis, Minnesota. Association for Computational Linguistics.

Lu, Xuan & Mei, Qiaozhu (2020). COVID-CORE social media dataset.

Martinc, Matej, Kralj Novak, Petra, & Pollak, Senja (2020). Leveraging contextual embeddings for detecting diachronic semantic shift. In *Proceedings of the 12th Language Resources and Evaluation Conference*, (pp. 4811–4819)., Marseille, France. European Language Resources Association.

Martinc, Matej, Montariol, Syrielle, Zosa, Elaine, & Pivovarova, Lidia (2020). Capturing evolution in word usage: Just add more clusters? In *Companion Proceedings of the Web Conference 2020*, WWW '20, (pp. 343–349)., New York, NY, USA. Association for Computing Machinery.

Mikolov, Tomas, Chen, Kai, Corrado, Greg S., & Dean, Jeffrey (2013). Efficient estimation of word representations in vector space.

Mikolov, Tomas, Sutskever, Ilya, Chen, Kai, Corrado, Greg, & Dean, Jeffrey (2013). Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, (pp. 3111–3119)., Red Hook, NY, USA. Curran Associates Inc.

Mu, Jiaqi, Bhat, Suma, & Viswanath, Pramod (2016). Geometry of polysemy.

Park, Baekkwan, Greene, Kevin, & Colaresi, Michael (2020). Human rights are (increasingly) plural: Learning the changing taxonomy of human rights from large-scale text reveals information effects. *American Political Science Review*, *114*(3), 888–910.

Pearson, Karl (1901). Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, *2*(11), 559–572.

Pennington, Jeffrey, Socher, Richard, & Manning, Christopher D. (2014). GloVe: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, (pp. 1532–1543).

Quinn, Kevin M., Monroe, Burt L., Colaresi, Michael, Crespin, Michael H., & Radev, Dragomir R. (2010). How to analyze political attention with minimal assumptions and costs. *American Journal of Political Science*, *54*(1), 209–228.

Reimers, Nils & Gurevych, Iryna (2019). Sentence-bert: Sentence embeddings using siamese bert-networks.

Roberts, Margaret E., Stewart, Brandon M., Tingley, Dustin, & Airoldi, Edoardo M. (2013). The structural topic model and applied social science. In *ICONIP 2013*.

Rodman, Emma (2020). A timely intervention: Tracking the changing meanings of political concepts with word vectors. *Political Analysis*, *28*(1), 87–111.

Rodriguez, Pedro L., Spirling, Arthur, & Stewart, Brandon M. (2021). Embedding regression: Models for context-specific description and inference.

Rogers, Anna, Kovaleva, Olga, & Rumshisky, Anna (2020). A primer in bertology: What we know about how bert works.

Rosenfeld, Alex & Erk, Katrin (2018). Deep neural models of semantic shift. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, (pp. 474–484)., New Orleans, Louisiana. Association for Computational Linguistics.

Rudolph, Maja & Blei, David (2018). Dynamic embeddings for language evolution. In *Proceedings of the 2018 World Wide Web Conference*, WWW '18, (pp. 1003–1011)., Republic and Canton of Geneva, CHE. International World Wide Web Conferences Steering Committee.

Rush, Alexander (2018). The annotated transformer.

Sagi, Eyal, Diermeier, Daniel, & Kaufmann, Stefan (2013). Identifying issue frames in text. *PLOS ONE*, *8*(7), null.

Sun, Yifan, Rao, Nikhil, & Ding, Weicong (2017). A simple approach to learn polysemous word embeddings.

Tenney, Ian, Das, Dipanjan, & Pavlick, Ellie (2019). BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, (pp. 4593–4601)., Florence, Italy. Association for Computational Linguistics.

Tsur, Oren, Calacci, Dan, & Lazer, David (2015). A frame of mind: Using statistical models for detection of framing and agenda setting campaigns. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, (pp. 1629–1638)., Beijing, China. Association for Computational Linguistics.

Vaswani, Ashish, Shazeer, Noam, Parmar, Niki, Uszkoreit, Jakob, Jones, Llion, Gomez, Aidan N, Kaiser, Ł ukasz, & Polosukhin, Illia (2017). Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., & Garnett, R. (Eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Wang, Chong, Blei, David, & Heckerman, David (2008). Continuous time dynamic topic models. In *Proceedings of the Twenty-Fourth Conference on Uncertainty in Artificial Intelligence*, UAI'08, (pp. 579–586)., Arlington, Virginia, USA. AUAI Press.

Wang, Xuerui & McCallum, Andrew (2006). Topics over time: A non-markov continuous-time model of topical trends. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, (pp. 424–433)., New York, NY, USA. Association for Computing Machinery.

Wittgenstein, Ludwig (2009). *Philosophical Investigations*. Wiley-Blackwell.

