From Job Descriptions to Occupations: Using Neural Language Models to Code Job Data

Lynda Laughlin
U.S. Census Bureau
lynda.l.laughlin@census.gov

Xi Song
University of Pennsylvania
xisong@sas.upenn.edu

Megan Wisniewski
University of Pennsylvania
meganwis@sas.upenn.edu

Jiahui Xu
Pennsylvania State University
jpx5053@psu.edu

April 8, 2024

Keywords: Jobs and occupations, natural language processing, automated coding, coding errors

Word count (not including references, footnotes, tables, and appendices): 10,862 words, 4 tables, 3 figures

* The authorship is alphabetic. Xi Song acknowledges support from the University of Pennsylvania Population Studies Center, which receives support from the Eunice Shriver Kennedy National Institute of Child Health and Human Development Population Research Infrastructure Program (NIH P2C-HD044964). We are grateful to Zack Han and Aadit Juneja for their excellent research assistance. Any views expressed are those of the authors and not those of the U.S. Census Bureau.

Abstract

Occupation is a fundamental concept in social and policy research, but classifying job descriptions into occupational categories can be challenging and susceptible to errors. Traditionally, this involved expert manual coding, translating detailed, often ambiguous job descriptions to standardized categories, a process both laborious and costly. However, recent advances in computational techniques offer efficient automated coding alternatives. Existing autocoding tools, including the O*NET-SOC AutoCoder, the NIOCCS AutoCoder, and the SOCcer AutoCoder, rely on supervised machine learning methods and string-matching algorithms. Yet these autocoders are not designed to understand semantic meanings in occupational write-in text. We develop a new autocoder based on Google's Text-to-Text Transfer Transformer (T5) model. Like GPT and other large language models, T5 is pretrained on vast amounts of text data. We develop a T5-based occupational classifier (T5-OCC) model with fine-tuned model parameters and training data from occupation write-ins from the 2019 American Community Survey. By comparing our T5-OCC with existing methods, we show that the autocoding accuracy rate increases from 61.8% to 71.1%. Considering the rapid change in neural language models, we conclude by offering suggestions on how to adapt our method for the development of occupational autocoding models in future research.

1 Introduction

Coding job descriptions into occupational categories is a common but challenging task for researchers, analysts, and organizations to classify job roles systematically (Cain and Treiman 1981; Miller, Treiman, Cain, and Roos 1980; Treiman 1979). The process involves translating the detailed content of a job description, sometimes abridged and ambiguous, into a standardized occupational category. Traditionally, the gold standard for such occupational classification has been manual coding by expert coders. This approach involves a meticulous process where the coder begins by thoroughly reading the job description. Once familiarized with the content, the coder matches the job titles and duties specified in the description against predefined criteria in a chosen classification system. Based on this evaluation, the job description is matched with the most fitting category or code. Although this traditional approach can adeptly categorize a vast number of job descriptions into distinct occupations, spanning from a few to hundreds of categories, it is labor-intensive, time-consuming, and often costly.

With the recent development of computational techniques for automated language understanding and production, automated coding methods become a useful alternative to the manual coding method (Census Bureau 2015). These techniques stem from the fields of machine learning and artificial intelligence, specifically designed to enable computers to understand, interpret, and produce human language. These techniques present a suite of advantages that streamline and enhance the coding process. First, these techniques can process large datasets in a fraction of the time a human would take, ensuring efficiency even as the data volume surges. Second, the method provides more inter-coder consistency than human coding. Once a model is trained, it can classify job descriptions uniformly, eliminating the chances of human error or biases that might emerge from different coders. Third, while the initial investment for model development and training might be substantial, the long-term value for cost-saving is very promising. Employing computer-assisted models can be more economical than consistently hiring human coders, particularly for large datasets. Fourth, the dynamic nature of these techniques means that the model is always learning and refining its performance. For example, models built on active and reinforcement learning algorithms enhance their classification precision as they encounter more data. Finally, these methods are well known for

their adaptability. As the landscape of occupational classifications shifts or expands, these models can be easily recalibrated or retrained, making them indispensable tools that cater to a wide range of research demands.

In this paper, we first describe how occupations are collected, coded, and classified from occupation write-ins in U.S. administrative data and social surveys. We then provide a review of existing autocoding methods that have been used to reduce the workload of clerical coders and implemented by the Census Bureau, Occupational Information Network (O*NET), Centers for Disease Control and Prevention (CDC), and the National Cancer Institute (NCI) (Gweon et al. 2017; Savic et al. 2022; Schierholz et al. 2018; Schierholz and Schonlau 2021; Wan et al. 2023). Yet, one limitation of these existing autocoders is that they primarily focus on exact text matches and do not understand semantic meanings. We review several natural language processing models that can be used to improve the accuracy and consistency of current occupational autocoders. The models are developed from Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), and Transformer-Based models, such as the Bidirectional Encoder Representations from Transformers (BERT), the Generative Pre-trained Transformer (GPT), and the Text-To-Text Transfer Transformer (T5). These methods can handle variations in language, including synonyms, paraphrases, and contextual differences, which string-matching algorithms cannot easily accommodate. We further fine-tune these models using manually coded occupations from public use industry and occupation write-in file[1] in the American Community Survey so that the models are not only adept at tasks like text data extraction and production but can discern nuanced, socially formed, and constantly evolving conceptions of occupations. We show that the fine-tuned FLAN-T5-based occupational classifier, referred to as T5-OCC, outperforms existing autocoders and achieves the highest accuracy for coding occupations collected in the 2019 American Community Survey. We conclude with a discussion about how to use T5-OCC and how it can help future sociological research on jobs and occupations.

[1]The U.S. Census Bureau's Disclosure Review Board and Disclosure Avoidance officers reviewed that data product for unauthorized disclosure of confidential information and approved the disclosure avoidance practices applied to this release. CBDRB-FY22-290.

2 Defining Job Titles and Occupations

Occupations are a fundamental component of social stratification. Compared with jobs, occupations carry more social significance and convey information about an individual's social status, identity, and role within society. They are often linked to levels of education, income, and prestige. Beyond academic research, standardized descriptions of occupational titles can also help match job seekers with the right job opportunities. These titles serve as an initial touchpoint for both employers and potential employees, providing a snapshot of the nature and scope of the job.

The development of standardized occupational titles, however, poses a significant challenge. Without a uniform framework for these descriptions, there can be ambiguity and potential misinterpretation. This lack of occupational classifications can lead to job seekers applying for positions that might not align with their skills or expectations, and employers may find themselves sifting through applications that do not match their criteria. Overall, the mismatch caused by non-standardized descriptions can result in longer hiring processes, increased costs, and missed opportunities for both job seekers and employers.

Since 1820, the U.S. Bureau of the Census began to develop a standard index of occupational classifications. Before 2005, occupations of U.S. workers were collected in the federal decennial Census's long-form questionnaire, but this practice changed with the debut of the annual American Community Survey (ACS). The ACS is currently the primary source of occupation data. The ACS questionnaire asks respondents to write descriptions of the type of work and activities they do on the job (see Figure 1). The data also collect information on a respondent's industry. Industry data describe the kind of business conducted by a person's employing organization. These questions ask: "What was the name of this person's employer, business, agency, or branch of the Armed Forces?," "What kind of business or industry was this?," and an item with four check boxes from which respondents are to select one to indicate whether the business was primarily manufacturing, wholesale trade, retail trade, or other (agriculture, construction, service, government, etc.). The occupation and industry questions are asked of all people 15 years old and over who had worked in the past 5 years. For employed people, the data refer to the person's job during the previous week. Both questions about jobs and industry are used in the coding of occupations.

*** FIGURE 1 ABOUT HERE ***

To convert written questionnaire responses into specific Census Occupation Codes, the Census Bureau has developed a complex coding process. The main components of this process are the *Alphabetical Indexes of Industry and Occupation*, the Census Bureau Military Occupation Index, the Employer Name List (ENL), a staff of clerical coders, and more recently, the ACS Industry and Occupation (I & O) Autocoder.[2] The *Alphabetical Indexes of Industry and Occupation* have been developed over time and are continuously updated through the review of survey responses and SOC updates. To classify a respondent's industry and occupation, the *Alphabetical Indexes* contain over 22,000 industry descriptions and 32,000 job titles in alphabetical order.[3] Each combination of industry description, job title, and job duties is assigned a four-digit numeric code by trained clerical Industry and Occupation (I & O) coding staff at the National Processing Center (NPC) in Jeffersonville, Indiana.[4] I & O coding experts convert the written questionnaire responses to codes by comparing these descriptions to entries in the indexes (Census Bureau 2019). We describe the steps in collecting and coding occupation data in ACS in Figure 2.

*** FIGURE 2 ABOUT HERE ***

The Census Occupation Code List is based on the Standard Occupational Classification (SOC) system. The Federal government reclassifies occupations around every 10 years through updates to the SOC system.[5] The Census Bureau code lists are based on even more comprehensive lists issued by the Office of Management and Budget (OMB). The Census occupation classification systems are aggregate versions of the SOC and each Census occupation code can be crosswalked to a SOC code.

As occupations grow or decline over time, new occupations may be broken out and declining occupations may be merged with other occupations to maintain a sufficient sample to be able to

[2]The Census Bureau implemented the ACS Industry and Occupation autocoder in 2012 to supplement clerical coding of industry and occupations in the ACS.

[3]A general overview of the Alphabetical Indexes of Industries and Occupations can be found at <https://www.census.gov/topics/employment/industry-occupation/guidance/indexes.html>.

[4]Similar to occupation, the Census Bureau maintains the Census Industry Code List, an aggregated version of the North American Industry Classification System (NAICS) that is updated every 5 years.

[5]The SOC revision process is overseen by the Standard Occupational Classification Policy Committee (SOCPC), which includes representatives of 11 federal agencies, including the Census Bureau.

publish data on them. As occupations are merged, more differentiated job titles will be present within the merged occupations. In the case of residual occupations (e.g., "all other," "not elsewhere classified"), a large assortment of job titles may be present. Occupations vary in the number of job titles they contain, as well as the number of incumbents. Updating the SOC involves using a variety of methods guided by social and economic theories of work and the economy, with the goal of producing a classification system that is both descriptive and enumerative. The SOC organizes all work performed for pay or profit into detailed occupations by the work performed or usual activities at the job. Education and training are only considered in certain cases.

In addition to ACS, other surveys implemented by the U.S. Census Bureau, such as the Current Population Survey, have adopted similar questions about jobs and industries for the collection of occupational information. Other non-federal government surveys, such as the General Social Survey, also followed a comparable methodology. By employing consistent questions about jobs and industries, these surveys ensure a standardized approach, making it easier to compare and analyze occupation data across different sources and over time. The autocoding methods that we discuss below can be applied to a wide range of surveys that include free-text or open-ended questions about occupations.

3 A Review of Existing Automated Coding Methods

Automated coding methods are widely adopted within federal agencies working with occupation data to increase the standardization of coding and mitigate costs. The autocoders currently used by the U.S. Census Bureau, Occupational Information Network (O*NET), the National Institute for Occupational Safety & Health (NIOSH), and the National Cancer Institute (NCI) share similar methodologies, relying heavily on supervised machine learning techniques, regression models, and string matching applications. We seek to build upon the current autocoder methodology by accounting for semantic similarity using probabilistic/statistical and neural network language models. Below we provide an overview of existing autocoding methods developed for U.S. occupational classifications. We compare the performance of these autocoders with our T5-OCC model and several other language models in the Results Section. A few recent papers have also compared exist-

ing autocoders for coding European and Canadian occupation data into the International Standard Classification of Occupations (ISCO) (De Matteis et al. 2017; Gweon et al. 2017; Rémen et al. 2018; Savic et al. 2022; Wan et al. 2023). For the purpose of our analysis, we focus on U.S. occupation data and codes.

3.1 Current U.S. Census Bureau Autocoder

The American Community Survey (ACS) is a nationally representative survey administered to over 3.5 million addresses across the United States annually. To determine a person's occupation, the ACS asks about a person's job title and main job duties. These open-ended text responses are then recorded in the industry and occupation write-in file. Traditionally, these write-in files would be coded into standard occupational codes by clerical coders. Yet, since 2012, an autocoder was introduced to assign industry and occupation codes using the data found in the industry and occupation write-in file (Thompson, Kornbau, and Vesely 2012).

