Netskope: A Case Study in Text Classification with Unreliable Labels

Corey Sarcu, *Graduate Student, Georgia Institute of Technology*

***Abstract*—Netskope [1] is a global company that assists with the development of cloud-based cyber security solutions. Through various outreach events, both in-person and digital, Netskope aims to build a customer pipeline and boost service sales by marketing and sales campaigns targeted at relevant stakeholders in potential client organizations that fall within the Ideal Customer Profile (ICP). From the list of event attendees, Netskope categorizes individuals by department in order to prioritize follow-ups with stakeholders in the Information Technology (IT) and cybersecurity space in senior positions. Netskope also hopes to quantify its penetration into relevant industries by monitoring the attendance of stakeholders within those industries.**

Currently, Netskope uses keywords to classify attendees by job title, which has led to inconsistent results. Tasked with updating this methodology, I created a model based on pre-trained DistilBERT transformer encoder architecture [2] with custom classification layers added to match the output dimension of the historical data once it had been cleaned and consolidated. With this model, it is easier to more precisely quantify the priority of keywords for each category, as well as anti-keywords that detract from them. The final model has a 98.5% accuracy score in reproducing historical data, and is more internally consistent and precise with its implicit handling of keyword priority.

***Index Terms*—Georgia Tech, LLM, Text Classification, DistilBERT**

I. Introduction

CONSIDERING the ease of access to various webinars and abundance of direct marketing via in-person events in today's highly-leveraged digital information age, companies that rely on the sale of products and services need to prioritize their focus on key customer segments. In the case of Netskope [1], this includes individuals in the IT sphere, as well as senior decision makers. Netskope's goals include determining the penetration of their public-facing campaigns into desired segments, and better informing their marketing and sales teams on which leads to prioritize following up on.

Historically, Netskope's approach was to classify attendees by Job Title according to an ever-evolving keyword list. Keywords would be given a priority, with 1 being the highest; Titles would get tagged to associated Job Roles/Job Functions (i.e. the nature of the attendee's work), and Job Levels (i.e. the seniority/management status of the attendee) based on the hard-coded classification of the highest priority keyword present in the associated Title. This led to inconsistent results with the same Title over time, several examples of which are shown in Appendix A.

Keywords also required a manual addition for every word not yet explicitly accounted for in the already-existing method. Due to the rigidity of keyword matching, several permutations of the same word had to be encoded separately as well (ex.: mapping keywords ”automate”, ”automatic”, and ”automation” to the ”Engineer” Function). In addition, Netskope was less than 6 months away from retiring the system that hosted their keyword methodology. Thus, they needed to come up with a new solution to handle mapping Roles, Functions, and Levels to Titles relatively quickly.

As a further simplifying step, Netskope wanted to consolidate their go-forward mapping to focus on the target customer segment within the ”IT” Function. The desired go-forward hierarchy is:

i Function: Non-ICP
- Role: None

ii Function: Risk/Legal/Compliance
- Role: None

iii Function: Procurement
- Role: None

iv Function: Engineering
- Role: None

v Function: IT
- Role: Information Security
- Role: Networking
- Role: IT General
- Role: Development
- Role: Systems

And across all Functions and Roles:

i Level: Contributor
ii Level: Manager
iii Level: Director
iv Level: Executive
v Level: C-Level

II. Materials and Methods

A. Data

The original dataset was a filtered-down subset of over 10 years (gathered from campaigns October 2013 through February 2024) of historical Job Title data, along with the assigned Role, Function, and Level at the time. All data was collected from US-based campaigns, with the assumption that all resulting data would be in English. Initial data exploration revealed the following:

i 865,671 records of Title, Role, Function, and Level
ii 186,749 unique Titles, 35 unique Roles, 52 unique Functions, and 28 unique Levels
iii The most common Role was Information Security, accounting for 31% of the data; the top 5 Roles accounted for 91% of the data.

iv The most common Function was IT, accounting for 70% of the data; the top 5 Functions accounted for 91% of the data.
v The most common Level was Contributor, accounting for 36% of the data; the top 5 Levels accounted for 99% of the data.
vi Records for which any one of Title, Role, Function, or Level were blank/null accounted for 4% of the data
vii Across all Title fields, there were 31,772 unique words. Removing low information words (also known as stop words [3]) reduced this to 31,669 unique words.
viii There were cases of foreign language Titles coming through - Spanish and Japanese, among others. Without the ability to filter on language, it's unclear how much of these there were in the dataset.

Based on the above, as well as the aforementioned historical label inconsistency, I decided that the data needed to be cleaned and consolidated before modeling; details on data cleaning steps are listed in Appendix B.

After cleaning, 837,232 records remained, meaning that 28,439 were eliminated, around 3% of the original dataset. There were 7 distinct Roles, 5 Functions, and 6 Levels remaining in the dataset, which can be seen along with their distribution in Appendix C. The unique number of words was reduced slightly due to case redundancy to 29,279, or 29,190 without stopwords. A few more things I noted:

i Role ”GOVERNANCE RISK COMPLIANCE” and Function ”RISK/LEGAL/COMPLIANCE” almost never overlapped, even though intuitively they should represent the same idea. In fact, only 2 records in the cleaned data had this overlap. 1,566 records had the Function but not the Role, and 20,222 records had the Role but not the Function. I made a note that these might need to be consolidated in the new go-forward ”RISK/LEGAL/COMPLIANCE” function as an after-the-fact overwrite of model predictions.
   - Since the data is so thin for Function ”RISK/LEGAL/COMPLIANCE”, modeling based solely on that for the go-forward compliance Function would not be sufficiently robust.

ii Out of the 592,609 total records with Function ”IT”, 27,097 (5%) were labeled with Role ”NON-ICP”. This does not make sense, considering IT is in fact a core element of Netskope's ICP. Modifications will have to be made to the model's output at inference to make sure that this Role is not assigned to records with Function ”IT”.
iii A small proportion of Levels (slightly less than 0.4%) were ”UNKNOWN” and could not intuitively be grouped with any other categories; modifications will have to be made to model outputs that predict this Level.

Considering the unreliability of the labels in light of their basis on a changing keyword methodology that produced logically inconsistent results, care needed to be taken during model construction to not simply reproduce the results of the keyword methodology.

B. Review of Literature

Text modeling is a rich field of study that has seen many developments in recent years. At its most fundamental level, text modeling involves the processing of an input sequence to produce some kind of desired output. The tasks of a model are varied, including sequence classification, question answering, sentence continuation, machine translation, masked token prediction, next sentence prediction, and many others. Some of these tasks are merely ”pre-training” tasks designed to create favorable initialization values for internal model parameters for training on a truly desired production task which might have a more complicated objective function [4].

As a first step, the input has to be converted into a comprehensible form. The concept of tokenization was developed to convert input words into tokens and map those tokens to integers. A token represents a fundamental unit of linguistic meaning; sometimes, a token may be only a part of a word, rather than the whole word itself; in other cases, a single punctuation mark can be a token. As various large language models have developed, each one has their own specially-developed tokenizer that breaks down input sequences into their meaningful parts. Depending on the tokenizer, some methods to achieve this are:

i Stemming: Words are reduced to their word stems. This is not necessarily a morphological root, but merely a shorter version of the word. For example, ”leafs” $\rightarrow$ ”leaf”, ”leaves” $\rightarrow$ ”leav”
ii Lemmatization: Words are reduced to morphological roots and consolidated. This is akin to looking up the word in the dictionary. For example, both ”leafs” and ”leaves” $\rightarrow$ ”leaf”
iii Parts of Speech Tagging: Each word is assigned its relevant part of speech - noun, verb, etc. [5]

While tokenization does reduce the overall number of unique inputs, there are still issues associated with using raw tokens as input. First, the dimensionality is extremely high; representing all tokens as a unique sparse input vector would require the dimension of each vector to be as large as the number of unique tokens in the entire input dataset. Second, tokens that are similar in meaning are still treated as distinctly separate when input as sparse one-hot-encoded vectors. To address this, using a variety of pre-training tasks as a starting point, the idea of word embedding was developed. In this process, each input token is mapped to an n-dimensional ”embedding” vector, with dimensionality n as a modifiable parameter. This greatly reduces the overall dimensionality of the inputs from several tens of thousands to just a few hundred in most cases [6]. Also, the dense embedding space tends to reduce the euclidean distance between tokens that have similar intrinsic meaning, and even imputes some degree of semantic regularity into the data, as shown in Figure 1.