Wolf, Thomas, Debut, Lysandre, Sanh, Victor, Chaumond, Julien, Delangue, Clement, Moi, Anthony, Cistac, Pierric, Rault, Tim, Louf, Rémi, Funtowicz, Morgan, Davison, Joe, Shleifer, Sam, von Platen, Patrick, Ma, Clara, Jernite, Yacine, Plu, Julien, Xu, Canwen, Scao, Teven Le, Gugger, Sylvain, Drame, Mariama, Lhoest, Quentin, & Rush, Alexander M. (2020). Huggingface's transformers: State-of-the-art natural language processing.

Wu, Patrick Y. & Mebane, Jr., Walter R. (2021). MARMOT: A deep learning framework for constructing multimodal representations for vision-and-language tasks.

Wu, Patrick Y., Mebane, Jr., Walter R., Woods, Logan, Klaver, Joseph, & Due, Preston (2019). Partisan associations of twitter users based on their self-descriptions and word embeddings.

4.8 Supplemental Information

4.8.1 COVID-CORE Dataset Details

The COVID-CORE dataset is derived from the Twitter Decahose, which represents a 10% sample of all tweets. The filter queries used to extract the coronavirus-related tweets are: "covid 19", "covid 2019", "covid", "19 ncov" "2019 ncov", "ncov", "corona virus", "pandemic", "wuhan virus", "chinese virus", "china virus", "sars cov 2", "hcov 19", and "ncov 19". Connectors and spacing were considered using regular expressions, and casing was ignored; for example, a tweet with "COVID19" in its text would be included.

4.8.2 Words Removed from the COVID Masking Tweet Text

The following words were removed from the corpus: "covid", "corona", "masks", "mask", "coronavirus", "wear", "wears", "face", "faces", "facemask", "facemasks", "wearing", "pandemic", "people", "virus", "amp" (an artifact of Twitter links), "like", "ncov", "rt" (which stands for "retweet"), "says", and "new." These were terms used to compile the COVID-CORE dataset together or were terms used to identify the tweets discussing masking during the COVID-19 pandemic. They were removed to make the closest words more interpretable.

CHAPTER 5

Conclusion

This dissertation consists of three papers about leveraging representation learning for political science research. The first paper proposes a method using word embeddings to calculate partisan associations from Twitter users' bios. The second paper develops a deep learning architecture that calculates representations for observations with both images and text and can handle observations that are missing either modality. The third paper outlines a framework for calculating time-specific embeddings usable with small corpora. This concluding chapter briefly reviews the contributions of the three papers specifically to political science and examines the potential links across the three papers.

5.1 Contributions of the Three Papers to Political Science

The technical contributions of these papers help political scientists further their research. The first paper allows political scientists to characterize the partisan associations of users that may not explicitly reveal their partisan associations through their network, tweets, or bio. This contrasts with existing methods of calculating partisan associations or ideal points, which require users to follow partisan accounts or use partisan words in their tweets. Using the partisan association method can lead to calculating partisan associations for a much broader set of users compared to other methods; this means that political scientists can include a much more diverse group of Twitter users in their analysis. This broader set of users includes, but is not limited to, those who may not associate themselves with any parties, those who are less politically involved on social media, and those who choose not to explicitly disclose their partisan associations. In other words, political scientists can gain a clearer picture of how certain phenomena or perceptions of events are related to users' partisan associations across a more comprehensive set of users.

The second paper develops a deep learning architecture that allows political scientists to classify observations with both image and text automatically and can be used with datasets missing one of the two modalities. MARMOT is a bespoke multimodal architecture for political science. Multimodal approaches in computer science require computationally expensive multimodal pretraining and need all observations to have both image and text. MARMOT does not require multimodal pretraining, and it does not require that all observations have both image and text. MARMOT increases the dimensions of social media behavior that can be analyzed: not only does MARMOT look at both modalities in posts that have both image and text, but it also makes multimodal posts comparable to unimodal posts. This allows political scientists to understand further the multifaceted ways individuals use social media to communicate the messages they want to express to the world.

The third paper describes the pretrained-augmented embeddings (PAE) framework, which calculates time-specific embeddings using pretrained and non-pretrained components. The primary contribution of the framework is that it allows political scientists to use natural language processing techniques to automatically study word usage change in small corpora. Previous approaches to studying word usage change using word embeddings can only be used with large corpora. Small corpora mean that certain words occur very infrequently, which can lead to poorly trained word embeddings. Pretrained embeddings are often higher quality word embeddings, as they are usually trained on a large corpus. However, pretrained embeddings may not encode nuances of the words specific to certain corpora. This is particularly relevant for political texts, such as legislative speeches or legal writing, which often contain highly specialized language. The PAE framework takes advantage of both the strengths of non-pretrained and pretrained embeddings to obtain high-quality time-specific embeddings. Such a technique allows an entire body of corpora to be automatically analyzed in ways that were not possible before.

All three papers aim to include a more diverse set of people in the research of politics, better understand the multifaceted ways that people may communicate online, and further advance the view that individuals are dynamic in their viewpoints, stances, and beliefs.