Data dictionaries are the foundation of the industry and occupation autocoder. Data dictionaries contain common words or phrases found in the industry and occupation write-in file and their corresponding industry and occupation codes. The data dictionaries are divided into three segments: a one-word dictionary, a two-word dictionary, and a dictionary containing complete write-in entries. ACS industry and occupation responses are then matched to these data dictionaries. The matching process returns the industry and/or occupation code(s) most associated with the inputted text response. A logistic regression model is then used to determine which of the codes generated is the most likely to match the industry and occupation code assigned clerically. The accuracy rate of the Census autocoder varies between 20% and 40% (Thompson, Kornbau, and Vesely 2012).[6] However, the Census autocoder is not currently accessible to the public, preventing us from comparing its accuracy with other autocoders when applied to the ACS data in our analysis.

[6]These accuracy rates are estimated based on our communication with Census Bureau researchers.

3.2 O*NET-SOC AutoCoder

The Occupational Information Network (O*NET) is a comprehensive database of job titles and descriptions, as well as skills, knowledge, tasks, and other attributes, for approximately 1,000 occupations in the United States. The database is developed under the sponsorship of the U.S. Department of Labor/Employment and Training Administration. O*NET-SOC refers to the integration of the O*NET database with the SOC system. This modification provides a more detailed and nuanced classification of SOC occupations based on both the nature of the work and the skills and knowledge required. For example, O*NET-SOC breaks the 11-1011 (Chief Executives) code in SOC into two categories: 11-1011.00 (Chief Executives) and 11-1011.03 (Chief Sustainability Officers).

The O*NET-SOC AutoCoder (version 14.1), developed by R.M. Wilson Consulting, Inc. for the U.S. Department of Labor, is an automated coding system designed to assign SOC 2018, O*NET 2019, and Occupational Employment Statistics (OES) 2020 codes to unstructured text data detailing job title and/or job descriptions. Unlike traditional keyword searches, it uses a proprietary parsing and coding algorithm that assigns occupational codes and fit scores to job content. Specifically, the AutoCoder functions by tokenizing job data into individual words and phrases, which are then matched with words and phrases that are associated with O*NET-SOC codes in the O*NET database. The AutoCoder developer has assigned different weights to words and phrases in the O*NET database, ensuring that important words for a specific O*NET-SOC occupation carry more weight in the final match evaluation.

The AutoCoder's user interface allows for input of a job title and a description, and supplemental details like firm name, job category, education level, and industry code, to enhance the accuracy of the code assignment. The generated match score reflects the weighted average from various match methods, with higher scores indicating higher accuracy. A score above 70, for example, indicates at least a 70% accuracy rate in predicting the correct occupational code (O*NET 2023). Note that the accuracy rate reported by O*NET-SOC is not directly comparable to that of the Census autocoder. The O*NET-SOC Autocoder generates multiple possible matches, each of which is associated with a score, whereas the Census autocoder reports a single occupation code with the highest likelihood

of matching. The O*NET-SOC AutoCoder can be accessed on the O*NET website.[7]

3.3 NIOCCS AutoCoder

The National Institute for Occupational Safety and Health (NIOSH) within the Centers for Disease Control and Prevention (CDC) developed the NIOSH Industry and Occupation Computerized Coding System (NIOCCS) V4.0. NIOCCS assigns NAICS 2017 and SOC 2018 codes to unstructured text describing industry and occupation. Similar to the Census Bureau autocoder and the O*NET-SOC AutoCoder, the NIOCCS uses supervised machine learning techniques, such as weighted matching, exact matching, and n-grams to assign industry and occupation codes. NIOCCS conducts these matches based on the NIOCCS Knowledgebase, a repository that contains detailed information about industry and occupation titles. Additionally, the NIOCCS Knowledgebase includes entries for prevalent misspellings, enhancing the precision of the matching process (NIOSH 2023).

The NIOCCS AutoCoder proceeds in several steps. First, similar to the O*NET AutoCoder, the NIOCCS first tokenizes input data into words and phrases. It uses both word swapper and synonym finder to replace unrecognized words with words in their database using a substitute word table ("Knowledgebase"). Then the AutoCoder tries to find the best occupation code for a given phrase using different techniques, including phrase exact match, word proximity match, weighted word match, n-gram match, rule match, SQL full-text search, company match, and phonetics match. Along with the best occupation code, the algorithm also generates an associated confidence level. If the confidence level is above the minimum level required for the autocoding method, then the occupation code selected by the coding engine will be reported. Otherwise, the input data require human intervention for generating an occupation code (NIOSH 2010, 2018; Roberts et al. 2022). The NIOCCS AutoCoder can be accessed either via its website or its application programming interface (API).[8]

[7]The O*NET-SOC AutoCoder can be accessed on its website: https://www.onetsocautocoder.com/plus/onetmatch.

[8]More information about NIOCCS can be found on the website (https://csams.cdc.gov/nioccs/SingleCoding.aspx) and its free API (https://wwwn.cdc.gov/nioccs/IOCode). A batch job can be submitted to autocode multiple entries at a time. However, both job title and industry write-ins are required to complete the autocoding process.

3.4 SOCcer AutoCoder

Standardized Occupation Coding for Computer-assisted Epidemiologic Research (SOCcer) is an automated algorithm designed to compare job descriptions to the U.S. SOC 2010 6-digit codes (Russ et al. 2016, 2023). Similar to the NIOCCS AutoCoder, SOCcer (v2) relies on a knowledge base, which builds on publicly available occupational information from sources like the O*NET, the Bureau of Labor Statistics Direct Match Title File, and the U.S. Census Occupational Code Index. The algorithm first uses job descriptions obtained from a study called the New England Bladder Cancer Study (NEBCS), which consists of 2,631 participants and 14,893 jobs as training data. The SOCcer then estimates the similarity between the job title in the description and those in the knowledge base using three classifiers (J classifier, Jmax classifier, and Pmaxent classifier).[9] The algorithm then incorporates industry information because SOC-2010 codes are often industry-specific. The S>0.01 classifier identifies SOC codes frequently observed within an industry. That is, only SOC codes that are observed with a frequency of 1% or more within an industry are considered significant for classification.[10] To evaluate job tasks, SOCcer employs the fuzzy fingerprint approach (FPP) classifier, which compares task descriptions in the job description to task information associated with SOC-2010 codes from the O*NET database. Specifically, the task information from job descriptions was compared to both the noun and verb fingerprints for each SOC. This comparison yielded a fingerprint similarity measure on a continuous scale between 0 and 1 for each job description and SOC code. The calculation of FPP involved assessing the sum of weights for words in the job description that matched those in the SOC fingerprints for both nouns and verbs, relative to the total sum of weights for all words in the noun and verb SOC fingerprints (see Appendix A in Russ et al. 2023).

SOCcer combines the results from all five classifiers (J, Jmax, Pmaxent, S>0.01, and FPP)

[9]The J classifier uses a soft Jaccard index to compare job titles, considering factors like word variants, misspellings, and abbreviations. The Jmax classifier identifies the SOC code(s) with the highest Jaccard index for a given job title. The Pmaxent classifier uses maximum entropy modeling to estimate the probability of a job title-SOC code match based on the training data.

[10]This classifier assigns equal weight to a group of SOC codes that were observed with a frequency of 1% or more within a given industry. The prevalence of a SOC code within each Standard Industry Classification (SIC) was estimated from the National Industry-Occupation Employment Matrix reported by the U.S. Bureau of Labor Statistics. The classifier then dichotomizes the prevalence data, categorizing SOC codes based on a specified threshold (i.e., 0.01). All SOC codes surpassing this threshold were considered equally valid assignments for that particular industry. SOC codes below this threshold are excluded from the autocoded output.

using logistic regression. This step determines the relative importance of each classifier and scales their results to provide a single score for each job-SOC comparison. The resulting score represents the estimated probability that an expert coder would assign a particular SOC-2010 code to the job description. The SOCcer algorithm provides the top 10 highest-scoring SOC codes for each job description. For each job in the evaluation data, SOCcer selects the SOC code with the highest score as its prediction. This predicted SOC code is then compared to the SOC code assigned by expert coders. The SOCcer AutoCoder can be accessed via the National Cancer Institute's website.[11]

4 Natural Language Processing Models for Automated Coding

The aforementioned autocoders can be considered fine-tuned language models, which are subsets of machine learning models trained to handle unstructured text data and to analyze a sequence of words in a given language. A language model can be used for many different natural language processing (NLP) tasks, such as text summarization, classification, named entity recognition, content generation, and translation. For the purpose of our analysis, we divide language models into three broad categories: heuristic models, probabilistic language models, and neural network language models. We then adapt these models to code occupational write-in raw data in ACS into Census occupational codes and SOC codes. This analysis can be considered a text classification task in NLP.

4.1 Heuristic Language Models

Heuristic language models rely on rules, guidelines, and heuristics, such as common-sense principles or strategies, to analyze language patterns and make predictions. Fuzzy matching methods are commonly considered a heuristic approach, which uses heuristics, rules, and similarity metrics to compare and match strings or data elements that may be similar but not identical. The goal is to find approximate matches or similarities between strings, often in scenarios where exact matches are not expected or required. Heuristic models may not rely heavily on large training datasets or

[11] Users can upload datasets with job titles to SOCcer's website (http://soccer.nci.nih.gov) and receive output datasets containing the 10 highest-scoring SOC codes and their corresponding scores for each job. The results can be reviewed using the companion software, SOCAssign.

expert knowledge. They do not typically use statistical probabilities to inform their predictions. The aforementioned O*NET-SOC, NIOCCS, and SOCcer AutoCoders all fall into this category.

4.1.1 Fuzzy Matching Methods

Fuzzy matching is a method used to identify the approximate equivalence of two or more free-text entries. In fuzzy matching, researchers define rules or similarity thresholds to determine what constitutes a match or a similarity. The similarity distance is mostly calculated by counting character operations like insertion, deletion, substitution, and swapping. The method is particularly useful when dealing with data that may contain spelling errors, abbreviations, or variations of a term (Apostolico and Galil 1997; Gusfield 1997; Sankoff and Kruskal 1983; Wagner and Fischer 1974). Fuzzy matching can be useful in various applications, such as spelling check, record deduplication, approximate string matching, and text similarity analysis. It allows researchers to find relevant results even when the input data contains typos, misspellings, or variations. This method has a long tradition in studies on sequence analysis in linguistics, biology, and sociology (reviewed in Abbott 1995).

Two commonly used algorithms for measuring the difference between two sequences in the fuzzy matching method are the Levenshtein distance[12] and the Jaro-Winkler distance.[13] The output of fuzzy matching for text classification is a list of labels sorted by their similarity distance with the text

[12]The Levenshtein distance measures the minimum number of single-character edits (i.e., insertions, deletions, or substitutions) required to change one word into the other (Levenshtein 1966). When two string texts are identical, their distance is 0. As the distance increases, the strings become more dissimilar (Lesnard 2010). Formally, the Levenshtein distance between two strings, s_1 and s_2, is defined as

$$\text{lev}_{s_1,s_2} = \begin{cases} \max(i,j), & \text{if } \min(i,j) = 0, \\ \min \begin{cases} \text{lev}_{s_1,s_2}(i-1,j)+1 \\ \text{lev}_{s_1,s_2}(i,j-1)+1 \\ \text{lev}_{s_1,s_2}(i-1,j-1)+1_{s_{1i} \neq s_{2j}} \end{cases} & \text{otherwise} \end{cases} \tag{1}$$

where i and j are indexes of characters in strings s_1 and s_2. For example, s_{1i} refers to the character in the s_1 string at the i-th position. The Levenshtein distance can be converted to a similarity score, $\text{sim}_{\text{lev}}(s_1, s_2)$, via normalization as follows

$$\text{sim}_{\text{lev}}(s_1, s_2) = 1 - \frac{\text{lev}_{s_1,s_2}}{\max(|s_1|, |s_2|)} \tag{2}$$

where $\max(|s_1|, |s_2|)$ refers to the maximum Levenshtein distance (all characters are different) between the two strings.