Input sequences in text modeling tend to vary in length, and there is importance inherent in the ordering of tokens as well. For this reason, recurrent neural networks (RNNs) were developed, a form of neural network which generates a time step t hidden layer, H_t [7, Equation 1]:

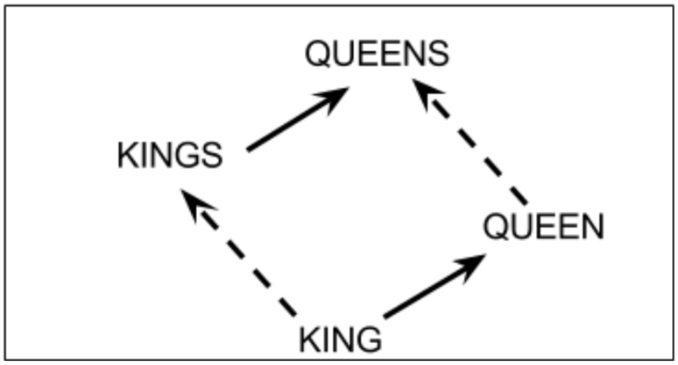


Fig. 1. The vector translation implying singular to plural, as well as the vector translation for switching sexes is preserved between different tokens [6, Figure 1]

$$H_t = \phi_h \left(X_t W_{xh} + H_{t-1} W_{hh} + b_h \right) \tag{1}$$

ϕ_h corresponds to an activation function, usually a sigmoid or hyperbolic tangent. $X_t \in \mathbf{R}^{n \times d}$ and $H_t \in \mathbf{R}^{n \times h}$, where n is the number of sequences passed in, d is the number of inputs in each sequence, and h is the number of hidden units; X_t represents the input embedded vectors at time t. $W_{xh} \in \mathbf{R}^{d \times h}$ and $W_{hh} \in \mathbf{R}^{h \times h}$ represents the input weight matrix and hidden-state-to-hidden-state weight matrices, respectively. $b_h \in \mathbf{R}^{1 \times h}$ is a bias parameter.

One shortcoming of RNNs is the issue of vanishing or exploding gradients as we have to backpropagate through longer time periods; every iteration of an RNN requires backpropagation to the very first time step. To combat this, the principle of attention was introduced; a particular hidden state would be based on a weighted average of trailing hidden states, the weights to be determined themselves as trainable model parameters. Each hidden state corresponds to a token; hidden states further back from the current token likely would have lower weights associated with them, thus focusing the attention on more recent tokens and their hidden states. Altogether, the learned weighted average of hidden states would form a numerical representation of the overall context of the sequence.

However, the RNN is still run once for every single token in a sequence, which is very computationally expensive, especially for longer sequences. In order to address this, the transformer architecture was developed, wherein the entire sequence is fed into a single model iteration. To differentiate the order of tokens in a sequence, each token received a position embedding along with the aforementioned token definition embedding; both embeddings should be the same dimension, so they are added together. Using multiple instances of weighting the hidden states for each token in the sequence gives rise to multi-headed attention, which allows for parallel instances of context that capture slightly different information. Attention is applied between distinct sequences of data, whereas self-attention applies to different tokens within the context of a single sequence [7].

There are many configurations possible for a transformer model, with chosen architecture dependent on the task at hand. In general, transformers include some type of encoding layer, made up of one or more self-attention mechanisms and feed-forward layers after. It's also common for residual connections and normalization layers to be part of a single encoder block. The overall encoder can consist of several sequential encoder blocks; the result is a condensed representation of input sequence context. Optionally, transformers can then have decoder blocks that take this context representation and convert it into an output sequence, using both the hidden states and output from the encoder layers for the decoder blocks [8].

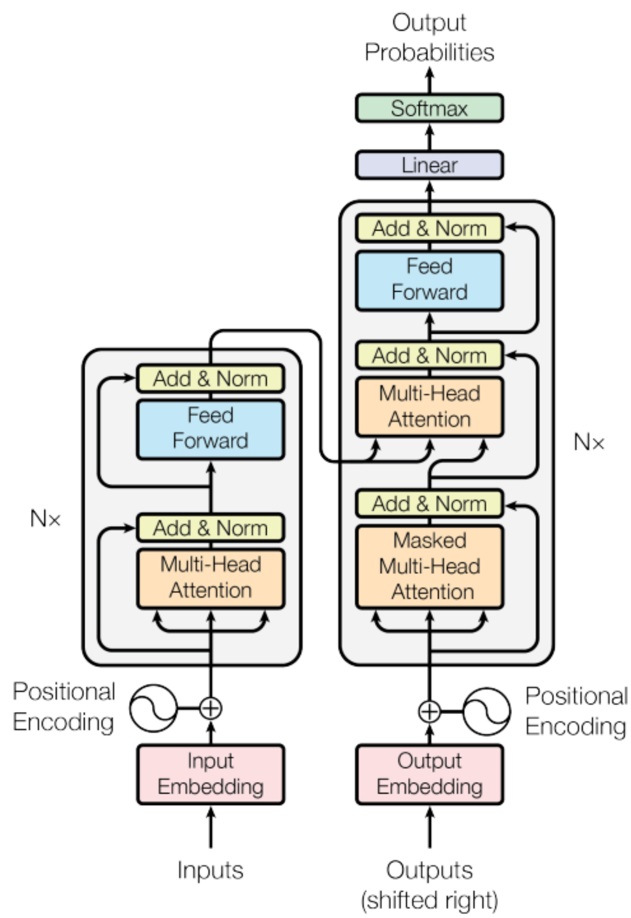


Fig. 2. Generalized Encoder-Decoder Transformer Architecture [8, Figure 1]

Encoder-decoder transformers are useful when the goal is taking an input sequence and generating an output sequence from the context information. A classical example of this is machine translation - the encoder takes the input in one language and represents it to the decoder, which reconstructs the input sequence from its contextual representation in the target language [8]. There is no constraint limiting the length of either the input or output sequences; the output could, for example, be much longer than the input if the target language requires more tokens/words to express the same idea.

For our particular task of text classification, we have no need for decoder blocks; the output will always be the same dimension (a vector for each of Role, Function, and Level, with dimensionality equal to the number of unique potential values for each field), and can simply use the context information generated by encoder blocks as inputs for feed-forward classification layers. While our particular classification task has no off-the-shelf production model publicly available, we can make use of encoder-only transformer models that have been pre-trained on other simpler tasks, with the addition of a few layers to match our desired output dimension. This strategy is known as transfer learning and has been widely adopted in many text modeling and other machine learning contexts [9].

