2020 No. 100

Streamlining the Identification of Emerging Tasks in the O*NET System Using Natural Language Processing (NLP): Technical Summary

Prepared for:	National Center for O*NET Development 313 Chapanoke Road, Suite 130 Raleigh, NC 27603	Prepared under:	Subcontract Number (through RTI International): 1-312-0207142--41224L
Authors:	Jeffrey A. Dahlke Dan J. Putka	Date:	February 18, 2021

Streamlining the Identification of Emerging Tasks in the O*NET System Using Natural Language Processing (NLP): Technical Summary

Table of Contents

Table of Contents (Continued)

List of Tables

List of Figures

Streamlining the Identification of Emerging Tasks in the O*NET System Using Natural Language Processing (NLP): Technical Summary

Introduction

Each year as part of the Human Resources Research Organization's (HumRRO) Occupational Information Network (O*NET) Emerging Tasks project, HumRRO analysts review write-in task statements obtained from job incumbents or occupational experts to determine if new tasks should be added to O*NET. Specifically, HumRRO analysts review and make several judgments regarding each new write-in statement to see if it (or some variation on it) warrants inclusion as a new task (or a revision to an existing task) for a given O*NET Standard Occupational Classification Code (O*NET-SOC). Once a new task is developed, it is included in future O*NET Program data collection to gather relevance, importance, and frequency information from job incumbents and occupation experts (OEs). "Emerging Tasks" are also published within the O*NET Database (e.g., National Center for O*NET Development 2020a; 2020c). The identification of emerging tasks augments other efforts by the O*NET Program to develop and maintain updated task information for O*NET-SOC occupations (Dierdorff & Norton, 2011; Green & Allen, 2020).

Current Process

For well over a decade, HumRRO has built a database of write-in task statements consisting of over 100,000 statements that have been labeled with respect to several key decisions analysts made about them during the review and rating process above, namely:

1. Is the write-in statement an acceptable task?

    a. Write-in statements can be classified as unacceptable for a variety of reasons. For example, a write-in statement would not be acceptable if it is not actually a statement, is unclear, is too broad, describes a non-task characteristic of the job (e.g., knowledges, skills, abilities, citizenship behaviors, job requirements, work environment), or voices a complaint.

    b. HumRRO analysts classify unacceptable tasks into several categories, but that information is for informational purposes only and not relied upon in any subsequent step in the review and rating process. Thus, the most critical judgment for modeling purposes is whether a given write-in statement reflects an acceptable task or not.

2. Is the acceptable task completely redundant with an existing task for a given O*NET-SOC occupation?

    a. Additionally, HumRRO analysts classify acceptable tasks into three categories with respect to their similarity to an existing task statement from a given O*NET-SOC: (a) completely redundant, (b) partially redundant, or (c) unique.

3. Is there a sufficient number of acceptable-yet-not-completely-redundant tasks that are sufficiently similar to one another to warrant creation of a new task or revision to an existing task that captures a common theme among the similar tasks?

We provide examples of decisions 1 and 2 in Appendix A. All of the decisions above are made via the first seven steps of the eight-step write-in task statement analysis process summarized in Figure 1 (see Appendix B for a more detailed process schematic). Separate decisions are made for each write-in task statement provided for each O*NET-SOC included in a given yearly review-and-rating cycle. For example, each year, HumRRO is furnished with a list of write-in task statements for about 100 occupations (roughly 25 to 130 statements per occupation, with a median of 60). HumRRO analysts are assigned to a given occupation and go through the seven steps to make the decisions listed above for each write-in task statement in the occupation to which they were assigned. All of this review is done manually, facilitated by information presented to an analyst in a Microsoft Access application and judgments entered into that application.

In light of advances in natural language processing (NLP) over the past decade, as well as the fact that HumRRO is in the unique and enviable state of having a fully labeled database of over 100,000 write-in task statements with respect to the judgments above, we are in an unparalleled position to streamline the process of identifying emerging tasks while also potentially improving the quality and reliability of analysts' decisions.

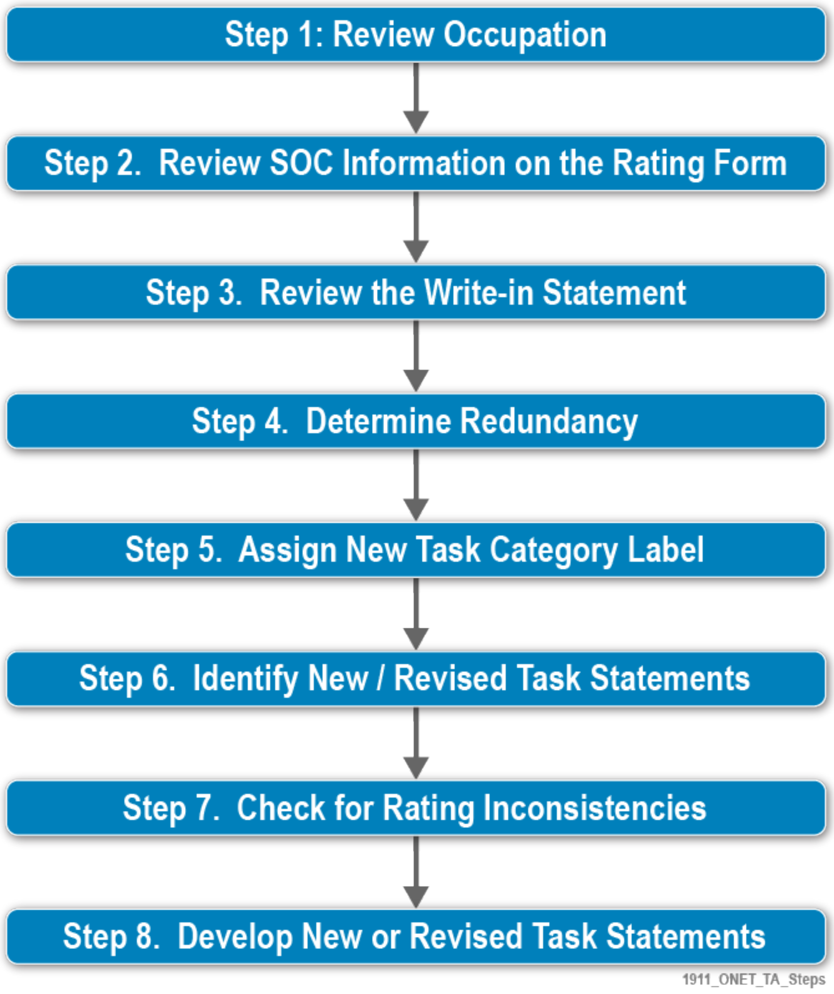

Step 1: Review Occupation
Step 2. Review SOC Information on the Rating Form
Step 3. Review the Write-in Statement
Step 4. Determine Redundancy
Step 5. Assign New Task Category Label
Step 6. Identify New / Revised Task Statements
Step 7. Check for Rating Inconsistencies
Step 8. Develop New or Revised Task Statements
1911_ONET_TA_Steps

Figure 1. Summary of current process for evaluating write-in statements.

Streamlining the Current Process Using NLP

In this work, we developed new solutions to answer three critical questions about write-in statements:

1. Is the write-in statement an **acceptable task**? (Step 3 of the current process)
2. Is the statement **completely redundant** with an existing task for a given O*NET-SOC? (Step 4 of the current process)
3. Are there enough unique statements that are **sufficiently similar to one another** to warrant the creation of a **new task, or revision to an existing task**? (Step 5 of the current process)

We trained machine learning models to address questions 1 and 2 and we conducted a judgment exercise with current HumRRO analysts to address question 3. In this report, we describe our automated statement-processing strategies. We also offer a summary of how these advances helped to redesign the statement analysis process into a more streamlined workflow that makes use of both machine-generated predictions and human judgments.