[13]The Jaro-Winkler distance measures the difference between two strings using 0 for exact match and 1 for no similarity (Cohen et al. 2003; Winkler 1990, 2006). The distance score is expressed as 1−similarity. The Jaro-Winkler similarity between two strings s_1 and s_2, sim_{jw}, is a modification of the original Jaro similarity, sim_{j}, and is

entry from low to high. This method has been increasingly used in sociological studies that analyze name or string data (de Leeuw and Keijl 2023; Heiberger, Munoz-Najar Galvez, and McFarland 2021). A potential limitation of the fuzzy matching method is that it does not understand the semantic meaning of a text entry. It only focuses on the structural and character-level similarities between two strings. It determines how closely two pieces of text resemble each other based on their characters, sequence of characters, or structure. For example, "overseeing all business" and "management" may refer to the same occupation but the Jaro-Winkler similarity is 0 and the Levenshtein similarity is 0.167.[14]

To address the limitation of fuzzy matching based on distance measures, some recent research suggests measuring similarity between vectors rather than between sequences of characters (Martin-Caughey 2021). This requires converting the occupational strings into vectors using techniques such as Term Frequency-Inverse Document Frequency (TF-IDF) or word embeddings. In these numerical vectors, each dimension represents a term (word) and its associated weight or frequency. Compared with the distance measures that consider the order of characters within the sequences, the cosine measure represents strings using "bag-of-words" vectors, which means it compares unordered sets.

The cosine measure quantifies similarity by evaluating the cosine of the angle formed between these two vectors. In other words, it computes the dot product of the vectors and divides it by the product of their lengths. This measure is independent of the magnitudes of the vectors and relies solely on the relative angle between them. Formally, the cosine similarity between two vectors X_1

defined as

$$\text{sim}_{\text{jw}}(s_1, s_2) = \text{sim}_{\text{j}}(s_1, s_2) + l \cdot p \cdot \big(1 - \text{sim}_{\text{j}}(s_1, s_2)\big), \text{ where} \tag{3}$$

$$\text{sim}_{\text{j}}(s_1, s_2) = \begin{cases} 0, & \text{if } m = 0 \\ \frac{1}{3}\Big(\frac{m}{|s_1|} + \frac{m}{|s_2|} + \frac{m-t}{m}\Big), & \text{otherwise} \end{cases} \tag{4}$$

$|s_1|$ and $|s_2|$ refer to the length of the two strings, m refers to the number of matching characters, and t is half the number of matching (but different sequence order) characters (i.e., the number of transpositions). When estimating m, two characters from strings 1 and 2 are considered matching if they are the same and not farther than $\left\lfloor \frac{\max(|s_1|,|s_2|)}{2} \right\rfloor - 1$ characters apart. If none of the characters contained in one string can be found in another string, then the strings are not similar and the resulting Jaro similarity score is 0. In addition, to convert the Jaro similarity to the Jaro-Winkler similarity in equation (4), l refers to the length of common prefixes at the start of the string up to a maximum of 4 characters, and p is a constant scaling factor for how much the score is adjusted upward for having common prefixes. Additional human intervention and adjustment through p can often improve the performance of the fuzzy matching method for a specific text classification task.

[14]The Levenshtein distance between the two strings is 20 and the maximum distance is the length of the first string, i.e., 24. Thus the Levenshtein similarity is 0.167 ($= 1 - 20/24$).

and X_2, which are converted from strings s_1 and s_2 respectively, is defined as

$$\text{similarity}_{\cos}(X_1, X_2) = \cos(\theta) = \frac{X_1 \cdot X_2}{||X_1||||X_2||} = \frac{\sum_{i=1}^{n} X_{1i} X_{2i}}{\sqrt{\sum_{i=1}^{n} X_{1i}^2} \cdot \sqrt{\sum_{i=1}^{n} X_{2i}^2}} \tag{5}$$

where X_{1i} and X_{2i} are the ith components of vectors X_1 and X_2.

In the context of occupational write-in data, where each word is assigned a distinct coordinate and a write-in string is represented as a vector detailing the frequency of each word's occurrence within the write-ins, the component values of the vectors cannot be negative and thus the cosine similarity is bounded in $[0, 1]$. In the next section 4.2, we also apply cosine similarity in neural network language models to fine-tune some of the pretrained transformer-based models.

4.2 Probabilistic (Statistical) Language Models

Probabilistic language models, also known as statistical language models, learn patterns of text data with clear model assumptions and likelihood functions. They need more human intervention to correct and learn in the training process than heuristic or neural network models. By contrast, neural network language models learn the underlying text patterns with their own layers of connected and weighted data nodes, which need limited human intervention during pre-training, training, and modeling processes.

Probabilistic language models learn text patterns by modeling the likelihood of specific word sequences. They typically start with the Markov assumption that the probability of the next word in a sequence is determined by the previous n-1 words, i.e., n-grams, and not by any earlier words. In order to assign a given text to one or multiple occupational codes, the models often proceed in two steps. First, the model transforms raw text into a feature vector. For example, the probabilities assigned to specific n-grams in the text can be considered as features. Next, based on the feature vector for a piece of text, the model relies on different machine learning classifiers (e.g., logistic regression and support vector machines) to determine the occupational category of the text. Below, we illustrate two different types of probabilistic language models for the task of autocoding occupational write-in data: conditional random fields model and topic modeling.

4.2.1 Conditional Random Fields Model

A limitation of the fuzzy matching method is that it treats each string independently and does not consider sequential context. The method does not allow researchers to consider the context and dependencies between neighboring words when making predictions. For example, there is a higher probability of encountering the word "president" in an occupational write-in when the preceding word is "vice" compared to when no contextual information is known. The conditional random fields (CRF) model is a type of probabilistic model often used in pattern recognition and classification tasks, and especially for sequence labeling tasks. The model identifies and learns text patterns using in-context word sequences (Lafferty et al. 2001). For text classification, it models the conditional probability of a word with the previous word combination and maps it to a class. Mathematically, the model is expressed as

$$p(\mathbf{y} \mid \mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{t=1}^{T} \exp \left\{ \sum_{k=1}^{K} \theta_k f_k \left(y_t, y_{t-1}, \mathbf{x}_t \right) \right\} \tag{6}$$

where $\mathbf{x}$ is a sequence of input data (e.g., the occupational write-in text for one person); $\mathbf{y}$ is a sequence of labels corresponding to the input data (e.g., Census occupation codes); $\theta_k f_k(\cdot)$ is a nonlinear model mapping extracted text features to a score; and $Z(\mathbf{x})$ is a normalization function converting the scores from word combinations to a probability.

Compared to fuzzy matching methods that generate similarity scores, a CRF model estimates probabilities for different label sequences (i.e., standard occupational classification codes) and uses statistical inference techniques to find the most likely label for the input data. Fuzzy matching, on the other hand, relies on heuristics and similarity metrics without probabilistic modeling. Also, CRFs excel at handling text data with linguistic complexities, whereas fuzzy matching lacks linguistic awareness.

CRFs require supervised learning with annotated training data before making inferences on a new dataset. In the training dataset, each text is assigned to an occupational class label. The goal of this step is to adjust parameters in the CRF model to maximize the likelihood of the observed labels. Then for a new dataset with unlabeled text, the trained CRF will predict the most probable occupational label for each text. In other words, the researcher finds the occupational label that

maximizes the conditional probability defined in equation (6).

One notable limitation of CRFs lies in the computational demands associated with training them, particularly when dealing with extensive datasets. In comparison to neural networks, which can leverage pretraining to expedite the learning process, CRFs rely on the availability of high-quality human-coded input. This reliance on human annotators for creating labeled training data can introduce an additional cost factor into the modeling process. Therefore, CRFs may necessitate more resources and effort in terms of data preparation and labeling, making them potentially less cost-effective, especially in scenarios where large-scale, high-quality annotations are required. In the application of occupational autocoding, this model may require more human clerical coding data as training data than other methods to achieve the same accuracy rates.

4.2.2 Topic Modeling

Topic modeling is widely used to identify clusters or groups of common themes within given unstructured textual data. Topic modeling belongs to unsupervised machine learning and there are two conventional and widely used statistical methods: latent semantic analysis (LSA) and latent Dirichlet analysis (LDA). The newly developed neural network language models advance topic modeling and we will cover those approaches in the next subsection. Given that LDA tends to have better performance than LSA, we use LDA as an illustration of the topic modeling method.[15]

The LDA model assumes that documents (i.e., each occupational write-in entry) on similar topics build on a similar group of words, and thus it relies on word co-occurrences to discover topics within documents. Specifically, each document is modeled as a mixture of various topics, and each topic is a mixture of various terms. Both the topics and words per topic have a sparse Dirichlet prior (Blei 2012; Blei et al. 2003). The total probability of the observed data is specified

[15]LSA extracts the abstract topics and assigns labels for each cluster based on the intuition that words with similar meanings tend to be used together more frequently. For text classification, LSA groups textual entries semantically by the surrounding context and word (and word combination) frequencies.

as

$$\begin{aligned}
&\mathrm{P}(\boldsymbol{\varphi}_{1:K}, \boldsymbol{\theta}_{1:D}, \boldsymbol{Z}_{1:D}, \boldsymbol{W}_{1:D}) \\
&= \prod_{k=1}^{K} \underbrace{\mathrm{P}(\varphi_k)}_{\substack{\text{distribution of} \\ \text{words in topic } k}} \prod_{d=1}^{D} \underbrace{\mathrm{P}(\theta_d)}_{\substack{\text{distribution of topics} \\ \text{in document } d}} \prod_{n=1}^{N} \underbrace{\mathrm{P}(Z_{d,n} \mid \theta_d)}_{\substack{\text{topic assignment of} \\ \text{word } n \text{ in document } d}} \underbrace{\mathrm{P}(w_{d,n} \mid \varphi_{1:K}, Z_{d,n})}_{\substack{\text{observed word } n \\ \text{in document } d}}
\end{aligned} \tag{7}$$

where $\boldsymbol{\varphi}_{1:K}$ are the topics where each $\varphi_k \sim \text{Dirichlet}(\beta)$ is a distribution over the vocabulary; $\theta_d \sim \text{Dirichlet}(\alpha)$ are the topic proportions for document d; $\theta_{d,k}$ is the topic proportion for topic k in document d; z_d are the topic assignments for document d; $z_{d,n}$ is the topic assignment for word n in document d; w_d are the observed words for document d; $w_{d,n}$ is the observed word n in document d; K refers to the number of topics; D refers to the number of documents; N refers to the number of words in document d. For each document, the LDA model first allocates the observed words to K topics, where K is predetermined by the researcher; then for each topic, a document receives a probability that it belongs to this topic based on the frequency of terms appearing in the document and the frequency of terms appearing in each topic. In our analysis, each document refers to an occupational write-in. We choose the number of standard Census occupational categories in 2018 as the predetermined number of latent topics, namely, $K = 568$.

4.3 Neural Network Language Models

Neural network language models are gaining popularity in the analysis of text data in social science research applications (Grimmer, Roberts, and Stewart 2022; Gentzkow, Kelly, and Taddy 2019). These models mimic the human brain in learning unstructured and highly dimensional texts with its network of neurons consisting of layers of connected and weighted data nodes. Different from probabilistic language models, neural network language models understand the semantic meaning of words and can perform multiple complex NLP tasks, such as question-answering. Moreover, they require less human interventions in the training process and in conducting new tasks. When a neural network language model is pre-trained on massive amounts of text data, we refer to this model as a large language model. Large language models contain a huge number of parameters (often billions)

that are used to store a vast amount of language information and linguistic patterns.[16]

There are three general architectures for neural network language models: convolutional neural networks (CNN), recurrent neural networks (RNN), and the transformer (Kim 2014; Medsker and Jain 2001; Vaswani et al. 2017). These three architectures differ in the direction of passing inputs through network layers, text encoding method, and how data nodes are connected. Below we adapt language methods based on the three architectures separately for our specific text classification task in converting ACS occupational write-in data to 2018 Census occupation codes and SOC codes.