An extremely popular encoder-only text transformer model with ample use-case documentation is BERT (Bidirectional Encoder Representations from Transformers). To develop this transformer, input data sequences randomly masked a few tokens and used the non-masked tokens to try to predict the vocabulary id of the missing ones. The ”bidirectional” label stems from the fact that information from tokens before and after the missing tokens could be used to fill them in [4]. The publicly available uncased base BERT model includes 110 million parameters, however there is a smaller version of this model, DistilBERT with only 66 million parameters, and an even smaller ALBERT model with 11 million parameters [10]. DistilBERT was created through distillation, a process by which a smaller student model is trained to reproduce the behavior of a teacher model (BERT). With only 60% of the parameters, DistilBERT achieved 97% of the performance of the teacher BERT model [2]. RoBERTA, a slightly larger version of BERT with a larger vocabulary size for input tokens, a more focused pre-training objective, and longer pre-training on bigger batches, demonstrates improvement over BERT on various language understanding benchmarks and adds approximately 15-20 million parameters [11].

C. Methodology

While there are other transformer encoder models out there, I decided that BERT was a good starting point for Netskope's text classification task; if the chosen model was ineffective, further options could be explored. Considering the task involved a single sequence input per sample leading to a single instance of predicted Role, Function, and Level, each of which only had a handful of desired output categorizations, I opted to use one of the simpler models: ALBERT or DistilBERT. While ALBERT was the smallest, I discovered that the installation of its tokenizer required a separate package (SentencePiece) which conflicted with other packages I was using, and opted to go with DistilBERT instead for my initial implementation. To see a full summary of the model architecture, refer to Appendix D.

The output of the desired model would be 3 vectors for each datapoint: a Role vector $r \in \mathbf{R}^{1\times 7}$, a Function vector $f \in \mathbf{R}^{1\times 5}$, and a Level vector $l \in \mathbf{R}^{1\times 6}$. The positions of each value in each vector would correspond to their integer encoding, starting with position 0; integer encoding is shown in Appendix C. The model would be trained to output the highest score for the correct label index, with ”correct” in this case corresponding to what the value was according to the keyword methodology at that time. With a single model to output all 3 labels, the inherent interactions between separate labels will be learned. A few augmentations would need to be made to these results that would affect the production-level inference predictions made by Netskope:

i When Function returns ”IT” and Role returns ”NON-ICP” (i.e. argmax(f) = 1 and argmax(r) = 5), Role will instead be classified based on the 2nd highest score in r, i.e. the next-most-likely Role after ”NON-ICP”. This is due to incompatibilities under the new hierarchy, see Introduction section (I) for more details.

ii For similar reasons, any Level that returns ”UNKNOWN” (i.e. argmax(l) = 5) will instead be classified based on the 2nd highest score in l.

iii After the above 2 steps, any Role that returns ”GOVERNANCE RISK COMPLIANCE” will set Function to ”RISK/LEGAL/COMPLIANCE”.

iv After all of the above, any Functions not ”IT” will have their Roles set to ”NONE”

These changes will *not* be applied to training predictions, since this will result in deviations from the training data, as they are based on historic keyword mappings. The go-forward adjustments are meant to be applied only during the inference step when the model is finalized. Additional assumptions and parameters:

i Model checkpoints will be saved after each training epoch.

ii Training data will be 80% of the dataset, with another 10% validation set to check out-of-sample accuracy and loss metrics in order to determine which checkpoint is best.

iii The final 10% will be a test set to evaluate the best checkpoint quality.

iv Maximum sequence length will be truncated at 64 tokens. The maximum sequence length in the entire dataset is 46 tokens, so this will not result in the loss of any information during training. Hypothetically, during inference there could be a very long Title with more than 64 tokens that *would* be truncated, but the assumption in that case is that there is enough information in the first 64 tokens for effective classification.

v Learning rate will be set to 1×10^{-5}; if the results are not satisfactory, hyperparameter tuning may be considered.

vi Data during training and inference will be fed in batches of 128 sequences.

vii The loss function for sequence i will be as follows:

$$L_i = \alpha_r * CE(r) + \alpha_f * CE(f) + \alpha l * CE(l) \quad (2)$$

Where CE represents the cross entropy loss of Role, Function, or Level, and α corresponds to $\frac{1}{\bar{A}_{1000}}$, with $\bar{A}_{1000}$ being the average accuracy over the last 1000 batches. α will further be normalized to add up to 3, with a starting value of 1 for each output field. This represents a weighted average of CE losses for each of Role, Function, and Level, with more weight being given to categories with lower accuracy, so as to prioritize

model updates to correct those errors and encourage similar accuracy between all 3 output categories.
viii Training will run for 20 epochs.

III. Results

There was very quick improvement over the training set, reaching trailing 1000 batch accuracy of above 90% in all categories within the first epoch. Figure 3 shows the improvement of the loss function over successive epochs, with the lowest loss on validation recorded after epoch 9. Figure 4 details the accuracy over validation and test sets for the model checkpoint after completion of epoch 9.

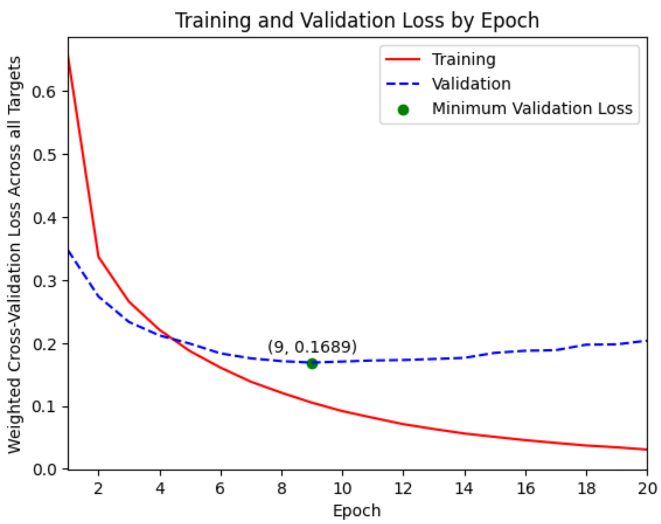


Fig. 3. Weighted loss function value across the entire training and validation sets at the end of each epoch. Note that the loss was not re-evaluated for training batches earlier in an epoch despite accruing several model updates; this explains why earlier epochs have lower losses on validation than training, despite validation being out-of-sample.

Output Field	Validation	Test
Job Role	0.9713	0.9719
Job Function	0.9953	0.9955
Job Level	0.9887	0.9875
Total	0.9851	0.9850

Fig. 4. Accuracy on validation and test data for the model checkpoint that achieved the lowest average weighted loss function value across the entire validation set, epoch 9

Accuracy curves over validation by epoch are shown in Appendix E, and test set confusion matrices and metrics (accuracy as well as individual class F1 scores/global average F1 scores) are shown in Appendix F. Using weights derived from accuracy on the validation set, weighted loss figures are determined for each training observation; the top 20 such losses are shown in Appendix G.