Data Preparation

HumRRO constructed the database used in this research as part of the O*NET Emerging Task project work for well over a decade. The database includes 136,955 rows, each reflecting a write-in task statement. To prepare this database for analyses, we performed basic text cleaning on the write-in task statements (i.e., converted all text to lower case, converted symbols to text [e.g., "+" to "plus", "&" to "and"], replaced all non-ASCII characters with a single white space, removed punctuation, removed extra white space, eliminated blank statements, performed spell-checking, and made spelling corrections) and created criterion variables for predictive modeling that corresponded to the first two key decisions noted earlier:

- Key Decision 1: Acceptable Task (Yes vs. No)
- Key Decision 2: Acceptable-Yet-Not-Completely Redundant vs. Acceptable-Yet-Completely Redundant

The main purpose for our spelling corrections was to ensure that as many words as possible could be mapped to our chosen pre-trained NLP model.[1] We only made spelling corrections if (a) a word was flagged as misspelled by a spell-checking function, (b) the word was not included in our chosen pre-trained NLP model, and (c) the spell-checking function was able to offer a suggested spelling (if conditions a and b were satisfied but condition c was not, the misspelled text was omitted from its statement in our analyses). When we corrected the spelling of a word, we recorded the substitution along with a Jaro-Winkler string-distance coefficient expressing how much the substituted text differed from the original text. In later steps of our analyses, we used

[1] We use the term "spelling corrections" loosely here, as it includes both corrections of misspelled text (e.g., "abscences" was corrected to "absences") and substitutions of text that are correctly spelled, but simply not recognized by the spell checker _and_ not included in our pretrained NLP model. Examples of correctly spelled text that were substituted include hyphenated terms that were not included in our pretrained NLP model (e.g., there was no embedding for "acid-bath," so we had to use the words "acid" and "bath" separately) and plural terms that were not included in our pretrained NLP model (e.g., our model did not include "alkalinities," so we used "alkalinity" instead). True spelling corrections for misspelled words were only appliable to write-in statements, not published O*NET task statements.

these string-distance metrics as penalties and down-weighted our substitutions as a function of how much they differed from the original text they replaced. This allowed us to make use of as much text as possible while accounting for uncertainty in the assumptions that underly a programmatic spelling-correction procedure. For write-in statements, we replaced a total of 8,693 words from 8,103 statements (5.9% of 136,389 non-null statements). For published tasks, we replaced a total of 507 words from 478 unique tasks (2.0% of 23,769 unique tasks).

Embeddings

The primary NLP-based portion of our analysis process used pre-trained word embeddings to quantitatively represent semantic meaning. A word embedding is a way of numerically expressing how words relate to each other so that one can essentially do math with words (e.g., king – man + woman = queen). By combining the embeddings for words that occur together in a sentence, one obtains a sentence-level aggregate embedding that gives a numeric approximation of the sentence's semantic meaning. Of course, this is an incomplete way of accounting for meaning, as this process treats a sentence as a "bag of words" without regard to the context or order in which words are used. Despite this limitation, embeddings are a powerful tool for evaluating the similarity of text because they permit the detection of similar text involving different words. For example, the proclamations "I like cats" and "I enjoy felines" each have only the word "I" in common, but they are essentially identical in meaning; these statements appear quite different if we only consider the words they have in common, but their similarity can be revealed by analyzing word embeddings.

Embeddings, regardless of whether they represent sentences or individual words, can be compared to each other to quantify the similarity between two pieces of text. The most common metric for expressing this similarity is a *cosine*, which indexes how close two vectors of embeddings are to each other. Cosines range from -1 (complete opposites) to +1 (completely identical), with 0 indicating no similarity at all. Cosines are effectively correlations between numeric expressions of text, and, in this research, cosines are our primary method for comparing write-in statements to published O*NET tasks and to each other.

We noted earlier that we used "pre-trained" embeddings, which means the embeddings were produced by others who analyzed generic text rather than text specific to our research context. Pre-trained embeddings are convenient because they are based on a large corpus of text (often obtained from web sources) and can be applied to research contexts in which there is insufficient text to derive custom embeddings. In our case, we used embeddings from the Global Vectors for Word Representation (GloVe; Pennington, Socher, & Manning, 2014) algorithm trained on a "common crawl" of the internet consisting of 42 billion tokens (i.e., word-like pieces of text). These GloVe embeddings represent 1.9 million unique tokens, each summarized as a 300-dimension numeric vector. Large collections of pre-trained embeddings like GloVe include most words in the English language and are therefore quite useful for numerically encoding novel pieces of text, such as the write-in statements that are the focus of this research. Although GloVe embeddings have 300 dimensions, we discovered that three of these dimensions are extreme outliers that have biasing effects on cosines. Other researchers have found the same oddities in these data (Lee, Ke, Huang, & Chen, 2016), so we used 297 of the 300 GloVe dimensions in our analyses and omitted GloVe dimensions 7, 97, and 225.

GloVe embeddings are encoded at the level of individual words, so we aggregated word-level vectors into statement-level vectors using a differential weighting strategy known as TF-IDF (short for "term frequency–inverse document frequency") weighting. TF-IDF weighting is a method for quantifying the importance of words as a function of both (a) their term frequencies

(“TF”), such that words that are used more often within a given document are more important to that document, and (b) their inverse document frequencies (“IDF”), such that words that are used in more documents are less important, overall, by virtue of being commonplace.[2] TF-IDF weights are the product of TF and IDF metrics, which allows these weights to balance the goals of quantifying importance based on word usage within and between documents. In this research, each published task or write-in statement was considered a separate document.

*Published O*NET Tasks*

Each cycle of write-in statements was matched with a database containing published task statements from the period in which the write-in statements were collected.[3] For the database of published O*NET tasks with which the write-in statements would be compared, we generated task-level embeddings by (1) extracting the GloVe embeddings for the words in each task, (2) computing TF-IDF weights for all published tasks, and (3) aggregating the word-level embeddings for each task into a TF-IDF weighted average. This process yielded one 297-dimension vector per published task.

Write-In Statements

We applied the same general process described above for published O*NET tasks to write-in statements, but with two modifications to the ways in which GloVe embeddings were aggregated. The first important difference affected how we computed TF-IDF weights. When TF-IDF weights are applied to novel text (e.g., write-in statements) in settings where a curated corpus of text already exists (e.g., the database of published O*NET tasks), it is common for the TF portion of the weights to be derived from the novel text while the IDF portion of the weights is borrowed from the curated corpus. By deriving IDF weights from a curated corpus, this method establishes a meaningful referent for cross-document word usage. We followed this practice and derived IDF weights based on the complete corpus of published O*NET tasks so that the IDF portion of the TF-IDF weights would not differ across applications of this method (e.g., the same IDF weights apply to the write-in tasks analyzed in this research as would apply to a collection of write-in statements analyzed in a future cycle). For words that appeared in write-in statements that were not part of the published-task corpus, we assigned an IDF weight of *e* (i.e., 2.718282), which gives them high-importance on the IDF side of our TF-IDF weighting scheme due to their novelty.