4.3.1 Convolutional Neural Network Models

Convolutional neural networks (CNNs) are a type of artificial neural network commonly used in deep learning. CNNs are composed of one or more convolutional layers. In each layer of the network, CNNs apply a set of learnable filters (also known as kernels) to the input data to extract features and learn patterns of the data. CNNs, which are widely used for image analysis and image classification, have also been adapted for text data analysis (Kim 2014). The model typically starts with converting text to numerical vectors, such as word embeddings, which capture semantic relationships between words. Then convolutional layers are applied over the word embeddings to extract local features or patterns from the text. These convolutional filters are small windows that move across the input text, like CT scans, capturing patterns at different scales (e.g., word combinations or n-grams of varying lengths). Each filter focuses on a specific section of the text and detects patterns within that region.

After the convolutional layers, pooling layers are often used to reduce the dimensionality of the extracted features while preserving the most salient information. Following the convolutional step, the collected features are forwarded through a classifier for the subsequent text classification process. This step is also known as fully connected layers, namely by converting the representation of text in word embeddings back to a text format. In essence, CNNs extract textual characteristics from various lengths of word sequence within the input text, which are then employed in the classification task. CNNs are effective at capturing local patterns and features within the text.

[16] In most cases, neural network language models after the Generative Pre-trained Transformer 2 (GPT-2) released in 2019 are considered large language models.

Compared to transformer-based models that require a large number of parameters, CNNs can be computationally efficient with fewer parameters. If computational resources are constrained, smaller CNNs may be preferred to large transformer models due to their lower model complexity. CNNs work well when the focus is on local patterns and the order of words is less important. Yet, it is unknown whether the temporal or sequential aspect of the occupational write-in text data is important for occupational classification.

4.3.2 Recurrent Neural Network Models

Recurrent neural networks (RNNs) are a class of artificial neural networks designed for processing sequential data, where the order of words or word combinations in the text input matters. RNNs introduce a form of memory into the network. At each time step, the RNN model processes the current input and maintains an internal hidden state that depends on the previous state and input. This hidden state effectively captures information about the context or history of the sequence up to that point.[17] RNNs incorporate loops within their networks to integrate new words with the information previously encountered and processed in earlier steps. This integration allows RNNs to learn from the sequential data by updating their internal state with each new input, facilitating the identification of relationships and patterns within the sequence.

For example, when processing a job description such as "direct the overall business activities for a credit card company," the RNN model takes in each word individually and sequentially. As the model processes a word like "business," it has already factored in the context provided by preceding words, such as "direct," "the," and "overall." The RNN model then passes the features learned from this contextual sequential information through a text classification classifier.

RNNs, with their sequential processing capability, typically outperform CNNs in text classification tasks. Our analyses compare these two algorithms to better understand whether the order of words in occupational write-in data matters for occupational classification. However, RNNs are also computationally expensive, as they process one data element at a time, which limits the

[17] In a standard feed-forward neural network, each neuron in one layer is connected to every neuron in the adjacent layer, and each connection has its own weight. However, in RNNs, the same set of weights and biases is used for each time step in the sequence. This weight sharing is applied to all the neurons in the RNN layer. This means that the network parameters (weights and biases) are not specific to a particular time step but are shared across all time steps in the sequence.

potential for parallel computation. Moreover, as the input text length increases, RNNs encounter a vanishing gradient problem, making it difficult to retain information from distant parts of the sequence. Recently, transformer-based models, such as BERT, have gained prominence in text analysis due to their ability to capture global context and long-range dependencies more effectively.

4.3.3 Transformer-Based Models

The Transformer is a deep learning architecture introduced in 2017 that has become the foundation for recently developed large language models (Vaswani et al. 2017). Unlike traditional sequential models, the Transformer utilizes a non-sequential feed-forward neural network that processes input sequences in parallel. This approach offers several advantages over recurrent neural networks (RNNs). In contrast to RNNs, transformer-based models require less training time and do not rely on recurrence. Instead, they leverage a self-attention mechanism that enables the language model to assign varying degrees of importance to input text tokens during analysis and prediction. This self-attention mechanism effectively addresses the vanishing gradient problem often encountered in RNNs, allowing transformer models to capture long-range dependencies without the constraints of sequential processing.

Transformer-based language models typically proceed in three steps. First, an input text is tokenized and positional encoding is used to encode the relative order of each word as a sequence. Second, the encoder applies a set of hidden states representing the input text at different levels of abstraction. Third, the decoder learns from the hidden layers to guess the next word in a local context and generates an output sequence based on the encoded text sequence.

The original transformer is a full encoder-decoder architecture. Different transformer-based language models adapt the full encoder-decoder architecture to be encoder-only, decoder-only, and encoder-decoder models. Generally, encoder-only models (e.g., BERT and its variants) only mask tokens in the input and are suitable for NLP tasks such as text classification and named entity recognition. Decoder-only models (e.g., GPT and its variants) randomly mask tokens on the output side in the pre-training process and are suitable for text generation. The encoder-decoder models (e.g., T5 and its variants) require both the input and output text in the pre-training process and randomly mask tokens on both sides. The full encoder-decoder models are suitable for text-infilling

tasks, such as text translation and text summarization.

BERT Models Bidirectional Encoder Representations from Transformers (BERT) is an encoder-only transformer-based language model introduced in 2018 (Devlin et al. 2018). The model was pre-trained on a large volume of English corpus from various sources on the internet. This corpus consists of books, articles, websites, and other text sources. BERT processes this massive collection of sentences and paragraphs in English using a self-supervised learning approach. Self-supervised learning involves understanding and representing the underlying English language without any specific guidance or labeled data. Specifically, BERT aims to learn meaningful representations of language by trying to predict missing words in sentences. It takes a sentence, randomly masks out 15% of the words, and then tries to predict what those missing words are. This task forces BERT to understand the context and relationships between words in a sentence. Different from RNNs, which process the words one after the other sequentially, BERT can learn a bidirectional representation of the sentence. It can also learn the semantic order of two sentences and their relationships by concatenating two masked sentences as inputs and making predictions about whether the two text inputs are following each other.

There are multiple variants of BERT. Sentence-BERT (sBERT) is designed for sentence embedding and focuses on learning meaningful representations of an entire sentence or a short text. A Robustly Optimized BERT Pretraining Approach (RoBERTa) developed by Facebook AI is an enhanced version of the original BERT model with various modifications and optimizations. DistilRoBERTa is a distilled and smaller version of the RoBERTa model, which is designed for faster inference and reduced resource requirements while maintaining the performance of the original RoBERTa model. These models differ in their model size, raw texts used for pre-training, and performance. We test all these model variants in our analysis of the ACS occupational write-in data.

GPT Models Generative Pre-trained Transformer (GPT) is a decoder-only transformer model. During pre-training, GPT was trained on a huge amount of existing data to model the relationships between words for learning spelling, grammar, logic, and knowledge of human language without

any specific purposes. Unlike BERT, which uses a masked language modeling task and predicts masked words by looking at both the left and right context of a word, GPT uses an autoregressive language modeling task. It learns to predict the next word in a sequence given the preceding words, but it only looks at the left context (unidirectional). Notably, starting from GPT-3.5, GPT models incorporate a technique known as reinforcement learning from human feedback (RLHF). This enhancement allows the model to understand specific tasks without requiring explicit training examples. For instance, given a job description such as "direct the overall business activities for credit card company", we can instruct GPT by providing a "prompt" to classify it into Census and SOC codes. By refining and adapting these prompts, GPT progressively enhances its performance on the specified task, showcasing its remarkable flexibility and adaptability.

During fine-tuning, BERT updates the hyperparameters with the training dataset for a specific task. For text classification, BERT extracts features from the text entries, which are subsequently used as inputs in a standard classifier for text classification. To classify textual data, BERT accepts text input and tokenizes the words. For example, for the occupational write-in entry, "direct the overall business for credit card company", BERT tokenizer divides the full text into single words and special tokens indicating task types (e.g., [CLS] means a sentence level classification task), as well as sentence separations (i.e., [SEP]), are padded. Then, all the tokens are fed forward to the embedding layer and stack of encoders. The stack of encoders encodes all tokens to represent the input at different levels of abstraction for learning the semantic meanings of the input. In each layer of the underlying neural network, the encoder sees all tokens and only assigns weights to each element, in which way there is no vanishing gradient problem encountered by RNNs when the input sentence is too long. The final layers of the encoders are usually different summarizations of the input. Finally, the outputs from the stack of encoders are normalized and converted to output embedding. Subsequently, a classifier is used for classifying the converted output embeddings.[18]

T5 Models The Text-To-Text Transfer Transformer (T5) is an encoder-decoder transformer language model, which reframes all NLP tasks to a unified text-to-text format (Raffel et al. 2020).

[18]We use GPT-3.5-turbo-1106 (released in November 2023) for model estimation and fine-tuning. GPT-4 is a more powerful model yet does not support fine-tuning for a specific customized task, meaning that there is no possibility to update the values of hyperparameters within the model directly.

With this unified framework, T5 allows both the input and output to be text strings. Moreover, the unified text-to-text framework suggests that the same model, loss function, and hyperparameters can be used on any NLP task. T5 model was pre-trained in a teacher forcing style, which means that during the pre-training process, the model is provided with both input text sequences and their corresponding target sequences. The term "teacher forcing" comes from the idea that during pretraining, the model is provided with the correct or "teacher" output (target sequence) for a given input, and it learns to generate that output. This approach can expedite the training process and help the model acquire the ability to generate meaningful and contextually appropriate responses. In contrast to BERT, which is well-suited for tasks requiring contextual understanding within sentences, and GPT, which is known for text generation capabilities, T5 is often considered flexible and adaptable for a wide range of text transformation tasks.

After pretraining with teacher forcing, T5 can be fine-tuned on specific downstream tasks by providing task-specific input-output pairs, further customizing its behavior for a variety of natural language understanding and generation tasks. Fine-tuned LAnguage Net T5 (FLAN-T5), an enhanced version of T5, is the most widely used variant of T5. FLAN-T5 is fine-tuned with instructions with chain-of-thought, which improves the language model's performance in a mixture of NLP tasks (Chung et al. 2022).

5 Fine-tuning Transformer-Based Models for Occupational Coding

Transformer-based language models are typically pre-trained on large-scale existing corpora to learn semantic relationships between words in a pre-existing collection of data. Pre-trained language models can be fine-tuned with a new dataset for a specific task. Fine-tuning a pre-trained model significantly improves performance in specific tasks and reduces computation costs. The fine-tuning is task-specific and may substantially improve the performance of the model. After experimenting with different strategies, we develop the following fine-tuning steps illustrated in Figure 3.

*** FIGURE 3 ABOUT HERE ***

We first preprocess occupation data from the Alphabetical Indexes of Industries and Occupations. The data contain occupational descriptions and corresponding 2018 U.S. Census occupation

codes. The preprocess involves tasks such as removing stopwords, dividing text into smaller units (tokens), reducing words to their base forms (lemmas), and expanding abbreviations, acronyms, and initialisms. Some entries also include industry information. To improve the performance of the neural network model, we concatenate the industry and the occupation text to form a single text.

We then pick a neural network model, such as T5, and use its encoder to convert the text occupation data to numerical vectors. Next, we train the neural network model by defining the number of layers, hidden units, and other hyperparameters. We fine-tune the model by minimizing the loss function and use the fine-tuned model to obtain the vector representations of industry titles and occupation titles in the 2018 Census classifications. These steps are described in Algorithm 1.

*** ALGORITHM 1 ABOUT HERE ***

For downstream tasks, such as coding free-text occupation data, we further fine-tune the pre-trained occupation autocoder developed from Algorithm 1. This process helps adapt the model to perform well on domain-specific tasks. Within our Algorithm 2, we employ a two-step approach. In the first step, we identify and select three potential industry codes and three potential occupational codes for each entry based on the merged text. This step provides us with multiple candidate codes to consider for each aspect. Specifically, we use the neural network model to calculate each merged text entry's cosine similarity score (equation 5) with all available industry and occupation titles in 2018. Then we sort the similarity scores from highest to lowest by industry and occupation, respectively, to select the three industry and occupation titles that rank the highest. Note that this step involves sequence generation, as we need to convert the vector representation of industries and occupations back to their titles.

Next, we look up industry restriction rules in occupation titles described in the 2018 *Alphabetical Indexes of Industries and Occupations*. The restriction rules dictate that certain occupations are restricted to a specific industry. In the presence of industry information, certain occupation codes must or must not be used. If there are no restrictions specified for a particular occupation, we choose the first occupational code from the respective lists as our output. This straightforward selection process is employed when the occupation is not constrained by a specific industry context.