As an added step to explain the model's inner workings, I derived a method to determine implied keywords and keyword priorities. For each sequence S_j for $j \in N$, where N is the total number of sequences in the training dataset, this methodology involved:

i Determine exact logit score outputs for the model's predicted categories. That is, for output Role/Function/Level vectors r, f, and l, determine max(r), max(f), and max(l), as well as argmax(r), argmax(f), and argmax(l).
ii Compile a list of tokens t in the sequence, t_i for $i \in \{1, 2, ..., n\}$ where n_j is the number of *unique* tokens in the sequence S_j
iii Within the embedding section of fitted model, replace every element of the word embedding vector for given token t_i with a zero, effectively removing any representation of that token's implicit meaning from the model.
iv Determine exact logit score outputs for the same indices of r, f, and l as derived in step [i] (the argmax values). Subtract the score from this step from the score derived in step [i] (the max values) to determine how much the score *decreased* from the removal of this particular token's meaning. If it is negative, i.e. the score increased from the exclusion of that tokens information, this will be floored at 0. For token t_i, this will be referred to as m_i, the importance score of the token.
v With the importance of all tokens in the sample computed, add them up to get the total score reduction across all tokens from one-out exclusion, $M_j = \sum_{i=1}^{n} m_i$
vi Determine the average score reduction per token, $A_j = \frac{M_j}{n_j}$
vii Determine the priority index I_i for token t_i in sequence S_j: $I_i = \frac{m_i}{A_j}$. This is the ratio between the score reduction from token t_i and the average score reduction across all tokens in sequence S_j

Once all sequences are computed, then the results can be aggregated by token across all sequences that token appears in. For a given token t_i:

i Count the number of sequences that token appears in, C
ii Count the number of sequences for which that token has a priority index above 1 and divide by C to get the probability that a token has a greater than average positive influence on a predicted category's score: $P(I > 1)$.
iii Determine the mean μ_I and standard deviation σ_I of the priority index over all sequences token t_i appears in.

Intuitively, the derived index μ_I for token t_i will be high when, on average, token t_i tends to contribute on average more than other tokens in the sequence to the output logit score for the given predicted category. I_i also controls for sequence length, as longer sequences will tend to attribute less logit score impact to individual tokens that shorter ones. Thus, we do not want to arbitrarily inflate the importance of a token simply because it tends to appear in shorter sequences. I also omitted tokens that occurred in less than 100 distinct sequences in the training set, as the credibility associated with such small sample sizes would be suspect. Appendix H lists the top 5 keyword tokens for each predicted category. Taking the mirror image of the above process and instead computing the score *reductions* for each of the categories *not* predicted from a given sequence generates anti-keyword tokens, i.e. tokens that tend to indicate that a given Title does *not* belong to a particular category. The top 5 anti-keyword tokens for each predicted category are shown in Appendix I.

IV. Discussion

The model learned extremely quickly but still saw continuous, smooth improvement over the validation set for the first few epochs, indicating that the initial training rate was not set too high. Interestingly enough, while the objective loss function was minimized after training epoch 9, the accuracy on validation for each of Role/Function/Level continued to improve with further epochs. In the case of Function, this was maximized by epoch 17, but Role and Level are maximized after epoch 20 and thus may have continued improving if further epochs were run. My decision to choose the model based on the loss metric is due to the fact that accuracy only measures a model's ability to predict the correct output, whereas cross entropy loss incentivizes the model to predict the correct output but also to bring down the scores of each incorrect output; it is a more comprehensive view of model insight. In addition to this, arbitrarily tracking to accuracy would likely result in overfitting to certain quirks in the data, considering the historical target data is known not to be 100% correct. As such, the goal was to give the model enough time to learn the inherent classification power behind each distinct model token, but not too much time as to start to entrench the aforementioned inherent self-contradictions of the prior keyword model. Thus, the selected final model was the epoch 9 checkpoint.

The overall accuracy on the testing set as shown in Figure 4 were comparable across Role, Function, and Level; the slight differences were likely due to the number of distinct classification categories in each output. For example, since Role had 7 distinct output fields against Function's 6 and Level's 5, it had a slightly lower overall accuracy. The fact that all 3 accuracy metrics are rather close indicates that the combined weighted loss metric did a good job at weighting model updates toward erroneous predictions, and thus preventing a single label from dominating at the expense of the others.

As part of the investigation into divergences between my model results and the remapped historical data, I highlighted in Appendix G the top 20 training observations where my model was most inaccurate according to the weighted loss function. From that set I noted:

- i 1 observation that appeared to be in a foreign language
- ii 8 observations that the model appeared to incorrectly categorize, one of which was the aforementioned foreign language Title, another of which was a duplicate (i.e. the exact same title appeared in the data twice with the same labels), and one of which the original keyword methodology also seemed to tag incorrectly
- iii 9 observations that the model appeared to correctly categorize *despite* the historical keyword tagging appearing to be incorrect.
- iv 3 observations where the model and the keyword tagging differed, but either result seemed plausible

Even by a conservative evaluation of where the model tends to most differ from the prior tagging method, about half of the "inaccuracies" in the model's tagging are probably actually correct. Since this is only drawing from among those with the greatest loss, i.e. differing the most from the keyword's tagging, likely *more* than half of the 3%/0.5%/1.25% of Role/Function/Level that are "inaccurately" categorized in the test set shown in Figure 4 are actually "correct".

Even still, there are always inaccuracies in every model, and one cannot account for every single edge case. As such, I added an override table for certain Titles to overwrite model results with custom tagging. This might be useful in a case like Title = "SR PM", where it seems to be a NON-ICP Role/Function, but the model tags it otherwise.

In consideration of the caveats around the aforementioned accuracy metrics, the F1, precision, and recall metrics from Appendix F should be considered with care. Precision is based on the ratio of true positives to predicted positives of a category, recall is based on the ratio of true positives to actual positives of a category, and F1 statistic is the harmonic mean between them [12]. In consideration of this, we observe Level "UNKNOWN". Because we are just going to overwrite all "UNKNOWN" values with the next highest logit score, a low recall is actually desirable for that specific category; this means that results that were historically coming through as "UNKNOWN" are being re-categorized. By the same token, a high precision for "UNKNOWN" is also desirable, as this means that there are relatively fewer non-"UNKNOWN" levels that are being erroneously classified as "UNKNOWN". Indeed, this is the relationship observed in the metrics - a low recall and a significantly higher precision. Indeed, most of the re-classification is moving from "UNKNOWN" $\rightarrow$ "CONTRIBUTOR", which makes sense - more senior Levels are usually more obvious. Similar observations can be made about the relatively lower recall for Roles "IT GENERAL" and "SYSTEMS" (ITG&S). These are more general Roles when compared to more core segments such as "NETWORKING" and "INFORMATION SECURITY" (N&IS). ITG&S have lower F1 scores overall, but this is more driven by recall rather than precision; like for "UNKNOWN", these Roles are being reclassified from their historical tagging, and N&IS tend to receive most of these. By contrast, N&IS have recall right on level with their precision, if not slightly higher. This is another point of encouragement for the updated model; assuming that Netskope had a better handle on tagging the core customer segments, the model is not changing those taggings by much. It affects non-core segments to a greater degree, and, based on what I've observed so far, likely categorizes them more accurately.

The keyword and anti-keyword analysis helps explain how the model operates. Top keywords, when present, will increase the probability of the model selecting the relevant category; top anti-keywords will conversely reduce the probability. Although the keywords are categorized by outputs prior to the post-processing steps explained in the Methodology subsection of Materials and Methods (II-C), those post-processing steps are simple enough that the insight provided by keywords/anti-keywords is still valuable. It is worth noting that the model will likely mis-categorize the Level of Titles that include the "ASSISTANT" token but are still higher-level employees, due to the fact that ”ASSISTANT” is the most impactful keyword for the ”CONTRIBUTOR” Level. This is seen in one of the examples in Appendix G pertaining to ”ASSISTANT VP”. These Titles are likely somewhat rare, or else there would be more than one among the highest-loss examples; they will likely need to be handled via the overrides table, or custom model tweaking by retraining on a dataset that includes a lot of ”ASSISTANT” Titles at various Levels. Generally, the keyword tables provide an intuitive way to explain the inner workings of the model to a non-technical audience, as it fits the nomenclature of their current methodology. Anti-keywords, while slightly less intuitive, make sense. The token ”SOX”, for example, likely corresponds to a finance Function/Role associated with the Sarbanes-Oxley law; it appears very highly in several anti-keyword lists for various IT Functions/Roles.