The other difference affected how we handled misspelled words. As noted earlier, we down-weighted word substitutions as a function of how much the substitute text differed from the misspelled text it replaced. Thus, we computed the operational weights we used to aggregate word-level embeddings into statement-level embeddings as:

$$wt_i = TFIDF_i \times \left(1 - distance_{JW_i}\right)$$

[2] We quantified TF as the proportion of times a given word was used within a document and we quantified IDF as the natural logarithm of the total number of documents divided by the number of documents containing the word.
[3] The published task lists used in this research ranged from December 2002 (Cycle 1) to January 2019 (Cycle 20). Within the O*NET Database Releases Archive (National Center for O*NET Development 2020b) this corresponds with the O*NET 4.0 database through the O*NET 24.0 database. Reports describing “cycle” updates for the Skills and Abilities domains can be found at https://www.onetcenter.org/research.html?c=KSA.

where *i* indexes write-in statements, *wt* represents the operational weight, *TFIDF* represents the TF-IDF weight, and $distance_{JW}$ represents the Jaro-Winkler string distance between original and substitute text ($distance_{JW}$ coefficients range from 0 to 1, where 0 indicates no difference between pieces of text; if text was unchanged by the spell-checking process, $distance_{JW}$ was set to 0). We used these operational weights to compute the weighted average of word-level embeddings for each write-in statement.

Question 1: Is the write-in statement an acceptable task?

To answer question 1, we developed a strategy to predict the acceptability of write-in statements using a combination of NLP and machine learning methods. The result of this synthesis was a supervised machine learning procedure in which characteristics of statements' text can be used to estimate the probability that a write-in statement is acceptable as a task.

We treated our model-development approach as a two-stage process. In the first stage, we developed a set of features (i.e., predictors) based on our understanding of the data and our experience with previous NLP-oriented machine learning efforts. We used those features to train several different machine learning models, compared the models' performance, and selected a model to use as a started point for the second stage. The second stage involved consulting with subject matter experts (SMEs; in this case, HumRRO analysts experienced with the Emerging Task ratings) about the model's classification errors, brainstorming a set of additional features to resolve deficiencies in the original model, and training another round of models with the expanded set of features. This effort was centered around the prediction of a binary criterion representing analysts' decisions about whether write-in statements were acceptable as tasks, or unacceptable as tasks (the base rate of acceptability was 51.4% in the initial stage and 50.9% in the revision stage, with the differences in base rates occurring due to modifications to the data sets used in our analyses). We summarize both stages of this effort in the following subsections.

Initial Modeling Effort

In the first stage of our model-building process, we engineered a set of features intended to summarize a variety of text attributes we expected to be related to acceptability. This set included features such as summaries of write-in statements' similarity to published O*NET tasks, the statements' embeddings, text complexity/readability metrics, and part of speech usage. The full feature set consisted of:

- Distributional summaries of cosines between statements and published tasks (mean, median, SD, min, and max)
    - Each write-in statement's embedding for a given O*NET-SOC was compared to the embedding of each task for that O*NET-SOC published at the time the statement was originally evaluated by analysts, which resulted in a distribution of cosines for each statement. We used the mean, median, standard deviation, minimum, and maximum cosines of each such distribution as a feature.
- Statement-level GloVe embeddings (297 dimensions)
    - In addition to the distributions of cosines, we included statements' aggregate GloVe embeddings as features to capitalize on any indicators of acceptability the embeddings may contain. This added a lot of features to our set, but we used

cross-validation techniques when we trained our models to avoid potential problems due to overfitting/overlearning.

- Complexity/readability indices
    - We calculated a variety of readability metrics to account for how write-in statements' complexity affects the probability of being acceptable as a task:
        - Word count
        - Character count
        - Number of syllables
        - Number of polysyllables
        - Average number of characters per word
        - Average number of syllables per word
        - Flesch-Kincaid reading level
        - Flesch-Kincaid reading ease
        - Coleman-Liau index
        - Automated readability index
        - Gunning fog index
        - SMOG grade level
- Part of speech frequencies (counts of verbs, nouns, adjectives, etc.)
- Job family dummy variables
- Proportion and frequency of words without embeddings (after spelling corrections were made)
- Dummy variable indicating whether a write-in statement matched a known low-effort response (e.g., "yes, "no," "N/A," "I don't know," "I have no idea")

We trained three types of machine learning models to predict the acceptability of write-in statements: elastic net multinomial logistic regression (ENMLR), random forests (RF), and gradient boosting machines (GBM). Each of these methods uses hyperparameters to set constraints on the parameters the models estimate to generate predictions. We offer brief descriptions of the methods and their hyperparameters below. We used *R* (R Core Team, 2020) for all our analyses.

Elastic Net Multinomial Logistic Regression (ENMLR)

ENMLR is a form of regularized regression that aims to boost the generalizability of model predictions by blending the penalization functions from the ridge and LASSO regression methods. This prevents model coefficients from arbitrarily becoming too large (as can happen when predictors exhibit multicollinearity) while allowing the coefficients for unimportant predictors to be set to zero so that they effectively drop out of the model. ENMLR uses two hyperparameters: lambda (the magnitude of the regularization penalty) and alpha (the blending parameter used to combine the penalties, where 0 = pure ridge and 1 = pure LASSO). We tested 100 lambda values for that were equally spaced from 0.00001 to 1 and we tested alpha values from 0 to 1 in increments of .1. We used the *glmnet* package for *R* (Friedman, Hastie, &

Tibshirani, 2010) to train our ENMLR models and we standardized all features to ensure they satisfied the assumptions of ridge regularization.

Random Forests (RF)

RF is a tree-based model in which random subsets of predictors are selected to build decision trees that make up a "forest." Each tree uses its predictors to classify observations into categories and the outcomes of all the trees in the forest are tallied in a vote-counting procedure. The overall model generates predictions by assigning each observation to the category for which it received the most votes. Compared to the other methods we used, hyperparameter tuning is much less important for RF models; aside from generating a large number of trees with a reasonable number of predictors sampled for each tree, tuning hyperparameters generally offers little improvement in prediction for RF models. We trained models using 300 trees with 20 predictors per tree and used the *randomForest* package for *R* (Liaw & Wiener, 2002) to train our RF models.

Gradient Boosting Machines (GBM)

GBM is also a tree-based model, but it uses "boosting" methods to make predictions rather than relying on a vote-counting procedure. In machine learning, individual decision trees like those used in RF are known as "weak learners" because they produce predictions that weakly correspond to observations' labeled classes. By contrast, a "strong learner" is a classifier that generates predictions that correspond closely to the observations' labeled classes. Vote counting in RF models uses insights drawn from many weak learners to classify observations, with the goal of using enough trees to average out their errors. GBM models, however, attempt to convert weak learners to strong ones by learning from their mistakes: Each tree added to a GBM model consists of a random subset of predictors and attempts to refine the predictions of previous trees so as to reduce the errors made by the overall model. We tested GBMs with 50 to 300 trees in increments of 50 and interaction depths from 1 to 5 (interaction depth refers to the number of splits performed within a tree); we held the minimum number of observations per node of a tree constant at 10 and we set shrinkage (i.e., the learning rate) to .1. We used the *gbm* package for *R* (Greenwell, Boehmke, Cunningham, & GBM Developers, 2020) to train our GBM models.

Model Training and Evaluation

We evaluated all our models using five-fold cross-validation, with an additional 10-fold cross-validation process used to tune hyperparameters within each of the five primary folds. We used a grid search method to tune all hyperparameters, such that all possible combinations of hyperparameters were evaluated before selecting the best-performing combination to use in model training based on overall hit-rate accuracy. We used the *caret* package for *R* to manage our model-training and cross-validation workflow.