*** TABLE 1 ABOUT HERE ***

However, in cases where there are industry restrictions for the occupation, we implement an additional filtering mechanism. This filter is designed to examine the code lists and identify the most appropriate output result that satisfies the given industry restrictions. This ensures that the final output is in compliance with any specified constraints or dependencies between industry and occupation codes, resulting in a more accurate and context-aware coding output. There are seven different restriction rules described in Table 1. The AutoCoder will return a single occupation code with the best fit for each occupation (and industry) write-in entry. Algorithm 2 describes the procedures of coding occupation data using our fine-tuned neural network model.

*** ALGORITHM 2 ABOUT HERE ***

6 Results

6.1 Training Data: Alphabetical Indexes of Industries and Occupations

The *Alphabetical Indexes of Industries and Occupations* were developed primarily for use as an aid in classifying a respondent's industry (i.e., employer's type of business) and occupation (i.e., employee's type of work) as reported in demographic surveys conducted by the U.S. Census Bureau. The 2019 *Alphabetical Indexes* list over 22,000 industry descriptions and 32,000 job titles in alphabetical order. These are comprehensive lists of specific industries and occupations developed over time and continuously updated through review of the American Community Survey, decennial censuses, and other survey responses.

The industry index lists potential industry descriptions with their associated 4-digit Census Industry Code and the corresponding NAICS code. The occupation index contains a list of job titles and their equivalent 4-digit Census Occupation code and SOC code. The occupation index also provides guidance on when a class of work, education, or industry restriction should be applied. For example, the job title "letter carrier" is coded to Census Occupation code 5550 (postal services mail carries) only if the corresponding industry response is coded to Census Industry code 6370 (Postal Service).

6.2 Testing Data: ACS Occupation Write-Ins

We use a unique public use sample of occupation write-ins from the 2019 American Community Survey (ACS) Industry and Occupation write-in file.[19] The ACS is an ongoing survey conducted by the U.S. Census Bureau, collecting detailed demographic, social, economic, and housing data from a sample of households and individuals across the country throughout the year. Unlike the decennial census, the ACS provides current and frequent information about the American population, using sampled households to estimate characteristics for the entire nation. The surveys gather comprehensive information on age, race, education, income, occupation, industry, class of worker, housing, and more, producing data at various geographic levels.

The public use write-in file includes occupation and industry details from a selection of the American Community Survey (ACS) 2019 entries. The file offers users insight into the diverse raw responses (write-ins) from ACS participants and the corresponding 4-digit Census 2018 occupation and 2017 industry codes assigned by U.S. Census Bureau clerical coders. These codes align with the SOC 2018 and North American Industry Classification (NAICS) 2017 standards. Results in this paper rely solely on the write-in information of occupations and industries. However, when assigning occupational codes, the Census Bureau's clerical coders have access to more comprehensive data than this file reveals, including employer name, job roles, education, sex, and age.

Originally, the dataset contained 11,000 randomly selected entries with a write-in response for occupation title, job duties, and industry type. Entries that could reveal personally identifiable information (PII) were excluded. The most common type of PII removed was personal names or the name or location of a business. Additionally, cases with nonsensical or clear refusals such as "don't know", "D", "R", or expletives were removed. After applying the above steps, the final count of records is 10,499.

[19]The ACS Industry and Occupation write-in files are only available to special sworn staff at the U.S. Census Bureau. The file is used internally for training, quality assurance, and research. A public use sample was published in July 2022 to be used by external researchers and to help illustrate the coding process.

6.3 The Performance of Different Coding Methods

Our analyses rely on existing occupational autocoder methods, heuristic language models, probabilistic language models, and neural network language models to conduct occupational coding of the 2019 ACS Industry and Occupation write-in file. Based on the coding results, we calculate the accuracy, precision, recall, and F1 scores for each corresponding model. The accuracy rate evaluates how well the model can assign free text entries to correct Census occupation codes. The precision rate shows how often the model is correct. The recall rate measures if the autocoder can correctly find all entries belonging to a job title assigned by clerical coders. F1 scores combine both recall and precision measures to determine the effectiveness of the model. The mathematical definitions of these measures are shown below.

$$\text{Accuracy:} \quad \frac{\text{True Positives} + \text{True Negatives}}{\text{True Positives} + \text{True Negatives} + \text{False Positives} + \text{False Negatives}} \tag{8}$$

$$\text{Precision:} \quad \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{9}$$

$$\text{Recall:} \quad \frac{\text{True Positives}}{\text{True Positives} + \text{False Negative}} \tag{10}$$

$$\text{F1 Score:} \quad \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \tag{11}$$

We calculate accuracy, precision, recall, and the F1 score individually for each occupation. Then, to consider the differing sizes of the workforce across occupations, we aggregate each measure across occupations into a single metric through a weighted average, with the weights corresponding to the number of workers in each occupation. We present results for these four metrics in Table 2. In the ensuing discussion, we focus on model accuracy rates to evaluate overall model performance.

Table 2 shows several important findings. First, the NIOCCS AutoCoder has the highest accuracy rate out of the existing occupational autocoding methods. The NIOCCS AutoCoder reached the highest accuracy rate of 61.82% when using job title information and NAICS industry code data.[20] By contrast, the highest accuracy rates for the O*NET-SOC AutoCoder and SOCcer

[20]When including only job title and industry information, the NIOCCS results have higher accuracy (68.2%) compared to the results for inputs containing job titles, job descriptions, and industry codes. A possible reason is that the NIOCCS algorithm yields a higher overall probability of word match with fewer tokens. Therefore, adding more job information may lead to more coding errors in this algorithm.

AutoCoder are 58.19% and 43.16%, respectively. Second, the coding accuracy rates are highest when the coding input includes both industry and job title information. For example, the O*NET-SOC AutoCoder's accuracy improves from 54.68% with just job title information to 58.19% when job title, description, and industry code are all included in the input data.[21] Third, for both the NIOCCS AutoCoder and the SOCcer AutoCoder, including job descriptions does not markedly enhance accuracy compared to using job titles alone; however, incorporating industry information significantly increases the accuracy rate.

*** TABLE 2 ABOUT HERE ***

Next we compare the performance of different natural language models discussed in Section 4. We incorporate the fine-tuning procedures discussed in Section 5 into the transformer-based models. Table 3 shows that fuzzy matching, a heuristic language model, and conditional random fields model, a probabilistic language model, produce similar accuracy rates (32.70% and 31.76%, respectively). The fuzzy matching probabilistic language model is similar to the word matching methods currently used by the Census autocoder and the NIOCCS AutoCoder. Thus, we expected higher accuracy results of the fuzzy matching model due to its similarity with the Census autocoder. The topic modeling approach yielded a low accuracy rate of 9%. This limitation is likely due to the inherent challenges of topic models when the number of latent groups (in this case, several hundred) is very high.

Among the neural network models in Table 3, transformer-based models outperform CNNs and RNNs in accuracy rates and other evaluation metrics. This superior performance is attributable to the transformer models' bidirectional comprehension of text, their deep understanding of semantics and logic, and the use of extensive pretraining datasets. In fact, their occupational classification accuracy rates surpass not only those of other neural network models, probabilistic language models, and heuristic language models in Table 3 but also existing autocoders shown in Table 2. In particular, the FLAN-T5 model, which is a fine-tuned version of the original Text-to-Text Transfer Transformer Language Model, has the best overall performance, with an accuracy rate of 71.13%. We refer to our fine-tuned FLAN-T5 as T5-OCC (T5-based occupational classifier) in the following

[21]The O*NET-SOC autocoder does not allow users to include industry descriptions.

discussion.

*** TABLE 3 ABOUT HERE ***

We further test whether the accuracy rates vary across different occupational groups. For the sake of simplicity, we consider 23 major occupational groups defined in the Census Bureau's 2018 occupational classifications.[22] Table 4 reveals that T5-OCC is most reliable in legal, personal care and services, business and financial operations, and healthcare-related occupations and the least reliable in production occupations and architecture and engineering occupations. We present coding results across major occupational groups for other existing autocoders in Appendix Tables A1-A3. Similar to the T5-OCC model results, the O*NET-SOC, SOCcer, and NIOCCS AutoCoders consistently produced high accuracy rates for legal occupations. Occupational groupings with low accuracy rates varied per model. For example, the O*NET-SOC AutoCoder had low accuracy rates for management occupations and occupations belonging to the life, physical, and social science grouping. Results from the SOCcer AutoCoder indicate the majority of assigned SOC-2010 codes for computer and mathematical occupations do not crosswalk to a singular SOC-2018 code, resulting in frequent misclassification. Thus, the SOCcer AutoCoder had low accuracy rates among the computer and mathematical occupations grouping. Similar to the O*NET AutoCoder results, the NIOCCS AutoCoder also produced low accuracy rates for occupations belonging to the life, physical, and social science grouping. Our T5-OCC consistently outperforms these other autocoders in most of these major occupational groups. To further improve the performance of our model, a possible future direction is to develop separate neural network language models for coding occupations that belong to different major groups.

*** TABLE 4 ABOUT HERE ***

7 Conclusion

The present study develops a new model for coding occupations using job descriptions based on the T5 deep learning architecture. We have enhanced the T5 algorithm by fine-tuning the model

[22] More information on the Census 2018 occupation codes can be found on the Census Bureau's website https://www.census.gov/topics/employment/industry-occupation/guidance/code-lists.html.

with occupational descriptions and industry restrictions sourced from the Alphabetic Indexes of Industries and Occupations and coding guidelines adopted by the Census Bureau's trained clerical coding staff. The T5-OCC model, in contrast to existing AutoCoders, can accurately code on average 71% of raw write-in data into standard Census occupational categories. Unlike existing autocoding tools, such as O*NET-AutoCoder, which allow only one entry at a time, our method has the capability to process multiple entries in batches and efficiently manage substantial volumes of data and research requests. This versatility makes it a valuable asset for a wide range of research purposes.

Given the fast development of neural language models, it is likely that the performance of autocoding models can be further improved in future research through the integration of newer models. We offer several takeaways from our research and suggestions for future work on this topic. First, autocoders developed from neural language models do not necessarily perform better than those based on heuristic language models or probabilistic language models. For example, neural models, such as GPT, have lower accuracy rates than fuzzy matching and conditional random fields models. The reason is that the GPT model may produce varied occupational code output to the same write-in input at different instances. This problem arises from the randomness in the generation process introduced by the algorithm to avoid repetition. Second, fine-tuning with high-quality human coding data is important for improving the accuracy and external validity of autocoding models. For example, we find that the accuracy rates of O*NET-SOC AutoCoder and NIOCCS AutoCoder are significantly higher than SOCcer AutoCoder as the SOCcer relies on job descriptions obtained from a health survey rather than a nationally representative survey dataset. Most neural language models tend to perform poorly if they are not fine-tuned. The fine-tuning procedures developed in Figure 3 can be used as a guideline for future research. Third, the accuracy of autocoders is contingent on the type of information they use: whether they rely solely on job titles, or if they combine job titles with detailed job and industry descriptions and other relevant information about workers. Clerical coders working with ACS and CPS data typically use job titles, job descriptions, industry descriptions, and the educational background of workers. In our analysis of the ACS write-in data, our sample does not contain any educational information. Often the more comprehensive the information is used, the greater the accuracy in coding. Finally, the level

of coding accuracy we observe is not uniform across major occupational categories. This discrepancy indicates that the effectiveness of a single coding model can vary significantly depending on the specific occupational group being coded. As a result, there may be a need for deploying different models tailored to the unique characteristics and requirements of each occupational group in the future.