In consideration of the multiple concerns about deriving accuracy compared to a historical dataset of questionable fidelity, I determined that it made little sense to tune hyperparameters or explore alternate modeling architectures; at most, it could move the accuracy a few percentage points, and is likely not worth the effort.

V. Issues and Further Exploration

As previously mentioned, the target data for this model were inconsistent over time relative to unique Titles. The extent of the inaccuracy of historic labels in this data is also inherently unknowable, making the evaluation of model quality more difficult. Using a more all-encompassing loss metric such as weighted cross-entropy loss (as opposed to simple metrics like accuracy) to determine model training progress helped to offset this, but did not perfectly bridge the gap, since predicted logit vectors were, in certain instances, incentivized to boost the logit scores of multiple output indices: the historically tagged one, as well as one or more additional that are ”theoretically correct” based on inherent token meaning and context. Bringing in additional metrics, such as precision and recall, helped to better understand the precise nature of the model’s errors and whether they were error types that were actually desirable. However, there is still uncertainty as to the true quality of the model.

Determining the impact of each token on the predicted class, which ultimately culminated in the production of keyword and anti-keyword lists with a methodology that I created, proved challenging. There is extensive literature on various methods for influence attribution, such as Grad-CAM [13] and integrated gradients [14] among others. I was able to implement both of these in a way which traced back the influence on the predicted output class of each element of the word embedding vector associated with each token in a sequence. However, it was not exactly clear how to condense the 768 values associated with each embedding vector into a single value for the relevant token. Unadjusted averaging as well as averaging only positive values produced bizarre keyword results that contravened intuitive reasoning. Indeed, most of the literature on Grad-CAM, integrated gradients, and other input attribution methods appears to be in the field of image modeling, where inputs are simple RGB pixel values and do not require abstraction in the same way token embeddings do. Thus, I came up with my own methodology to determine token impact, and the results proved to be intuitive and explanatory.

Even still, there is potential to improve the keyword/anti-keyword methodology. Tokens with high μ_I (average priority index) may appear at the top of the list in Appendices H and I, but one must not ignore the σ_I (standard deviation of priority index). For example, the ENGINEERING Function has keyword ENGINEER as the 5th highest based on μ_I, but the associated σ_I is lower than any others in the top 5, indicating that, while it may tend to contribute less on average to the positive logit score for the ENGINEERING Function than the 4 higher keywords, its contribution is more stable and tends to vary less. A more refined token impact attribution scheme should take into account the variance of the priority index, prioritizing keywords with a high index but also keywords whose contribution tends not to vary as much.

While computational resources generally prove limiting in application of the largest language models, the choice of a more compact model such as DistilBERT effectively managed this. Training runs took only 40 minutes to run through each epoch. The most time-consuming step of this analysis, the generation of keyword and anti-keyword lists, took only 10-15 hours to run through the entire training dataset. Production-level inference should run even faster, as there is no need to engage model dropout layers, calculate gradients, or update parameters. In consideration of the extremely quick plateauing of loss improvement, an even more compact architecture might be desirable for quicker training in case the model would need to be updated again. This could take the form of ALBERT [15] or some other extremely lightweight model.

Any further updates to the model would require a restatement of historical data to adhere to the new hierarchy. This could be achieved by running all of the historical data through the model, as it is calibrated to produce outputs that fall under the new hierarchy; the override table could be used to further iron out model inconsistencies in certain edge cases. It is recommended to update the model if (and when) the override table starts to grow sufficiently large; depending on the rate of update, several thousand overwritten Title classifications may warrant a model recalibration.

VI. Conclusion

The DistilBERT model architecture proved to be effective at capturing existing token relationships and context, achieving 98.5% accuracy over the test set compared to historical labels. The model’s divergence from historical tagging makes intuitive sense more often than not and appears to have implicitly learned to ”correct” several mis-tagged historical records. The model can be explained by a keyword intuition similar to the current structure, but additionally makes use of sensible anti-keywords that lower likelihood of prediction for certain outputs. The supplementary keywords/anti-keywords derived from the model serve as a useful tool to explain it to non-technical stakeholders. The underlying model architecture is lightweight and future updates are simple to manage, as the training loop is already set up. Thus, this model will allow Netskope to retire their old keyword methodology and move forward with a more robust and adaptable framework as their outreach campaigns continue to grow and evolve.

References

[1] *Netskope*. 2024. URL: https://www.netskope.com/.

[2] Victor Sanh et al. "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter". In: *CoRR* abs/1910.01108 (2019). arXiv: 1910.01108. URL: http://arxiv.org/abs/1910.01108.

[3] Yaohou Fan, Chetan Arora, and Christoph Treude. *Stop Words for Processing Software Engineering Documents: Do they Matter?* 2023. arXiv: 2303.10439 [cs.SE].

[4] Jacob Devlin et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2019. arXiv: 1810.04805 [cs.CL].

[5] Kerem Kargin. *NLP: Tokenization, stemming, lemmatization and part of speech tagging*. Feb. 2021. URL: https://medium.com/mlearning-ai/nlp-tokenization-stemming-lemmatization-and-part-of-speech-tagging-9088ac068768.

[6] Felipe Almeida and Geraldo Xexéo. *Word Embeddings: A Survey*. 2023. arXiv: 1901.09069 [cs.CL].

[7] Robin M. Schmidt. *Recurrent Neural Networks (RNNs): A gentle Introduction and Overview*. 2019. arXiv: 1912.05911 [cs.LG].

[8] Ashish Vaswani et al. *Attention Is All You Need*. 2023. arXiv: 1706.03762 [cs.CL].

[9] Jeremy Howard and Sebastian Ruder. "Universal Language Model Fine-tuning for Text Classification". In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Ed. by Iryna Gurevych and Yusuke Miyao. Melbourne, Australia: Association for Computational Linguistics, July 2018, pp. 328–339. DOI: 10.18653/v1/P18-1031. URL: https://aclanthology.org/P18-1031.

[10] URL: https://huggingface.co/transformers/v4.4.2/pretrained_models.html.

[11] Yinhan Liu et al. *RoBERTa: A Robustly Optimized BERT Pretraining Approach*. 2019. arXiv: 1907.11692 [cs.CL].

[12] David M. W. Powers. *Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation*. 2020. arXiv: 2010.16061 [cs.LG].

[13] Ramprasaath R. Selvaraju et al. "Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization". In: *International Journal of Computer Vision* 128.2 (Oct. 2019), pp. 336–359. ISSN: 1573-1405. DOI: 10.1007/s11263-019-01228-7. URL: http://dx.doi.org/10.1007/s11263-019-01228-7.

[14] Joseph Enguehard. *Sequential Integrated Gradients: a simple but effective method for explaining language models*. 2023. arXiv: 2305.15853 [cs.CL].

[15] Zhenzhong Lan et al. *ALBERT: A Lite BERT for Self-supervised Learning of Language Representations*. 2020. arXiv: 1909.11942 [cs.CL].