A high-level summary of model fit for our initial model-building phase is shown in Table 1. Compared to a logistic regression model featuring a random normal variable, the GBM, RF, and ENMLR models trained on all features performed similarly, although the RF model was slightly less accurate than the other two on all metrics.[4] Given the similarity in performance among

[4] The performance of the RF model could be due how we handled its hyperparameters. However, the performance of RF relative to GBM and ENMLR in this research is consistent with patterns we have observed in previous research where RF hyperparameters were selected through substantial tuning.

machine learning methods for these data, we chose to rely on ENMLR for subsequent model-building efforts because it is the most transparent model type. Whereas GBM and RF generate predictions by running feature data through hundreds of trees in a black-box way, ENMLR produces a simple vector of log-linear regression coefficients that are simple to examine and implement.

Table 1. Summary of Fit Statistics for Machine Learning Models in Initial Modeling Effort Addressing Question 1

Method	Features	*k*	AUC	PBIS	Acc	MCC
Baseline Model						
Logistic Regression	Random normal variable	1	.50	.00	.51	---
Focal Models						
GBM	All features	474	.77	.48	.70	.40
RF	All features	474	.75	.43	.69	.38
ENMLR	All features	474	.77	.47	.70	.40
	All features without all part of speech variables	340	.76	.46	.70	.39
	Cosines and embeddings	302	.75	.43	.68	.37
	Cosines only	5	.68	.31	.63	.26
	Embeddings only	297	.74	.41	.68	.35

Note. *N* = 133,654 write-in statements (51.4% judged acceptable); AUC = area under ROC curve; PBIS = point-biserial correlation between predicted probabilities and actual acceptability; Acc = Accuracy (overall proportion of correct classifications); MCC = Matthews correlation coefficient (phi coefficient for actual and predicted status).

We trained additional ENMLR models using subsets of features to understand which features were driving predictive accuracy and which features might be unnecessary. As shown in Table 1, omitting part-of-speech frequency variables had a negligible impact on model fit, and the model in which the only features were cosine descriptive statistics and embeddings performed similarly to the RF model trained on all features. The embeddings appear to be an important driver of predictive accuracy, with the cosine descriptive statistics adding a modest increment in model fit. Based on these results, we tentatively recommended using an ENMLR model trained on all features except for part of speech variables to predict task acceptability. In the second stage of our approach to answering question 1, we sought to improve upon this recommended model by capitalizing on insights from SMEs.

Revised Modeling Effort

Our initial approach to predicting the acceptability of write-in statements yielded models with useful levels of predictive accuracy for three different machine learning methods. The features we included in the initial set of models were all based on ideas from an analysis plan we had constructed based on our understanding of the data and insights drawn from our previous NLP-based machine learning research. That set of features provided us with a good starting point for predicting acceptability, but left enough room for improvement that we decided to engineer additional features to increase predictive accuracy.

A model developed to mimic human judgments can often be improved by showing the model's predictions to SMEs and soliciting their feedback about how to avoid the types of prediction errors the model made. Based on this feedback, new features can be designed to capture the types of information the model was lacking—adding these new features to the model will ideally bring its predictions closer into alignment with expert judgments.

We compiled lists of classification errors for our model's false-positive errors (i.e., statements labeled unacceptable that the model predicted were acceptable) and false-negative errors (i.e., statements labeled acceptable that the model predicted were unacceptable) and showed these misclassified statements to two SMEs who were experienced HumRRO write-in statement analysts. We asked the SMEs to evaluate the misclassifications and identify quantifiable characteristics of the statements that could be encoded into features to help the model avoid making these types of errors in the future. Based on the SMEs' feedback, we developed additional features to include in another round of model building:

- Cosines between statements' embeddings and the embeddings of the SOCs' occupation titles
- Dummy variable indicating whether statements consisted of a single word
- Dummy variable indicating whether statements had both a verb and an object
- Frequency of non-task terms (e.g., ability, knowledge, skill)
- Frequency of "people words" (e.g., people, customer, client)
- Frequency of vague verbs (e.g., check, manage, assist)
- Dummy variable indicating whether there was only one verb that was also vague
- Number of words in longest run of verbs in each statement (ignoring conjunctions)
- Number of words in longest run of nouns in each statement (ignoring conjunctions)

We also updated some of the features used in our initial models. Most notably, we used a different program to extract parts of speech because the new program produced broader categories that were more parsimonious for modeling (e.g., it used one category for verbs as opposed to using separate categories for the gerund form, past participle, etc.). By using broader categories to count parts of speech, we reduced the overall number of features while retaining the most useful distinctions among words' part-of-speech labels.

In addition to revising our feature set, we applied some new cleaning procedures to the data with the goal of improving the quality of observations on which the new models would be trained. Based on feedback from the analysts, we excluded statements consisting entirely of symbols (*N* = 25), known low-effort responses (N = 1,116), and single-word write-in statements that had been rated acceptable as tasks (i.e., clear human decision errors; *N* = 317) from our analysis data. We also identified 2,215 statements for which acceptability decisions were missing and, operating under the assumption that all acceptable statements would be explicitly labeled as such, coded these as unacceptable; these cases originated predominately from cycle 3 (N = 2,202), suggesting a single-cycle difference in how acceptability decisions were recoded.[5] Altogether, these changes to the database resulted in a net increase of 757 write-in statements used in our analyses.

We trained another round of models using our updated data sets. Once again, we fit a baseline logistic regression model with a random normal predictor and fit several ENMLR models that omitted sets of features we suspected were contributing little to predictions. We also re-trained our preferred ENMLR model from the initial model-building phase on the revised data set with revised part-of-speech features to evaluate whether the new features improved our accuracy. See Table 2 for a summary of these models.

Adding our newly engineered features did relatively little to improve predictive accuracy and all candidate models performed similarly. Our model trained on the complete expanded feature set performed only marginally better than the preferred model from the initial stage; these models had the same AUC and accuracy, with the former demonstrating only a .01 increase in PBIS and MCC coefficients. Removing two large feature sets (part of speech variables and job family dummy variables) had very little impact on model performance, suggesting they are uninformative for identifying acceptable write-in statements. After omitting these uninformative features, our more parsimonious model had the same fit statistics as our preferred model from the initial phase.

Table 2. Summary of Fit Statistics for Machine Learning Models in Revised Modeling Effort Addressing Question 1

Features	*k*	AUC	PBIS	Acc	MCC
Baseline Models					
Random normal variable	1	.50	.00	.51	.00
Preferred model from initial phase (All features without all parts of speech variables)	337	.76	.45	.69	.38
Focal Models					
All features	361	.76	.46	.69	.39
All features w/o part of speech variables	346	.76	.46	.69	.39
All features w/o job families	340	.76	.45	.69	.38
All features w/o (a) part of speech variables and (b) job families	325	.76	.45	.69	.38

Note. *N* = 134,411 write-in statements (50.9% judged acceptable); AUC = area under ROC curve; PBIS = point-biserial correlation between predicted probabilities and actual acceptability; Acc = Accuracy (overall proportion of correct classifications); MCC = Matthews correlation coefficient (phi coefficient for actual and predicted status).