Algorithm 1 Using Language Models to Pre-train an Occupation AutoCoder

Input: Occupational descriptions $\mathbf{X}$ and corresponding 2018 Census occupation codes $\mathbf{W}$ in the Alphabetical Indexes (2019); 2018 Census occupation titles OCC and 2017 industry titles IND

Output: A fine-tuned language model $g_{\boldsymbol{\Theta}}$ and vector representations of 2018 Census occupation titles and 2017 industry titles OCC_e and IND_e

Initiate/Load a pre-trained neural network language model f with hyperparameters $\boldsymbol{\Theta}$

Pre-processing:

$\mathbf{X}' = \text{preprocessing}(\mathbf{X})$ ▷ e.g., removing stopwords and handling abbreviations

Transformer encoder:

$\mathbf{Y} = \text{encoder}(\mathbf{X}')$ ▷ convert text data to numerical vectors

$\mathbf{Y}' = f_{\boldsymbol{\Theta}}(\mathbf{Y})$ ▷ obtain the output representation vector $\mathbf{Y}'$

Fine-tuning by minimizing the loss function:

$\text{argmin}_{\boldsymbol{\Theta},\mathbf{W}} \mathcal{L}(\boldsymbol{\Theta}, \mathbf{W}; \mathbf{Y}')$ ▷ $\mathbf{W}$ is the assigned 2018 Census OCC codes

Return $g \leftarrow$ Fine-tuned model f with updated hyperparameters

Vector representation of Census industry and occupation titles:

$\text{IND}' = \text{encoder}(\text{IND})$

$\text{IND}_e = g_{\boldsymbol{\Theta}}(\text{IND}')$

$\text{OCC}' = \text{encoder}(\text{OCC})$

$\text{OCC}_e = g_{\boldsymbol{\Theta}}(\text{OCC}')$

Algorithm 2 Using the Pretrained Occupation AutoCoder to Code Occupation Write-in Data

Input: Occupation (and industry) write-ins $\mathbf{X}$
Output: 2018 Census occupation code OCC

Pre-processing:
$\mathbf{X}' = \text{preprocessing}(\mathbf{X})$ ▷ e.g., removing stopwords and handling abbreviations

Transformer encoder:
$\mathbf{Y} = \text{encoder}(\mathbf{X}')$ ▷ convert text data to numerical vectors
$\mathbf{Y}' = g_{\mathbf{\Theta}}(\mathbf{Y})$ ▷ obtain the output representation vector $\mathbf{Y}'$ using the autocoder in Algorithm 1

Similarity score calculations:
for vector $\mathbf{v}$ in IND_e **do** ▷ IND_e is vector representations of all Census industry titles
  $\cos(\mathbf{Y}', \mathbf{v}) = \frac{\mathbf{Y}' \cdot \mathbf{v}}{||\mathbf{Y}|| \cdot ||\mathbf{v}||}$
end for
sort $\cos(\mathbf{Y}', \mathbf{v})$
return IND_1, IND_2, IND_3 ▷ retrieve the top three similar industry titles

for vector $\mathbf{u}$ in OCC_e **do** ▷ OCC_e is vector representations of all Census occupation titles
  $\cos(\mathbf{Y}', \mathbf{u}) = \frac{\mathbf{Y}' \cdot \mathbf{u}}{||\mathbf{Y}|| \cdot ||\mathbf{u}||}$
end for
sort $\cos(\mathbf{Y}', \mathbf{u})$
return OCC_1, OCC_2, OCC_3 ▷ retrieve the top three similar occupation titles

Occupation list refinement:
$n = 1$
while $n \leq 3$ **do** ▷ apply industry restriction rules $\mathbf{\Omega}$ defined in the Alphabetical Indexes
  if $\text{OCC}_n \notin \mathbf{\Omega}$ **then**
    $\text{OCC} = \text{OCC}_n$
    break
    else
    $n = n + 1$
  end if
end while

Table 1. Summary of Industry Restrictions in Occupation Titles

Format Examples	Meaning
1070	When this is the industry code, assign this occupation code
(1070)	This industry code is suggested for this occupation code
#1070	This industry code is mandatory for this occupation code
1070, 1080	If either of these is the industry code, assign this occupation code
1070-1390	If industry code is in this range, assign this occupation code
exc 1070	If this is not the industry code, assign this occupation code
\Any not listed	Check the other index lines; if none apply, assign this occupation code

Data source: The Alphabetical Indexes of Industries and Occupations (2019).

Table 2. Evaluation Metrics of Existing Autocoders in Coding Occupational Write-in Data

	Input Data Include				Overall (%)			
	Job Title	Job Description	Industry Code	Industry Description	Accuracy	Precision	Recall	F1 Score
O*NET-SOC AutoCoder	✓				54.68	60.33	54.71	54.11
(version 14.1)	✓	✓			57.21	62.97	57.21	56.94
	✓	✓	✓(2-4 NAICS)		58.19	66.16	58.19	57.03
NIOCCS AutoCoder	✓				59.58	71.04	59.58	61.85
(version v4)	✓	✓			55.18	62.96	55.18	56.26
	✓	✓	✓(2-6 NAICS)		60.98	66.67	60.98	61.56
	✓	✓		✓	61.82	66.70	61.82	62.42
SOCcer AutoCoder	✓				36.32	66.16	57.17	57.03
(version v2)	✓	✓			36.53	66.16	57.17	57.03
	✓	✓	✓(SIC)		43.16	47.72	43.16	43.57

Data sources: The Public-Use Sample of Occupation and Industry Write-ins from the American Community Survey, 2019.
Notes: The testing sample contains 10,449 occupation and industry write-in entries, randomly selected from the 2019 American Community Survey (ACS). We report each autocoder's overall Accuracy, Precision, Recall, and F1 score. Accuracy is the number of correctly classified occupational entries divided by the total number of entries in the testing sample. The overall Precision, Recall, and F1 score are the means of the Precision, Recall, and F1 score in the 568 Census 2018 occupational categories. For each category, Precision equals the number of write-in entries that are correctly classified into that category divided by the number of write-in entries classified into that category by the autocoder; Recall equals the number of write-in entries that are correctly classified into that category divided by the number of write-in entries classified into that category by ACS clerical coders; and the F1 score equals the harmonic mean of Precision and Recall, F1 score = 2 * (Precision * Recall) / (Precision + Recall). The NIOCCS AutoCoder allows the inclusion of industry descriptions along with job titles and job descriptions. In contrast, the O*NET-SOC AutoCoder and the SOCcer AutoCoder restrict users to input industry codes in either the North American Industry Classification System (NAICS) or Standard Industry Classification (SIC) formats.

Table 3. Evaluation Metrics of Different Language Models in Coding Occupational Write-in Data

	Input Data Include			Overall (%)			
	Job Title	Job Description	Industry Description	Accuracy	Precision	Recall	F1 Score
Heuristic Language Models							
Fuzzy Matching	✓	✓	✓	32.70	45.42	32.70	33.19
Probablistic Language Models							
Conditional Random Fields Model	✓	✓	✓	31.76	44.56	31.76	32.15
Topic Modeling	✓	✓	✓	9.00	7.00	8.63	6.63
Neural Network Models							
CNNs	✓	✓	✓	24.40	36.11	24.40	26.67
RNNs	✓	✓	✓	33.06	45.46	33.06	32.97
Transformer-Based Models							
BERT	✓	✓	✓	52.31	63.24	52.31	54.44
sBERT	✓	✓	✓	53.22	64.16	53.22	54.89
RoBERTa	✓	✓	✓	51.90	63.47	51.90	53.90
GPT-3.5	✓	✓	✓	$\sim$ 19*	-	-	-
GPT-4	✓	✓	✓	$\sim$ 19*	-	-	-
T5	✓	✓	✓	61.74	68.94	61.74	62.36
FLAN-T5 ("T5-OCC")	✓	✓	✓	71.13	78.63	71.13	69.57

Data sources: The Public-Use Sample of Occupation and Industry Write-ins from the American Community Survey, 2019.

Notes: The testing sample contains 10,449 occupation and industry write-in entries, randomly selected from the 2019 American Community Survey (ACS). We report each autocoder's overall Accuracy, Precision, Recall, and F1 score. Accuracy is the number of correctly classified occupational entries divided by the total number of entries in the testing sample. The overall Precision, Recall, and F1 score are the means of the Precision, Recall, and F1 score in the 568 Census 2018 occupational categories. For each category, Precision equals the number of write-in entries that are correctly classified into that category divided by the number of write-in entries classified into that category by the autocoder; Recall equals the number of write-in entries that are correctly classified into that category divided by the number of write-in entries classified into that category by ACS clerical coders; and the F1 score equals the harmonic mean of Precision and Recall, F1 score = 2 * (Precision * Recall) / (Precision + Recall). We do not obtain consistent evaluation metrics for the GPT models as the models might generate different occupational code outputs for the same write-in input on different occasions. This issue stems from the algorithm's intentional introduction of randomness in the generation process to prevent repetition.

Table 4. Evaluation Metrics of The Text-To-Text Transfer Transformer Model for Occupations (T5-OCC) Across Major Occupational Groups

Census 2018 Occupation Groups	# Occupations within Each Group	Overall (%)			
		Accuracy	Precision	Recall	F1 Score
Management	33	59.38	75.57	59.38	61.94
Business and Financial Operations	29	71.15	87.68	71.15	75.96
Computer and Mathematical Occupations	18	46.58	59.06	46.58	46.91
Architecture and Engineering	24	40.91	53.65	40.91	43.97
Life, Physical, and Social Science	27	47.62	71.43	47.62	52.86
Community and Social Services	16	63.64	77.71	63.64	67.32
Legal Occupations	6	86.89	95.83	86.89	85.65
Education, Training, and Library	12	70.44	83.37	70.44	74.91
Arts, Design, Entertainment, Sports, and Media	30	69.60	80.74	59.69	67.39
Healthcare Practitioners and Technical Occupations	46	74.65	88.89	74.65	79.11
Healthcare Support	14	67.80	87.15	67.80	74.50
Protective Service	20	72.34	83.33	72.34	76.91
Food Preparation and Serving Related	12	62.62	78.67	62.62	67.64
Building and Ground Cleaning and Maintenance	8	64.79	95.77	64.79	75.44
Personal Care and Service	20	58.82	78.52	58.82	64.56
Sales and Related Occupations	18	59.13	74.71	59.13	64.63
Office and Administrative Support	54	44.95	71.74	44.95	62.73
Farming, Fishing, and Forestry	8	54.00	83.33	54.00	61.67
Construction and Extraction	37	63.74	84.51	63.74	71.16
Installation, Maintenance, and Repair	36	43.64	68.18	43.64	49.64
Production Occupations	63	37.95	58.26	37.95	47.80
Transportation and Material Moving Occupations	33	61.54	86.33	61.54	69.27
Military Specific Occupations	4	31.62	75.52	31.62	35.65

Data sources: The Public-Use Sample of Occupation and Industry Write-ins from the American Community Survey, 2019.
Notes: Our input data include job titles, job descriptions, and industry descriptions. The T5 model used is the base version model. It is the same model shown in Table 3, except that we calculate the evaluation metrics by major occupation groups.

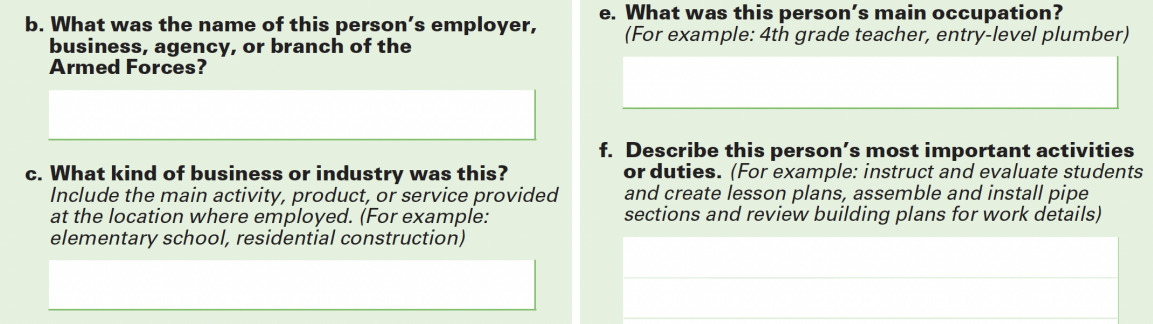

Figure 1. Excerpt of Industry and Occupation Questions from the ACS Questionnaire

Data sources: American Community Survey, 2019.
Notes: The figure shows questions related to occupations and industry in Section 42 on Description of Employment in the American Community Survey. For these questions, respondents were asked to describe the business, industry, or individual employer, job title, and task descriptions for the major activity at the place where they worked. For the full questionnaire, please refer to Sample ACS Forms and Instructions on the U.S. Census Bureau's website.