APPENDIX A
HISTORICAL DATA EXAMPLES WITH MULTIPLE TAGGED OUTPUTS

Title	Job Role	Job Function	Job Level	Record Count
Cio	Information Security	IT	C-Level	1143
Cio	IT General	IT	C-Level	889
Cyber Security Engineer	Development	IT	Contributor	806
Cyber Security Engineer	Information Security	IT	Contributor	245
Cyber Security Engineer	Development	IT	Executive	14
Cyber Security Engineer	Development	IT	Non-Manager	10
Cyber Security Engineer	Development	IT	Unknown	10
Cyber Security Engineer	Development	IT	Manager	9
Cyber Security Engineer	Development	IT	Director	7
Cyber Security Engineer	Development	IT		5
Cyber Security Engineer	Networking	IT	Contributor	1
Director Of It	IT General	IT	Director	619
Director Of It	Networking	IT	Director	560
Director Of It	Information Security	IT	Director	46
Director Of It	Program Management	IT	Director	3
Director Of It	Development	IT	Director	2
IT	IT General	IT	Contributor	470
IT	Information Security	IT	Contributor	465
IT	Information Security	IT	Manager	55
IT	Information Security	IT	Director	9
IT	Information Security	IT	C-Level	6
IT	IT General	IT	Manager	5
IT	Networking	IT	Director	4
IT	Information Security	IT	Non-Manager	3
IT	Information Security	IT	Engineer/Admin	2
IT	IT General	IT		1
IT	Information Security	IT		1
VP of IT	Information Security	IT	Executive	1032
VP of IT	IT General	IT	Executive	861

APPENDIX B
DATA CLEANING STEPS

1) Converted all fields (Title, Role, Function, Level) to uppercase
2) Any titles with multiple unique Role/Function/Level taggings were overwritten to the most common Role/Function/Level for that Title. For example, all Title "IT" entries were assigned "IT General"/"IT"/"Contributor" (refer to Appendix A)
3) Removal of data for which any one of the input or output fields had null values after steps 1 and 2
4) Any Titles without Latin characters were filtered out.
5) Titles that at a glance seemed devoid of information were removed (ex.: "**No Longer With Company**", "#NAME?").
6) Role reassignments
   i "NETOWORKING" → "NETWORKING"
   ii "IT FACILITIES", "IT", "SENIOR MANAGER, INFORMATION TECHNOLOGY" → "IT GENERAL"
   iii "BUSINESS SYSTEMS" → "SYSTEMS"
   iv "SENIOR MANAGER, SECURITY, RISK, AND COMPLIANCE", "IT/IS COMPLIANCE/RISK/CONTROL STAFF" → "GOVERNANCE RISK COMPLIANCE"
   v "DEVELOPMENT" stays as "DEVELOPMENT"
   vi All other Roles set to "NON-ICP"
7) Function reassignments
   i "INFORMATION TECHNOLOGY","IT - SECURITY","IT - NETWORK","INFORMATION SECURITY, INFORMATION TECHNOLOGY","IT OPERATIONS","IT-SEC ADMIN","DIRECTOR GLOBAL IT","INFORMATION SECURITY, INFORMATION TECHNOLOGY, ENTERPRISE ARCHITECTURE","INFORMATION TECHNOLOGY, INFORMATION TECHNOLOGY EXECUTIVE" → "IT"
   ii "ENGINEERING & TECHNICAL","ENGINEER SASE" → "ENGINEERING"
   iii "PURCHASING","SOURCING / PROCUREMENT" → "PROCUREMENT
   iv "LEGAL","RISK, LEGAL OPERATIONS","LAWYER / ATTORNEY","GOVERNMENTAL AFFAIRS & REGULATORY LAW" → "RISK/LEGAL/COMPLIANCE"
   v All other Functions set to "NON-ICP"
8) Level reassignments
   i "INDIVIDUAL CONTRIBUTOR","CONTRIBTUOR" → "CONTRIBUTOR"
   ii "MANAGEMENT","MANAGER LEVEL","THREAT HUNTING MANAGER","IT SECURITY MANAGER" → "MANAGER"
   iii "SENIOR EXECUTIVE","EXEC." → "EXECUTIVE"
   iv "DIRECTOR LEVEL","IT INFRASTRUCTURE DIRECTOR","DIRECTOR OF ENTERPRISE CLOUD BUSINESS","IT SECURITY DIRECTOR" → "DIRECTOR"
   v "CXO","C-SUITE","DIRECTOR (IT & PROJECT) & CHIEF INFORMATION SECURITY OFFICER","C LEVEL" → "C-LEVEL"
   vi All other Levels set to "UNKNOWN"

Appendix C
Cleaned Data Output Distribution

Role

Value	Proportion	Integer Encoding
INFORMATION SECURITY	0.3254	2
NETWORKING	0.2590	4
NON-ICP	0.2227	5
DEVELOPMENT	0.0975	0
IT GENERAL	0.0540	3
GOVERNANCE RISK COMPLIANCE	0.0242	1
SYSTEMS	0.0172	6

Function

Value	Proportion	Integer Encoding
IT	0.7078	1
NON-ICP	0.1928	2
ENGINEERING	0.0945	0
PROCUREMENT	0.0030	3
RISK/LEGAL/COMPLIANCE	0.0019	4

Level

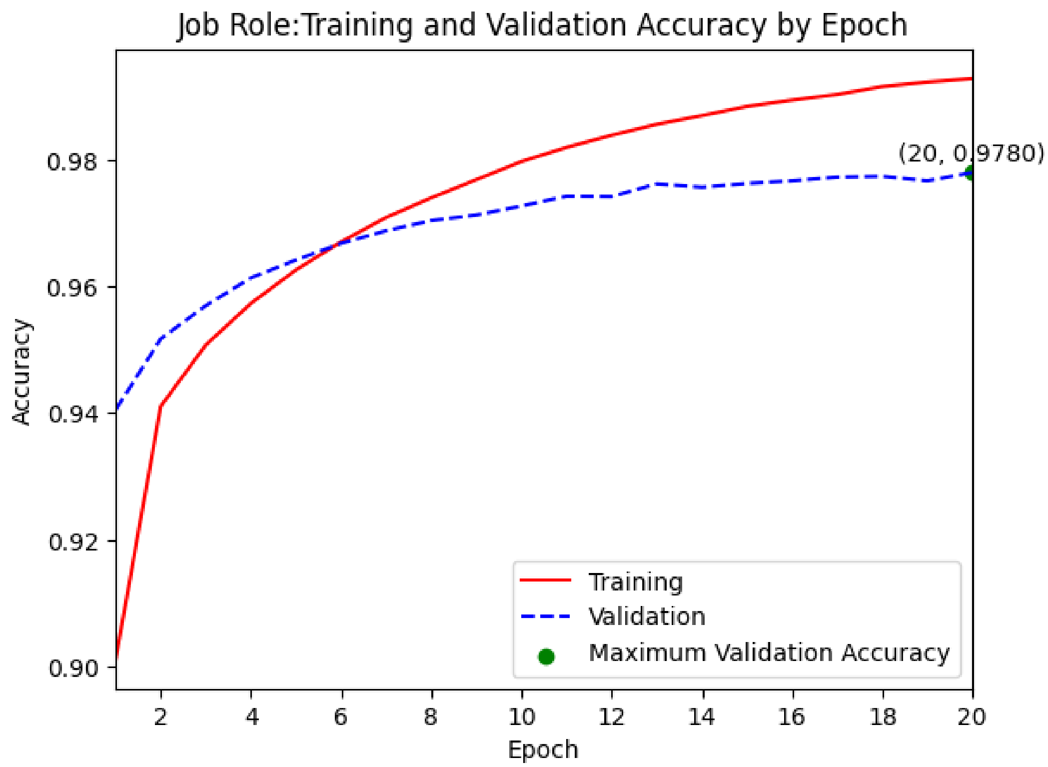
Value	Proportion	Integer Encoding
CONTRIBUTOR	0.3700	1
MANAGER	0.2087	4
DIRECTOR	0.1701	2
EXECUTIVE	0.1318	3
C-LEVEL	0.1155	0
UNKNOWN	0.0039	5