Figure 2 shows density distributions for the cross-validated predicted probabilities of acceptability for acceptable and unacceptable write-in statements. These distributions do demonstrate a useful magnitude of separation (a mean difference of 1.01 standard deviations); however, there is a substantial amount of overlap in the middle of the probability range, indicating that only cases toward the tails of the distributions can be classified with high levels of accuracy. Table 3 supports this, showing true positive rates (TPR), false negative rates (FNR),

[5] We subsequently re-evaluated this assumption and tested models in which we omitted statements that were missing acceptability decisions (revised total: 132,197 write-in statements; 51.8% judged acceptable). These models did demonstrate slightly better cross-validated fit than the models presented in this report, but the differences were quite small in magnitude (e.g., .01 on the various criteria).

true negative rates (TNR), and false positive rates (FPR) for probability thresholds ranging from .1 to .9 in increments of .1. For example, if statements with a probability of acceptability of .8 or greater were automatically retained as tasks, those classifications would correctly identify 17% of all acceptable statements and 96% of all unacceptable statements. Likewise, if statements with a probability of acceptability of .2 or lower were automatically rejected as unacceptable, those decisions would correctly identify 18% of all unacceptable statements and 98% of all acceptable statements. Jointly applying these two thresholds would help to identify the most obviously acceptable and unacceptable statements while making very few classification errors, leaving roughly 80% of the statements in the middle of the probability distribution for analysts to classify manually.

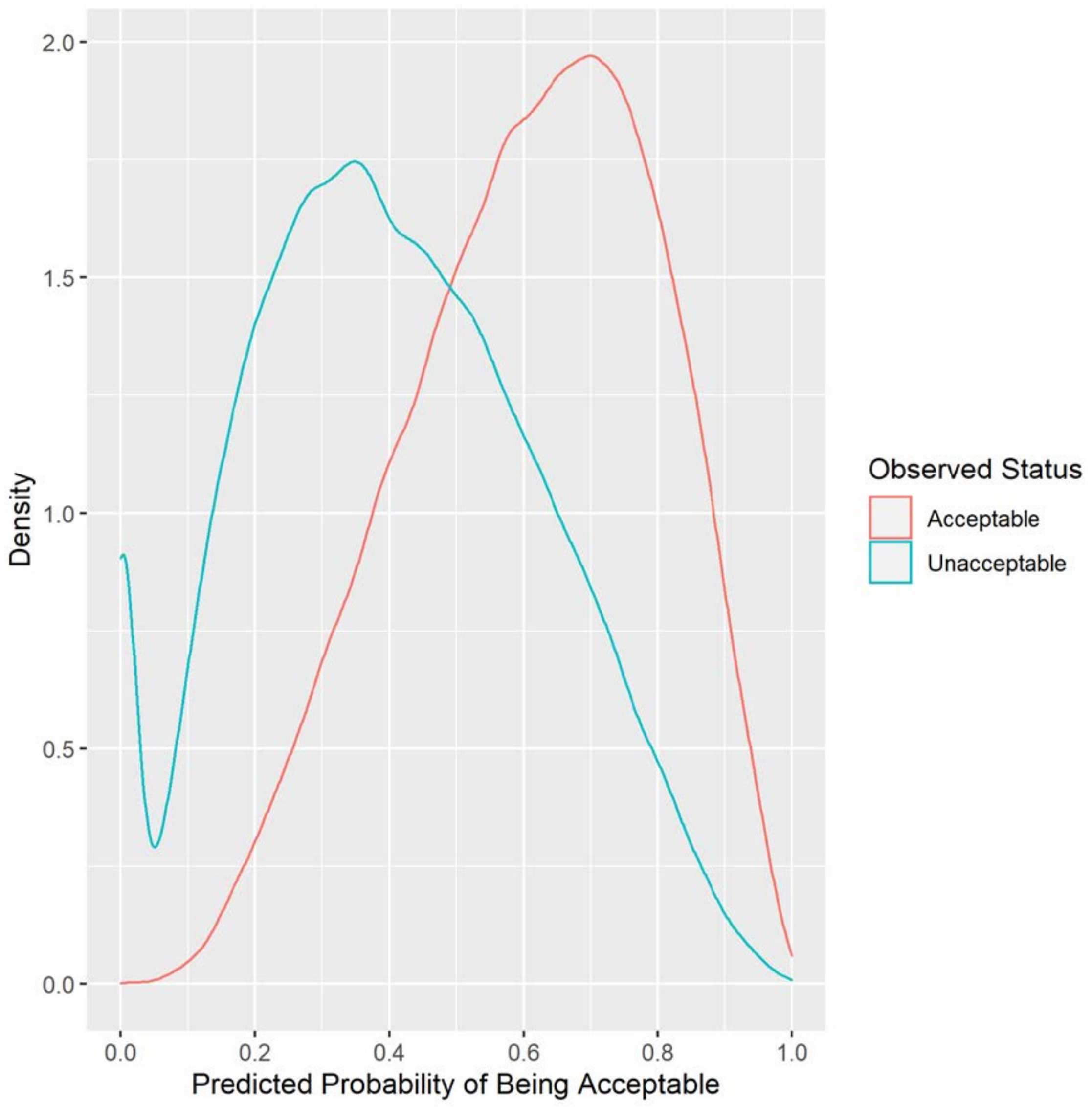


Figure 2. Distributions of predicted probabilities of acceptability as a task for acceptable and unacceptable write-in statements.

Table 3. Accuracy Statistics for Varying Probability Cutoffs When Predicting Write-In Statement Acceptability

Probability Cutoff	Accuracy	MCC	TPR	FNR	TNR	FPR	Percentage Below Cutoff
.1	.54	.19	1.00	.00	.07	.93	3.6%
.2	.59	.28	.98	.02	.18	.82	9.7%
.3	.64	.34	.93	.07	.34	.66	20.0%
.4	.68	.38	.85	.15	.51	.49	32.9%
.5	.69	.38	.72	.28	.67	.33	47.1%
.6	.67	.36	.55	.45	.80	.20	62.3%
.7	.62	.30	.35	.65	.90	.10	77.0%
.8	.56	.22	.17	.83	.96	.04	89.7%
.9	.51	.11	.04	.96	.99	.01	97.7%

Note. N = 134,411 write-in statements (50.9% judged acceptable); Probability Cutoff = minimum predicted probability to be classified as unacceptable; Accuracy = overall proportion of correct classifications; MCC = Matthews correlation coefficient (phi coefficient for actual and predicted status); TPR = true positive rate (proportion of acceptable statements correctly classified); FNR = false negative rate (proportion of acceptable statements incorrectly classified); TNR = true negative rate (proportion of unacceptable statements correctly classified); FPR = false positive rate (proportion of unacceptable statements incorrectly classified); Percentage Below Cutoff = percentage of statements classified as unacceptable.

Summary of Question 1

We trained machine learning models to predict whether write-in statements are acceptable as tasks, using NLP-based embeddings, lexical features, and similarity metrics as our key predictors. We found that three machine learning methods (ENMLR, GMB, and RF) performed similarly well at classifying write-in statements and all three produced predictions with useful levels of accuracy. Of these methods, we identified ENMLR as the most promising option for operationally predicting statements' acceptability because it produces transparent and simple-to-use coefficients. We used feedback from write-in statement analysts to engineer additional features and, although these new features did not markedly improve prediction, we were able to limit our features to a more parsimonious set that maintained the level of predictive accuracy observed in much larger models.

Question 2: Is the statement completely redundant with an existing task for a given O*NET-SOC occupation?