5.2 Examining Links Between the Three Papers

The three papers share a common methodological underpinning: they all use representation learning to help further study or answer political science research questions. There are also potential bridges between the papers that can lead to extensions of the methods found in this dissertation. This section briefly reviews these links between the three papers and how synergies across the three papers can help political scientists further answer important

research questions.

5.2.1 Links Between Paper 1 and Paper 2

Is there an association between users' partisan associations and their use of text-only, image-only, or multimodal posts? This is still an open research question, but one that may be answered by developing a framework between the first and second papers.

An immediate research project, which does not depend on the MARMOT architecture in the second paper, is to see whether the users' partisan associations are correlated with the use of text-only, image-only, or text-and-image posts. We can also use image captions, although imperfect, to study the relationship between image contents and partisan associations.

It would also be possible to augment the MARMOT architecture such that the MARMOT representation encodes the combination of the image, text, and partisan association. In such a case, the partisan association effectively acts as another "modality." Depending on the outcome of interest, the partisan association scores can give greater context to the MARMOT representations and yield better predictive outcomes than MARMOT without partisan associations. The underlying idea is that the partisan associations of users are directly correlated with the contents found in images or text and are correlated with the kind of messages communicated through multimodal posts. Such an approach can help political scientists further study or automatically classify content such as hateful memes, misinformation on social media, and posts discussing politics or the political process.

5.2.2 Links Between Paper 1 and Paper 3

The first and third papers share a close link—the PAE framework developed in the third paper, which allows political scientists to study how word usage changes over time, can be used to study how the partisan associations of users change over time. The idea is that certain words may not be politically inflected in one time period but may be imbued with partisan slants in the following time period, and words that are politically salient in one time period may lose their political relevancy in the next. For example, the word "resistance" was generally not associated with Democrats before Donald Trump was elected president. After his election, the hashtag or phrase "resistance" became synonymous with protests against President Trump and support for (usually Democratic) politicians that opposed Trump (Vogel, 2017). Using the word embedding space from one time period may not correctly reflect the relationships between partisan and (seemingly) non-partisan words in another period.

The PAE framework developed in the third paper calculates time-specific word embeddings. These time-specific word embeddings can calculate the partisan associations for each user's various bios. This may not reflect an actual change in underlying partisanship of the user—research has shown that a person's sense of partisanship is relatively stable later in life (see, e.g., Niemi & Jennings, 1991; Jennings et al., 2009; Iyengar & Krupenkin, 2018). Instead, it may reflect the extent that users associate themselves with specific parties or political candidates. For example, some users may have removed words that indicate support for Donald Trump from their bios after the 2021 United States Capitol attack. The nature of these shifts in partisan associations—e.g., the relationship between shifts in partisan associations and shifts in people's senses of partisanship—is still an open research question, but combining the methods found in the first and third papers could allow us to study such a question.

5.2.3 Links Between Paper 2 and Paper 3

It is well-known that word usage and meaning evolves, but how and which images are used, such as on social media or news media, also changes over time. For example, social media users attempting to post banned content, such as hate speech, often attempt to dodge automated content filters by using an image and text that, on their own, may not be detected as banned content. As these detection algorithms improve, users will slowly begin to use different images and text to avoid these filters while still communicating similar messages. For example, during the COVID-19 pandemic, anti-vaccination groups on Facebook used code words, such as "dance parties," to avoid detection (Collins & Zadrozny, 2021); other groups use images and memes to subtly suggest that vaccinations are dangerous (Ortutay & Seitz, 2021). As Facebook began to pick up on these keywords and ban groups, users quickly switched to other keywords such as "dinner parties" and "swimming clubs." In another example, followers of the extremist "boogaloo" movement use coded language and images of igloos to indicate their adherence to the movement (Collins & Zadrozny, 2021).

Understanding how the usage of both images and words changes adds another dimension to users. This additional dimension can further enhance political scientists' understandings of how users change their communication patterns in light of a changing social media landscape, such as evolving automated content filters, and a changing political environment. The technical aspect of constructing time-specific MARMOT representations is still an open question; however, such an architecture could help answer these critical research questions about behavior on social media and the strategic use of multimodal posts.

5.3 References

Collins, Ben & Zadrozny, Brandy (2021). Anti-vaccine groups changing into 'dance parties' on facebook to avoid detection. *NBC News*.

Iyengar, Shanto & Krupenkin, Masha (2018). Partisanship as social identity; implications for the study of party polarization. *The Forum*, *16*(1), 23–45.

Jennings, M. Kent, Stoker, Laura, & Bowers, Jake (2009). Politics across generations: Family transmission reexamined. *The Journal of Politics*, *71*(3), 782–799.

Niemi, Richard G. & Jennings, M. Kent (1991). Issues and inheritance in the formation of party identification. *American Journal of Political Science*, *35*(4), 970–988.

Ortutay, Barbara & Seitz, Amanda (2021). Defying rules, anti-vaccine accounts thrive on social media. *Associated Press News*.

Vogel, Kenneth P. (2017). The 'resistance,' raising big money, upends liberal politics. *New York Times*.