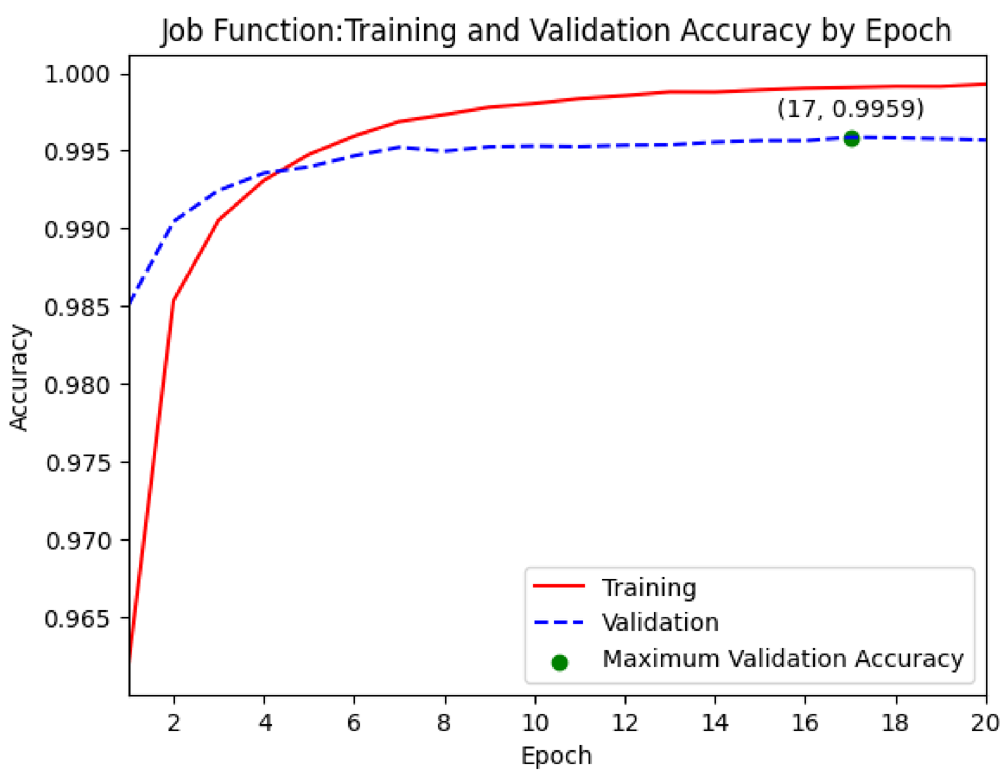
Appendix D
Selected Model Architecture

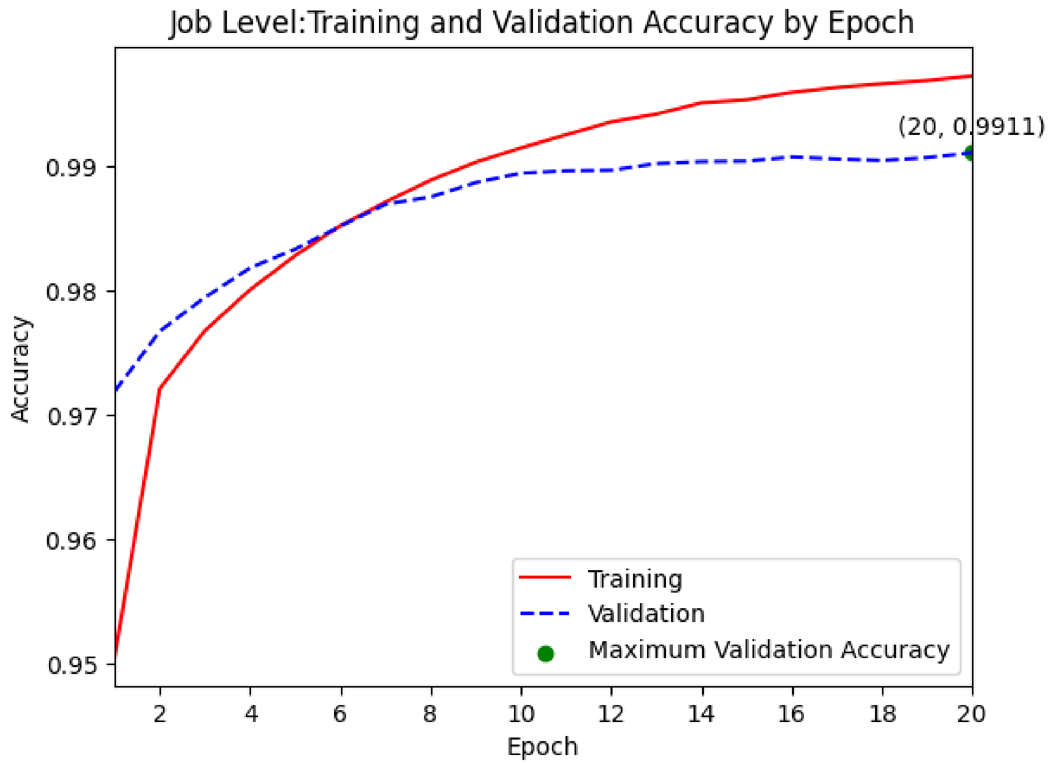
```
DistilBERTClass(
  (l1): DistilBertModel(
    (embeddings): Embeddings(
      (word_embeddings): Embedding(30522, 768, padding_idx=0)
      (position_embeddings): Embedding(512, 768)
      (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
    (transformer): Transformer(
      (layer): ModuleList(
        (0-5): 6 x TransformerBlock(
          (attention): MultiHeadSelfAttention(
            (dropout): Dropout(p=0.1, inplace=False)
            (q_lin): Linear(in_features=768, out_features=768, bias=True)
            (k_lin): Linear(in_features=768, out_features=768, bias=True)
            (v_lin): Linear(in_features=768, out_features=768, bias=True)
            (out_lin): Linear(in_features=768, out_features=768, bias=True)
          )
          (sa_layer_norm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
          (ffn): FFN(
            (dropout): Dropout(p=0.1, inplace=False)
            (lin1): Linear(in_features=768, out_features=3072, bias=True)
            (lin2): Linear(in_features=3072, out_features=768, bias=True)
            (activation): GELUActivation()
          )
          (output_layer_norm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
        )
      )
    )
  )
  (pre_classifier_role): Linear(in_features=768, out_features=768, bias=True)
  (pre_classifier_function): Linear(in_features=768, out_features=768, bias=True)
  (pre_classifier_level): Linear(in_features=768, out_features=768, bias=True)
  (dropout): Dropout(p=0.3, inplace=False)
  (classifier_role): Linear(in_features=768, out_features=7, bias=True)
  (classifier_function): Linear(in_features=768, out_features=5, bias=True)
  (classifier_level): Linear(in_features=768, out_features=6, bias=True)
)
```

Augmented DistilBERT model for training. The thick blue box on top corresponds to the portion of the model imported directly from the pretrained DistilBERT base model: an embeddings layer with both word and position embeddings with normalization and dropout, followed by a transformer block of 6 encoding layers. The first hidden state (representative of the entire sequence) is then passed to the adapted portion in the thin red box on the bottom, which takes the 768-dimensional hidden state and passes it through 3 different feed forward pre-classifier layers in parallel (one for Role, one for Function, and one for Level). At this point, a ReLU activation (not shown) is applied, 30% of the parameters are randomly dropped (only during training, dropout layers do not apply in evaluation/inference mode), before being fed through parallel classification layers for Role, Function, and Level in order to ouput the correct dimensions of 7, 5, and 6 respectively.