To answer question 2, we developed a strategy to predict whether acceptable write-in statements were completely redundant with O*NET tasks that had been published at the time the write-in statements were analyzed. All the data used to address question 2 were cleaned the same way as in our revised modeling effort for question 1. The criterion of interest in answering question 2 was a binary variable representing analysts' decisions about whether write-in statements judged as acceptable were completely redundant with existing O*NET tasks (the base rate of not-completely-redundant statements was 83.8%). Although analysts can classify write-in statements as partially redundant, our modeling effort was limited to predicting a lack of complete redundancy with published O*NET tasks; therefore, for economy of language, we refer to completely redundant statements as "redundant" and not-completely-redundant statements as "unique."

Like our approach to answering question 1, we used machine learning methods to address this question. However, given that predicting uniqueness is a more narrowly defined activity than predicting acceptability as a task (i.e., uniqueness is based on explicit comparisons with published tasks to determine whether statements overlap significantly with tasks, whereas the matter of acceptability is determined by a wider variety of evaluative factors), our feature set for this modeling effort was likewise narrower in scope.

We began with a simple logistic regression model in which uniqueness was regressed on write-in statements' maximum cosines with published tasks. The intention of this model was to determine whether write-in statements' magnitude of cosine similarity with their most similar published task was indicative of their redundancy status. We found that this model performed better than a baseline model in which the only predictor was a random normal variable, but its cross-validated probability estimates were only modestly correlated with write-in statements' actual uniqueness decisions (r = .17).

We aimed to improve upon the predictive accuracy of our initial logistic regression model by including additional descriptive statistics of statements' cosine distribution as features and once again using ENMLR. We regressed uniqueness decisions on the mean, median, SD, minimum, and maximum of statements' cosines with their SOCs' published tasks. This broader set of descriptive statistic features only improved prediction by a small amount (r = .19, Δr = .02; see Table 4 for more fit information). However, there were no additional features available to index statements' overlap with published tasks, so we proceeded to examine the distributions of probabilities predicted by the model.

Table 4. Summary of Fit Statistics for Machine Learning Models Addressing Question 2

Features	*k*	AUC	PBIS	Acc	MCC
Baseline Model					
Random normal variable	1	.50	.00	.84	---
Focal Models					
Max cosine	1	.62	.17	.84	.01
All cosine descriptives	5	.64	.19	.84	.04

Note. N = 68,730 acceptable write-in statements (83.8% judged unique); AUC = Area under ROC curve; PBIS = point-biserial correlation between predicted probabilities and actual uniqueness; Acc = accuracy (overall proportion of correct classifications); MCC = Matthews correlation coefficient (phi coefficient for actual and predicted status). MCC was undefined for the random normal variable model.

Figure 3 shows the distributions of cross-validated predicted probabilities of being unique for redundant statements and unique statements. Unlike Figure 2 where the contrasting probability distributions showed useful magnitudes of separation, the distributions in Figure 3 overlap substantially. The moderate degree of separation between the distributions (a mean difference of .53 standard deviations) implies that only the tails of the distributions are likely to be useful for cleanly classifying statements as redundant or unique (e.g., there are substantially more redundant statements with predicted probabilities below .7 than there are unique statements, but the distance between the distribution closes quickly as the probability threshold increases to .8). Table 5 shows the true positive rates (TPR), false negative rates (FNR), true negative rates (TNR), and false positive rates (FPR) for probability thresholds ranging from .50 to .95 in increments of .05.

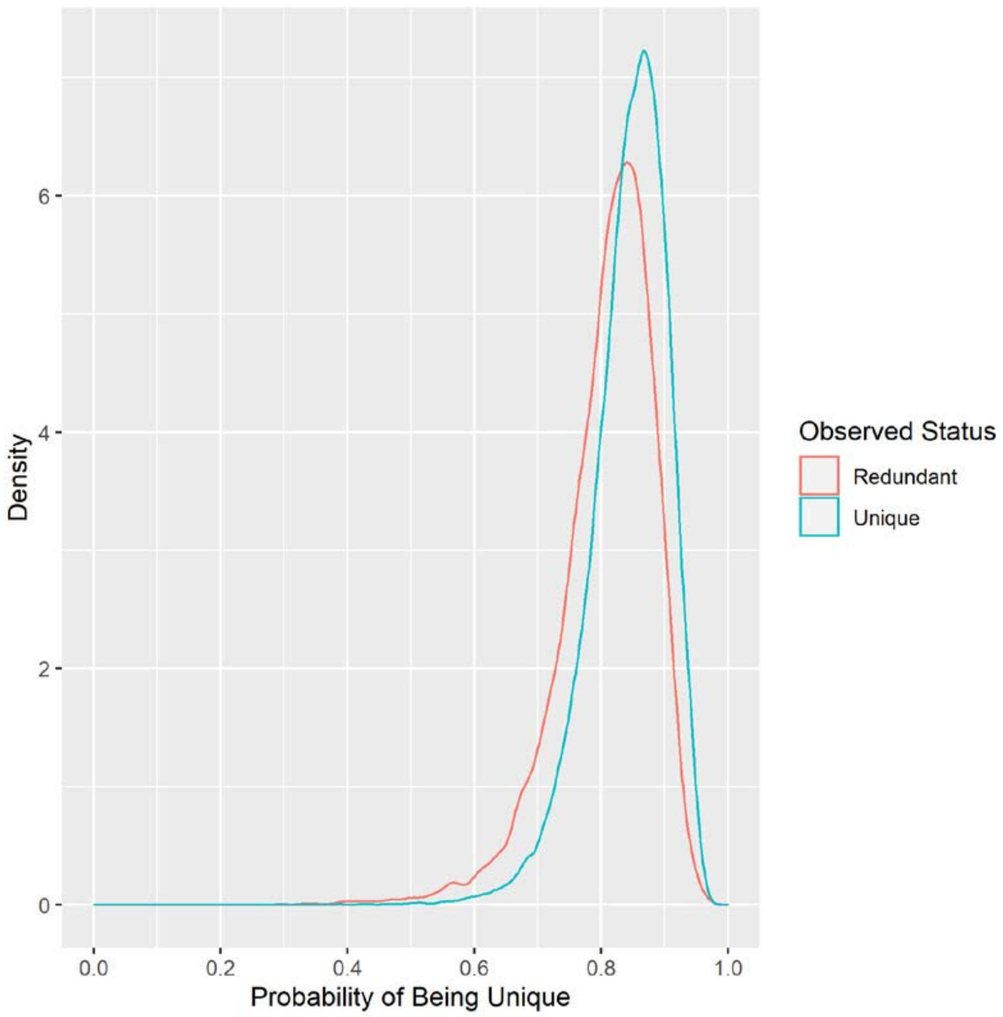


Figure 3. Distributions of predicted probabilities of uniqueness for unique and redundant write-in statements.

The model we developed to answer question 2 was less successful at prediction/classification than the model we developed to answer question 1. However, our model for predicting statements' uniqueness does provide some modest indication of the likelihood that a given statement is unique, so it can still help to streamline the process of analyzing write-in statements. Although the model is not accurate enough to be relied upon for automatic statement classification, we plan to pass its predicted probabilities along to analysts in the redesigned analysis process to give them an initial signal about how much each statement's text overlaps with tasks already on O*NET.