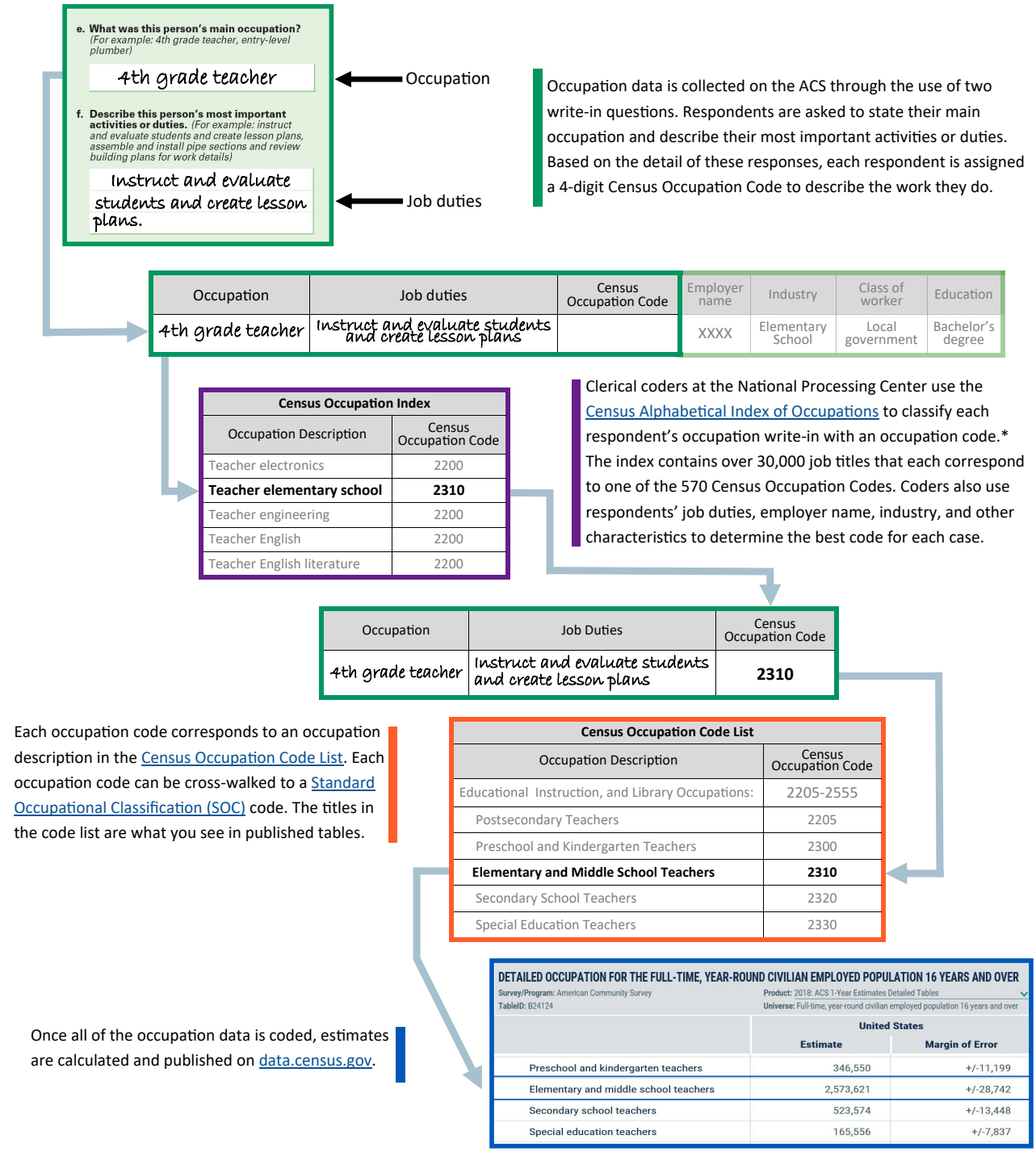

Figure 2. The Procedures of Occupation Data Collection and Coding in the ACS

Data sources: American Community Survey, 2019.
Notes: This figure illustrates the process of manually coding occupation write-ins for the ACS. In 2012, the Census Bureau introduced an autocoder for industry and occupation to reduce the burden of manual coding. During the processing phase, around 41% of the cases receive an automated occupation code before undergoing manual clerical coding.

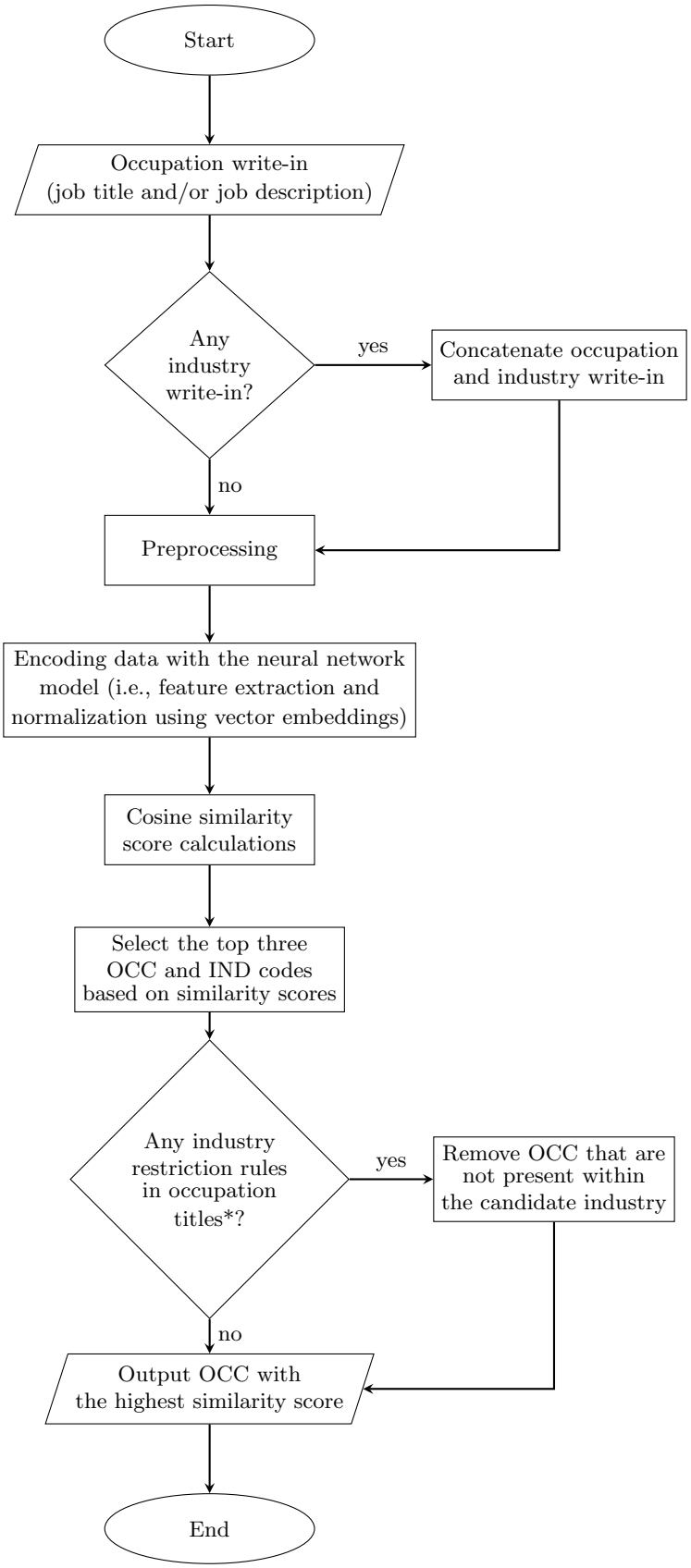

Figure 3. The Algorithm Flowchart Illustrating the Fine-tuning Process of Converting Write-in Data to Occupational Codes

Notes: We adopt industry restriction rules defined in the Alphabetical Indexes of Occupations and Industries (U.S. Census Bureau 2019). These rules state that certain occupations are restricted to specific industries, and no other combination is possible. Therefore, the occupational coding depends on whether industry information is provided (see Table 1).

References

Abbott, Andrew. 1995. "Sequence Analysis: New Methods for Old Ideas." *Annual Review of Sociology* 21:93–113.

Apostolico, Alberto and Zvi Galil. 1997. *Pattern Matching Algorithms*. New York, NY: Oxford University Press.

Blei, David M. 2012. "Probabilistic Topic Models." *Communications of the ACM* 55:77–84.

Blei, David M, Andrew Y Ng, and Michael I Jordan. 2003. "Latent Dirichlet Allocation." *Journal of Machine Learning Research* 3:993–1022.

Cain, Pamela S. and Donald J. Treiman. 1981. "The Dictionary of Occupational Titles as a Source of Occupational Data." *American Sociological Review* 46:253–278.

Census Bureau. 2015. "Using an Autocoder to Code Industry and Occupation in the American Community Survey." Working paper, U.S. Census Bureau https://www2.census.gov/library/working-papers/2015/demo/2015-Day-Chiu-Montalvo-01.pdf (accessed September 13, 2023).

Census Bureau. 2019. "General Overview of the Alphabetical Indexes of Industries and Occupations." Working paper, U.S. Census Bureau https://www2.census.gov/programs-surveys/demo/guidance/industry-occupation/overview2019.pdf (accessed October 17, 2023).

Chung, Hyung Won, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2022. "Scaling Instruction-Finetuned Language Models." *arXiv preprint arXiv:2210.11416* .

Cohen, William W., Pradeep Ravikumar, and Stephen E. Fienberg. 2003. "A Comparison of String Distance Metrics for Name-Matching Tasks." In *IIWeb*, volume 3, pp. 73–78.

de Leeuw, Tim and Steffen Keijl. 2023. "Combining Multiple Organizational-level Databases: An Empirical Evaluation of Different Matching Methods." *Sociological Methods & Research* 52:268–298.

De Matteis, S, D Jarvis, H Young, A Young, N Allen, J Potts, A Darnton, L Rushton, and P Cullinan. 2017. "Occupational Self Coding and Automatic Recording (OSCAR): A Novel Efficient Web-Based Tool to Collect and Code Lifetime Job-Histories in Large Population-Based Studies." *Scandinavian Journal of Work, Environment and Health* 43.

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. "BERT: Pre-training of deep bidirectional transformers for language understanding." *arXiv preprint arXiv:1810.04805* .

Gentzkow, Matthew, Bryan Kelly, and Matt Taddy. 2019. "Text as Data." *Journal of Economic Literature* 57:535–574.

Grimmer, Justin, Margaret E Roberts, and Brandon M Stewart. 2022. *Text as Data: A New Framework for Machine Learning and the Social Sciences*. Princeton University Press.

Gusfield, Dan. 1997. *Algorithms on Stings, Trees, and Sequences: Computer Science and Computational Biology*. Cambridge, UK: Cambridge University Press.

Gweon, Hyukjun, Matthias Schonlau, Lars Kaczmirek, Michael Blohm, and Stefan Steiner. 2017. "Three Methods for Occupation Coding Based on Statistical Learning." *Journal of Official Statistics* 33:101–122.

Heiberger, Raphael H, Sebastian Munoz-Najar Galvez, and Daniel A McFarland. 2021. "Facets of Specialization and Its Relation to Career Success: An Analysis of US Sociology, 1980 to 2015." *American Sociological Review* 86:1164–1192.

Kim, Yoon. 2014. "Convolutional Neural Networks for Sentence Classification." *arXiv preprint arXiv:1408.5882* .

Lafferty, John D., Andrew McCallum, and Fernando C. N. Pereira. 2001. "Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data." In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pp. 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Lesnard, Laurent. 2010. "Setting Cost in Optimal Matching to Uncover Contemporaneous Socio-temporal Patterns." *Sociological Methods & Research* 38:389–419.

Levenshtein, Vladimir I. 1966. "Binary Codes Capable of Correcting Deletions, Insertions, and Reversals." In *Soviet Physics Doklady*, volume 10, pp. 707–710. Soviet Union.

Martin-Caughey, Ananda. 2021. "What's in an Occupation? Investigating Within-Occupation Variation and Gender Segregation Using Job Titles and Task Descriptions." *American Sociological Review* 86:960–999.

Medsker, Larry R and LC Jain. 2001. "Recurrent Neural Networks." *Design and Applications* 5:2.