Table 5. Accuracy Statistics for Varying Probability Cutoffs When Predicting Write-In Statement Uniqueness

Probability Cutoff	Accuracy	MCC	TPR	FNR	TNR	FPR	Percentage Below Cutoff
.50	.84	.04	1.00	.00	.00	1.00	0.1%
.55	.84	.05	1.00	.00	.01	.99	0.2%
.60	.84	.07	1.00	.00	.02	.98	0.6%
.65	.84	.09	.99	.01	.04	.96	1.3%
.70	.83	.12	.98	.02	.08	.92	3.4%
.75	.80	.13	.93	.07	.18	.82	9.2%
.80	.72	.15	.79	.21	.38	.62	23.6%
.85	.54	.14	.51	.49	.68	.32	52.1%
.90	.29	.10	.17	.83	.93	.07	84.7%
.95	.17	.02	.01	.99	1.00	.00	99.3%

Note. N = 68,730 acceptable write-in statements (16.2% judged unique); Probability Cutoff = minimum predicted probability to be classified as unique; Accuracy = overall proportion of correct classifications; MCC = Matthews correlation coefficient (phi coefficient for actual and predicted status); TPR = true positive rate (proportion of unique statements correctly classified); FNR = false negative rate (proportion of unique statements incorrectly classified); TNR = true negative rate (proportion of redundant statements correctly classified); FPR = false positive rate (proportion of redundant statements incorrectly classified); Percentage Below Cutoff = percentage of statements classified as redundant.

Question 3: Are there enough unique statements that are sufficiently similar to one another to warrant the creation of a new task, or revision to an existing task?

In contrast to our machine learning approaches to questions 1 and 2, we administered a judgment exercise to SMEs to answer question 3. This exercise was designed to identify the point at which pairs of write-in statements are no longer similar enough to each other to merit consideration for inclusion in the same task category. The goal was to obtain a cosine-metric threshold for write-in statement similarity that could be used to group write-in statements into preliminary task categories to inform the creation of new tasks.

To generate the stimuli for the SME exercise, we computed the cosine similarities among all possible pairs of write-in statements for each SOC, which produced a total of 3,089,063 pairwise comparisons. This volume of cosines was obviously impractical for a SME exercise, so we selected 1,000 write-in statement pairs to use in the exercise via stratified random sampling. We stratified the statement pairs into bands of cosine magnitudes spanning ranges of .1 (i.e., .9-1.0, .8-.9, .7-.8, etc.; we grouped negative cosines and cosines from 0 to .1 in the same stratum), which ensured that write-in statement pairs of all levels of similarity were equally represented. We rank-ordered the statement pairs by their cosines and asked four HumRRO write-in statement analyst SMEs to identify the point in the list at which the preponderance of statement pairs were no longer sufficiently similar to each other to merit grouping them in the same task category (note that the analysts only saw the rank ordering of pairs; they did not see the cosine similarity metrics). SMEs reported the ranks at which they established their thresholds, and we linked those ranks to the cosines on which they were based.

The results of this exercise are summarized in Table 6. Three of the four SMEs provided very similar judgments, indicating that the threshold should be set around a rank of 320. When we mapped these ranks back to the distribution of cosines, we found that these three SMEs recommended a cosine threshold of about .67. SME number 4 was an outlier and recommend a rank threshold of 420, which corresponds to a difference of 100 ranks and a difference about .11 in the cosine metric relative to the other three SMEs. Given the consistency of judgments and the large difference with the outlying rater, we recommend a cosine threshold of .67 for grouping write-in statements into preliminary task categories.

Table 6. Summary of Similarity Thresholds for Question 3

SME Number	Rank Threshold	Cosine Threshold
1	319	.673
2	320	.673
3	322	.669
4	420	.561

Process Improvements and Next Steps

Based on the results of our modeling efforts and input from SMEs, we recommended several ways in which the write-in statement analysis process could be streamlined to reduce the burden on analysts and increase efficiency. The models we developed to answer questions 1 and 2 will be included in the analysis process for future cycles and their predicted probabilities will be used to rank-order lists of write-in statements, with the intent of facilitating analysts' reviews by indicating which statements are more likely to be (a) acceptable and (b) unique.

We also plan to use conservative probability thresholds to pre-populate acceptable/unacceptable and unique/redundant decisions for analysts (e.g., statements in the top 10% of acceptability probabilities are provisionally labeled as acceptable, those in the bottom 10% are provisionally labeled as unacceptable, and those in the middle 80% are entirely up to analysts to classify). This approach will help to automatically label the easiest-to-classify statements, which will allow analysts to focus their reviews on the statements with middling probabilities while still permitting them to override the automatically generated decisions if they see a classification error.

The SME-determined similarity threshold from our analysis of question 3 will be used to restructure the approach for grouping similar write-in statements into categories that inform the generation of new tasks. After analysts complete the streamlined activities for classifying acceptable and unique statements, the statements that were both acceptable and unique will be provisionally grouped into task categories based on their cosine similarity and analysts will edit and augment these categories while creating new categories as they see fit.

These process improvements have exciting potential to reduce the cognitive complexity of judgments while capitalizing on efficiencies afforded by machine-aided classification. Analyst SMEs estimated that these changes will reduce their reviewing time by 30-40% and the improvements are also expected to facilitate the training process for new analysts. The revised process will be piloted in a future write-in statement review cycle.

References

Dierdorff, E. C., & Norton, J. J. (2011). *Summary of procedures for O*NET task updating and new task generation*. https://www.onetcenter.org/dl_files/TaskUpdating.pdf

Friedman, J. H., Hastie, T., & Tibshirani, R. (2010). Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, *33*(1), 1–22. https://doi.org/10.18637/jss.v033.i01

Green, J. P., & Allen, M. T. (2020). *O*NET-SOC 2019 taxonomy development* (2020 No. 045). Human Resources Research Organization. https://www.onetcenter.org/dl_files/TaxonomyDev2019.pdf

Greenwell, B., Boehmke, B., Cunningham, J., & GBM Developers. (2020). *gbm: Generalized boosted regression models* (2.1.8) [Computer software]. https://CRAN.R-project.org/package=gbm

Lee, Y.-Y., Ke, H., Huang, H.-H., & Chen, H.-H. (2016). Less is more: Filtering abnormal dimensions in GloVe. *Proceedings of the 25th International Conference Companion on World Wide Web - WWW '16 Companion*, 71–72. https://doi.org/10.1145/2872518.2889381

Liaw, A., & Wiener, M. (2002). Classification and regression by randomForest. *R News*, *2*(3), 18–22. https://cran.r-project.org/doc/Rnews/Rnews_2002-3.pdf

National Center for O*NET Development. (2020a). *Emerging tasks - O*NET 25.1 data dictionary*. O*NET Resource Center. https://www.onetcenter.org/dictionary/25.1/excel/emerging_tasks.html

National Center for O*NET Development. (2020b). *O*NET® database releases archive*. O*NET Resource Center. https://www.onetcenter.org/db_releases.html

National Center for O*NET Development. (2020c). *O*NET 25.1 database*. O*NET Resource Center. https://www.onetcenter.org/database.html

Pennington, J., Socher, R., & Manning, C. (2014). GloVe: Global vectors for word representation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1532–1543. https://doi.org/10.3115/v1/D14-1162

R Core Team. (2020). *R: A language and environment for statistical computing*. R Foundation for Statistical Computing. https://www.R-project.org/