Miller, Ann R., Donald J. Treiman, Pamela Cain, and Patricia A. Roos. 1980. *Work, Jobs, and Occupations: A Critical Review of the Dictionary of Occupational Titles*. Washington, D.C.: National Academies Press.

NIOSH. 2010. "System Design Document Part B. Technical System Design for the NIOSH Industry and Occupation Computerized Coding System Version 1.1." Centers for Disease Control and Prevention, National Institute for Occupational Safety and Health, https://csams.cdc.gov/nioccs/ (accessed September 28, 2023).

NIOSH. 2018. "NIOSH Industry and Occupation Computerized Coding System (NIOCCS) V3.0." Centers for Disease Control and Prevention, National Institute for Occupational Safety and Health, https://www.cdc.gov/niosh/topics/coding/pdfs/NIOCCS_V3_User_Manual.pdf (accessed October 20, 2023).

NIOSH. 2023. "NIOSH Industry and Occupation Computerized Coding System (NIOCCS)." U.S. Department of Health and Human Services, Public Health Service, Centers for Disease Control and Prevention, National Institute for Occupational Safety and Health, Division of Field Studies & Engineering, Health Informatics Branch, https://csams.cdc.gov/nioccs/ (accessed September 28, 2023).

O*NET. 2023. "O*NET SOC Autocoder v14.1." Frequently Asked Questions, https://www.onetsocautocoder.com/plus/onetmatch?action=guide (accessed September 29, 2023).

Raffel, Colin, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer." *The Journal of Machine Learning Research* 21:5485–5551.

Rémen, Thomas, Lesley Richardson, Corinne Pilorget, Gilles Palmer, Jack Siemiatycki, and Jérôme Lavoué. 2018. "Development of a Coding and Crosswalk Tool for Occupations and Industries." *Annals of Work Exposures and Health* 62:796–807.

Roberts, Benjamin, Abas Shkembi, Lauren M Smith, and Richard L Neitzel. 2022. "Beware the Grizzlyman: A Comparison of Job- and Industry-Based Noise Exposure Estimates Using Manual Coding and the NIOSH NIOCCS Machine Learning Algorithm." *Journal of Occupational and Environmental Hygiene* 19:437–447.

Russ, Daniel E, Kwan-Yuet Ho, Joanne S Colt, Karla R Armenti, Dalsu Baris, Wong-Ho Chow, Faith Davis, Alison Johnson, Mark P Purdue, Margaret R Karagas, Kendra Schwartz, Molly Schwenn, Debra T Silverman, Calvin A Johnson, and Melissa C Friesen. 2016. "Computer-Based Coding of Free-Text Job Descriptions to Efficiently Identify Occupations in Epidemiological Studies." *Occupational and Environmental Medicine* 73:417–424.

Russ, Daniel E, Pabitra Josse, Thomas Remen, Jonathan N Hofmann, Mark P Purdue, Jack Siemiatycki, Debra T Silverman, Yawei Zhang, Jerome Lavoué, and Melissa C Friesen. 2023. "Evaluation of the Updated SOCcer V2 Algorithm for Coding Free-Text Job Descriptions in Three Epidemiologic Studies." *Annals of Work Exposures and Health* p. wxad020.

Sankoff, David and Joseph B Kruskal. 1983. *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*. Reading: Addison-Wesley Publication.

Savic, Nenad, Nicolas Bovio, Fabien Gilbert, José Paz, and Irina Guseva Canu. 2022. "Procode: A Machine-Learning Tool to Support (Re-)coding of Free-Texts of Occupations and Industries." *Annals of Work Exposures and Health* 66:113–118.

Schierholz, Malte, Miriam Gensicke, Nikolai Tschersich, and Frauke Kreuter. 2018. "Occupation

Coding During the Interview." *Journal of the Royal Statistical Society Series A: Statistics in Society* 181:379–407.

Schierholz, Malte and Matthias Schonlau. 2021. "Machine Learning for Occupation Coding—A Comparison Study." *Journal of Survey Statistics and Methodology* 9:1013–1034.

Thompson, Matthew, Michael E Kornbau, and Julie Vesely. 2012. "Creating an Automated Industry and Occupation Coding Process for the American Community Survey." http://ftp.census.gov/adrm/fesac/2014-06-13_thompson_kornbau_vesely.pdf. (accessed October 10, 2016).

Treiman, Donald J. 1979. *Job Evaluation: An Analytic Review*. Washington, D.C.: National Academies Press.

U.S. Census Bureau. 2019. "General Overview of the Alphabetical Indexes of Industries and Occupations." https://www2.census.gov/programs-surveys/demo/guidance/industry-occupation/overview2019.pdf. (accessed September 30, 2023).

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. "Attention Is All You Need." *Advances in Neural Information Processing Systems* 30.

Wagner, Robert A and Michael J Fischer. 1974. "The String-to-String Correction Problem." *Journal of the ACM* 21:168–173.

Wan, Wenxin, Calvin B Ge, Melissa C Friesen, Sarah J Locke, Daniel E Russ, Igor Burstyn, Christopher JO Baker, Anil Adisesh, Qing Lan, Nathaniel Rothman, et al. 2023. "Automated Coding of Job Descriptions from a General Population Study: Overview of Existing Tools, Their Application and Comparison." *Annals of Work Exposures and Health* 67:663–672.

Winkler, William E. 1990. "String Comparator Metrics and Enhanced Decision Rules in the Fellegi-Sunter Model of Record Linkage." Proceedings of the Section on Survey Research Methods, American Statistical Association.

Winkler, William E. 2006. "Overview of Record Linkage and Current Research Directions." Research Report Series (Statistics 2006-2), Washington, DC: U.S. Census Bureau Statistical Research Division.

Appendix

Table A1. O*NET-SOC AutoCoder Across Major Occupational Groups

Census 2018 Occupation Groups	# Occupations within Each Group	Overall (%)			
		Accuracy	Precision	Recall	F1 Score
Management	33	38.40	37.63	38.40	35.19
Business and Financial Operations	29	58.69	50.40	58.69	51.14
Computer and Mathematical Occupations	18	47.28	40.22	47.28	41.79
Architecture and Engineering	24	41.43	44.72	41.43	41.08
Life, Physical, and Social Science	27	38.18	37.24	38.18	34.33
Community and Social Services	16	42.35	50.99	42.35	43.68
Legal Occupations	6	81.15	71.72	81.15	75.49
Education, Training, and Library	12	67.17	61.57	67.17	64.05
Arts, Design, Entertainment, Sports, and Media	30	63.57	59.64	63.57	59.27
Healthcare Practitioners and Technical Occupations	46	82.55	81.48	82.55	81.37
Healthcare Support	14	72.58	62.53	72.58	66.82
Protective Service	20	76.25	69.07	76.25	71.93
Food Preparation and Serving Related	12	74.06	68.88	74.06	70.86
Building and Ground Cleaning and Maintenance	8	60.34	49.87	60.34	53.64
Personal Care and Service	20	68.03	54.74	68.03	59.93
Sales and Related Occupations	18	56.36	49.79	56.36	51.87
Office and Administrative Support	54	59.36	51.21	59.36	53.07
Farming, Fishing, and Forestry	8	63.86	45.53	63.86	52.86
Construction and Extraction	37	67.80	58.44	67.80	61.89
Installation, Maintenance, and Repair	36	56.14	51.98	56.14	52.67
Production Occupations	63	48.17	43.95	48.17	43.94
Transportation and Material Moving Occupations	33	64.01	57.30	64.01	59.35
Military Specific Occupations	4	12.50	25.00	12.50	16.67

Data sources: The Public-Use Sample of Occupation and Industry Write-ins from the American Community Survey, 2019.
Notes: Our input data include job titles, job descriptions, and industry descriptions. The O*NET-SOC model is the same model shown in Table 2, except that we calculate the evaluation metrics by major occupation groups.

Table A2. Evaluation Metrics of the Standardized Occupation Coding for Computer-assisted Epidemiologic Research (SOCcer) Across Major Occupational Groups

Census 2018 Occupation Groups	# Occupations within Each Group	Overall (%)			
		Accuracy	Precision	Recall	F1 Score
Management	33	31.67	34.29	31.67	27.99
Business and Financial Operations	29	53.11	39.76	53.11	44.56
Computer and Mathematical Occupations	18	1.36	.73	1.36	.91
Architecture and Engineering	24	30.48	22.77	30.48	24.52
Life, Physical, and Social Science	27	30.91	31.48	30.91	29.63
Community and Social Services	16	37.06	36.76	37.06	34.77
Legal Occupations	6	82.79	74.62	82.79	77.95
Education, Training, and Library	12	38.76	35.95	38.76	36.36
Arts, Design, Entertainment, Sports, and Media	30	56.88	52.42	56.88	50.91
Healthcare Practitioners and Technical Occupations	46	62.55	54.95	62.55	57.40
Healthcare Support	14	21.74	17.33	21.74	19.24
Protective Service	20	70.83	61.42	70.83	65.50
Food Preparation and Serving Related	12	62.78	58.52	62.78	59.70
Building and Ground Cleaning and Maintenance	8	56.66	42.24	56.66	46.52
Personal Care and Service	20	66.17	49.86	66.17	55.86
Sales and Related Occupations	18	48.75	41.47	48.75	43.33
Office and Administrative Support	54	52.31	38.28	52.31	43.23
Farming, Fishing, and Forestry	8	42.17	21.55	42.17	28.10
Construction and Extraction	37	60.34	45.01	60.34	50.18
Installation, Maintenance, and Repair	36	42.46	35.28	42.46	36.88
Production Occupations	63	42.06	32.23	42.06	35.03
Transportation and Material Moving Occupations	33	35.99	24.55	35.99	28.51
Military Specific Occupations	4	0.0	0.0	0.0	0.0

Data sources: The Public-Use Sample of Occupation and Industry Write-ins from the American Community Survey, 2019.
Notes: Our input data include job titles, job descriptions, and industry descriptions. The SOCcer model is the same model shown in Table 2, except that we calculate the evaluation metrics by major occupation groups.

Table A3. Evaluation Metrics of the NIOSH Industry and Occupation Computerized Coding System (NIOCCS) Across Major Occupational Groups

Census 2018 Occupation Groups	# Occupations within Each Group	Overall (%)			
		Accuracy	Precision	Recall	F1 Score
Management	33	48.95	74.82	48.95	56.73
Business and Financial Operations	29	63.77	86.28	63.77	72.42
Computer and Mathematical Occupations	18	56.46	66.38	56.46	60.28
Architecture and Engineering	24	44.29	69.17	44.29	52.41
Life, Physical, and Social Science	27	37.27	64.45	37.27	44.10
Community and Social Services	16	50.59	81.49	50.59	59.17
Legal Occupations	6	74.59	90.46	74.59	81.49
Education, Training, and Library	12	63.89	75.74	63.89	68.73
Arts, Design, Entertainment, Sports, and Media	30	64.31	84.01	64.31	71.52
Healthcare Practitioners and Technical Occupations	46	72.48	87.25	72.48	77.72
Healthcare Support	14	68.56	81.99	68.56	72.41
Protective Service	20	74.17	86.11	74.17	78.98
Food Preparation and Serving Related	12	65.79	78.02	65.79	70.61
Building and Ground Cleaning and Maintenance	8	78.19	90.22	78.19	83.69
Personal Care and Service	20	76.95	96.75	76.95	83.69
Sales and Related Occupations	18	60.40	80.90	60.40	67.37
Office and Administrative Support	54	57.26	75.19	57.26	62.67
Farming, Fishing, and Forestry	8	63.86	93.78	63.86	74.58
Construction and Extraction	37	70.15	85.76	70.15	76.43
Installation, Maintenance, and Repair	36	61.75	76.23	61.75	67.40
Production Occupations	63	52.01	67.20	52.01	55.88
Transportation and Material Moving Occupations	33	66.48	83.18	66.48	72.44
Military Specific Occupations	4	12.50	12.50	12.50	12.50

Data sources: The Public-Use Sample of Occupation and Industry Write-ins from the American Community Survey, 2019.
Notes: Our input data include job titles, job descriptions, and industry descriptions. The NIOCCS model is the same model shown in Table 2, except that we calculate the evaluation metrics by major occupation groups.