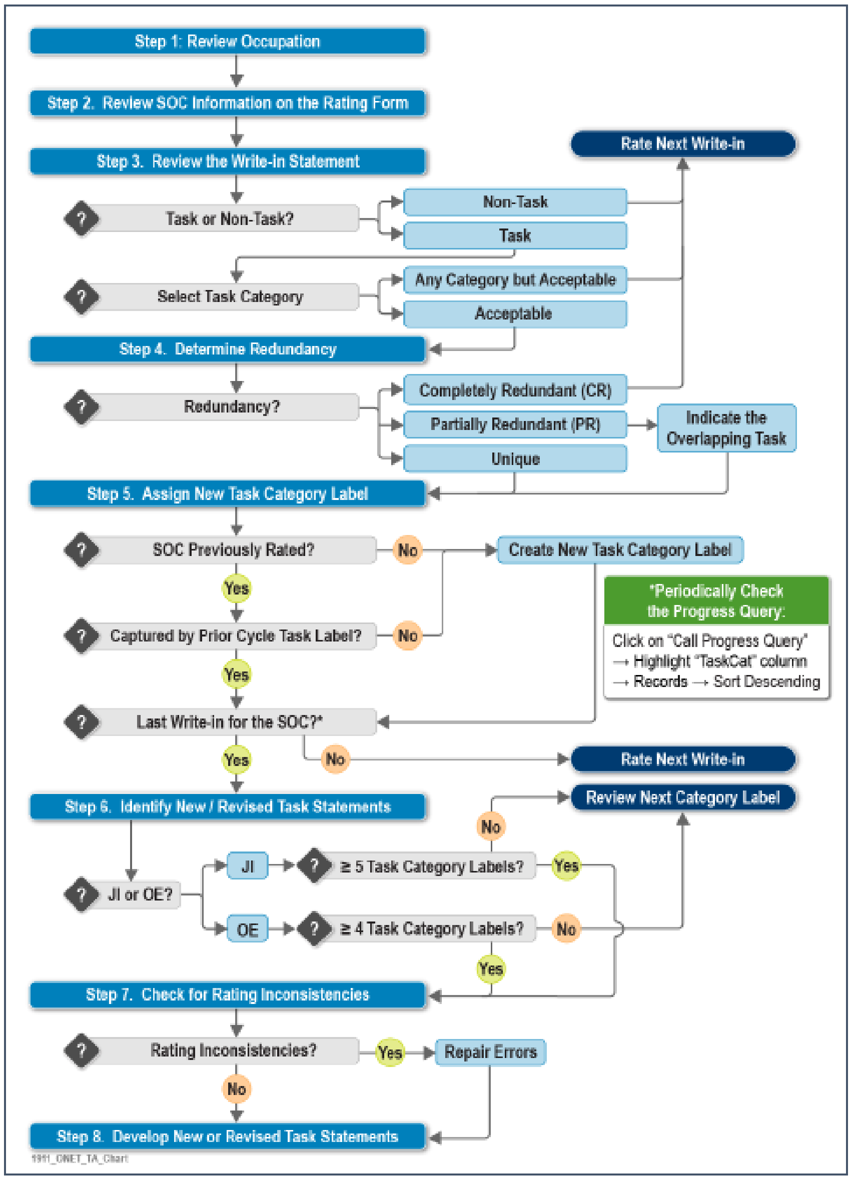
Appendix A: Examples of Write-In Statements

Analysts evaluate each write-in statement to determine whether it is acceptable as a task and, if it is acceptable, whether it is completely redundant or partially redundant with existing task statements for the O*NET-SOC in question. Write-in statements are only evaluated for partial redundancy if they are not completely redundant, and those judged to be neither completely redundant nor partially redundant are considered unique relative to the existing task list for the O*NET-SOC. Table B-1 shows the first 15 write-in statements in our database for O*NET-SOC 11-1011.00: Chief Executives. The write-in statements in Table B-1 are in their raw, uncleaned form and offer two examples of the need to correct statements for spelling errors before performing NLP-based analyses (both “subordinants” and “affiliated” are misspelled).

*Table B-1. Example Write-In Statements for Chief Executives (O*NET-SOC: 11-1011.00)*

Write-In Statement	Valid Task	Completely Redundant	Partially Redundant
Teach internal classes to subordinants	Yes	No	No
Teach university affilated management classes	Yes	No	No
Engineering	No	---	---
Direct sales effort	Yes	No	Yes
Interact with customers	No	---	---
Deal with human resource issues.	Yes	Yes	---
Build strategy for organization	Yes	No	Yes
Edit, write & design corporate communications for employees, members & public	Yes	No	No
Customer relations policy..	No	---	---
Operational policies.	No	---	---
Sales program to increase companies position in the market.	No	---	---
Financial planning for budgets/prospects.	Yes	No	Yes
Marketing strategies.	No	---	---
Coordinate between departments on large projects.	Yes	No	Yes
Virtual meetings with individual employers	No	---	---

Appendix B: Current Write-In Statement Analysis Process


Step 1: Review Occupation
Step 2. Review SOC Information on the Rating Form
Step 3. Review the Write-in Statement
Rate Next Write-in
Task or Non-Task?
Non-Task
Task
Select Task Category
Any Category but Acceptable
Acceptable
Step 4. Determine Redundancy
Redundancy?
Completely Redundant (CR)
Partially Redundant (PR)
Indicate the Overlapping Task
Unique
Step 5. Assign New Task Category Label
SOC Previously Rated?
No
Create New Task Category Label
Yes
Captured by Prior Cycle Task Label?
No
*Periodically Check the Progress Query:
Click on "Call Progress Query" → Highlight "TaskCat" column → Records → Sort Descending
Yes
Last Write-in for the SOC?*
Yes
No
Rate Next Write-in
Step 6. Identify New / Revised Task Statements
Review Next Category Label
No
JI or OE?
JI
≥ 5 Task Category Labels?
Yes
OE
≥ 4 Task Category Labels?
No
Yes
Step 7. Check for Rating Inconsistencies
Rating Inconsistencies?
Yes
Repair Errors
No
Step 8. Develop New or Revised Task Statements
1911_ONET_TA_Chart

Note. In the diagram above, references to "SOC" should be interpreted as "O*NET-SOC".

Appendix C: Revised Write-In Statement Analysis Process

Steps 1–6 of the workflow depicted in Appendix B were initially performed separately for each write-in statement (i.e., an analyst would go through those steps for one statement before moving on and applying it to another statement). The revised workflow makes use of our new machine learning models and reconfigures analysts' tasks so that they apply each step of the new process to all write-in statements for given O*NET-SOC before moving on to the next step. This change is expected to (a) limit the number of times that analysts have to "switch gears" during their judgment tasks and (b) reduce the burden on analysts' attentional resources.

Step 1: Review all write-in statements for the assigned O*NET-SOC and determine which ones are acceptable as tasks and which are unacceptable as tasks. To facilitate review, write-in statements are pre-sorted by descending model-predicted probability of being acceptable. Write-in statements with very high (e.g., .90) or very low probabilities (e.g., .10) are provisionally labeled as acceptable or unacceptable, respectively, prior to the analyst's review; the analyst may override any provisional classifications as they see fit.

Step 2: For each write-in statement judged acceptable in Step 1, use a list of existing statements for the O*NET-SOC sorted by magnitude of cosine similarity to the write-in statement to (a) evaluate the write-in statement's level of overlap with existing task statements, (b) classify it as unique, partially redundant, or completely redundant, and (c) if the statement is partially redundant, record the O*NET task number(s) of the task(s) with which it overlaps. To facilitate review, write-in statements are pre-sorted by descending model-predicted probability of being unique.

Step 3: For the acceptable write-in statements that are not completely redundant with existing tasks, decide which write-in statements should be combined into a task category from which to synthesize a new task statement.

Step 4: For write-in statements meeting criteria, write new task statements that (a) update existing task statements using information from partially redundant write-in statements or (b) consolidate information from write-in statements grouped within the same task category.[6]

[6] The criterion for developing new/revised task statements for Occupational Expert SOCs is four or more similar write-in statements. For Incumbent SOCs, the criterion is five or more.