Appendix E
Accuracy Curves by Output Field






Job Level:Training and Validation Accuracy by Epoch
(20, 0.9911)
0.99
0.98
0.97
0.96
0.95
Accuracy
Training
Validation
Maximum Validation Accuracy
2
4
6
8
10
12
14
16
18
20
Epoch

Appendix F
Test Set Confusion Matrices, Accuracy, and F1 Scores

Job Role (Accuracy: 0.9719)

		Actual						
		DEVELOPMENT	GOVERNANCE RISK COMPLIANCE	INFORMATION SECURITY	IT GENERAL	NETWORKING	NON-ICP	SYSTEMS
Predicted	DEVELOPMENT	8036	2	43	64	51	54	18
	GOVERNANCE RISK COMPLIANCE	2	1984	85	18	9	15	3
	INFORMATION SECURITY	42	40	26544	236	121	106	38
	IT GENERAL	47	8	92	3950	68	51	31
	NETWORKING	61	12	241	167	21370	120	31
	NON-ICP	39	5	100	93	127	18251	19
	SYSTEMS	13	1	26	30	13	16	1231

Class	F1 Score	Precision	Recall
DEVELOPMENT	0.9736	0.9719	0.9752
GOVERNANCE RISK COMPLIANCE	0.9520	0.9376	0.9669
INFORMATION SECURITY	0.9784	0.9785	0.9784
IT GENERAL	0.8972	0.9301	0.8666
NETWORKING	0.9767	0.9713	0.9821
NON-ICP	0.9800	0.9794	0.9806
SYSTEMS	0.9115	0.9256	0.8979
Global Average	0.9528	0.9563	0.9497

Job Function (Accuracy: 0.9955)

		Actual				
		ENGINEERING	**IT**	**NON-ICP**	**PROCUREMENT**	**RISK/LEGAL/COMPLIANCE**
Predicted	**ENGINEERING**	7870	16	23	0	0
	IT	31	59138	169	1	2
	NON-ICP	20	97	15947	1	3
	PROCUREMENT	2	0	6	241	0
	RISK/LEGAL/COMPLIANCE	0	2	2	0	153

Class	F1 Score	Precision	Recall
ENGINEERING	0.9942	0.9951	0.9933
IT	0.9973	0.9966	0.9981
NON-ICP	0.9900	0.9925	0.9876
PROCUREMENT	0.9797	0.9679	0.9918
RISK/LEGAL/COMPLIANCE	0.9714	0.9745	0.9684
Global Average	0.9865	0.9853	0.9878

Job Level (Accuracy: 0.9875)

		Actual					
		C-LEVEL	CONTRIBUTOR	DIRECTOR	EXECUTIVE	MANAGER	UNKNOWN
Predicted	C-LEVEL	9554	28	10	38	5	6
	CONTRIBUTOR	21	30861	73	64	69	125
	DIRECTOR	25	63	14067	34	24	16
	EXECUTIVE	54	40	38	10775	11	29
	MANAGER	13	135	28	12	17282	34
	UNKNOWN	7	38	3	0	6	136

Class	F1 Score	Precision	Recall
C-LEVEL	0.9893	0.9910	0.9876
CONTRIBUTOR	0.9895	0.9887	0.9902
DIRECTOR	0.9890	0.9886	0.9893
EXECUTIVE	0.9854	0.9843	0.9865
MANAGER	0.9903	0.9873	0.9934
UNKNOWN	0.5075	0.7158	0.3931
Global Average	0.9085	0.9426	0.8900

Appendix G
Training Observations with Top Weighted Loss

		Actual			Predicted			
	Title	Role	Function	Level	Role	Function	Level	Loss
377261	ÈNG	NON-ICP	NON-ICP	CONTRIBUTOR	DEVELOPMENT	ENGINEERING	CONTRIBUTOR	17.035435
281934	SR.PM	NON-ICP	NON-ICP	CONTRIBUTOR	DEVELOPMENT	ENGINEERING	MANAGER	14.895061
651377	SR.PM	NON-ICP	NON-ICP	CONTRIBUTOR	DEVELOPMENT	ENGINEERING	MANAGER	14.895061
584419	NETWORK SECURITY ENGINEER	INFORMATION SECURITY	IT	CONTRIBUTOR	NETWORKING	IT	CONTRIBUTOR	12.008307
310351	IT VICE PRESIDENT, INFORMATION SECURITY	INFORMATION SECURITY	IT	C-LEVEL	INFORMATION SECURITY	IT	EXECUTIVE	11.363048
127468	SENIOR MANAGER, INFORMATION TECHNOLOGY INFRAST...	INFORMATION SECURITY	IT	UNKNOWN	INFORMATION SECURITY	IT	MANAGER	10.772053
231917	ASSISTANT VP, NETWORK SECURITY OPERATIONS	INFORMATION SECURITY	IT	EXECUTIVE	NETWORKING	IT	CONTRIBUTOR	10.501913
176880	MANAGER, COMPUTING INFRASTRUCTURE	NON-ICP	IT	MANAGER	NETWORKING	IT	MANAGER	10.376395
21019	CTO / CSO	SYSTEMS	IT	EXECUTIVE	NETWORKING	IT	C-LEVEL	10.323332
489206	DIRECTOR, TECHNOLOGY SECURITY	INFORMATION SECURITY	IT	DIRECTOR	NETWORKING	IT	DIRECTOR	10.300467
349551	ACCOUNT MANAGER, OPERATIONS	NETWORKING	NON-ICP	CONTRIBUTOR	NON-ICP	NON-ICP	CONTRIBUTOR	10.280710
282273	DIRECTOR, INFORMATION SECURITY & COMPLIANCE, GRC	NETWORKING	IT	DIRECTOR	GOVERNANCE RISK COMPLIANCE	IT	DIRECTOR	10.186845
527554	SR.COMPLIANCE OFFICER	GOVERNANCE RISK COMPLIANCE	IT	EXECUTIVE	GOVERNANCE RISK COMPLIANCE	IT	C-LEVEL	10.081469
451748	VICE PRESIDENT, CYBER SECURITY & SENIOR RISK E...	IT GENERAL	IT	EXECUTIVE	INFORMATION SECURITY	IT	EXECUTIVE	10.063582
27672	SR. SOLUTIONS CONSULTANT VEC BUSINESS UNIT	NON-ICP	NON-ICP	CONTRIBUTOR	NETWORKING	IT	CONTRIBUTOR	9.883533
203315	SENIOR MANAGER, QUALITY MANAGEMENT	DEVELOPMENT	NON-ICP	MANAGER	NON-ICP	NON-ICP	MANAGER	9.842991
148406	IT SYSTEMS & SECURITY ADMINISTRATOR	INFORMATION SECURITY	IT	DIRECTOR	INFORMATION SECURITY	IT	CONTRIBUTOR	9.741222
147972	SENIOR MANAGER, SECURITY SERVICES	INFORMATION SECURITY	IT	UNKNOWN	INFORMATION SECURITY	IT	MANAGER	9.695919
289153	DATA SCIENCE MANAGER, INFORMATION TECHNOLOGY	IT GENERAL	IT	MANAGER	NON-ICP	IT	MANAGER	9.692524
82869	SENIOR DIRECTOR, CORPORATE TECHNOLOGY	INFORMATION SECURITY	IT	DIRECTOR	NETWORKING	IT	DIRECTOR	9.690509

Notes:

1) The first observation appears to be in a foreign language due to the accent mark over the E, thus the inaccurate prediction makes sense.
2) ”SR PM” was inaccurately categorized by the model
3) ”NETWORK SECURITY ENGINEER” differed on Role, with both the model’s result and the actual result appearing plausible.
4) ”IT VICE PRESIDENT, INFORMATION SECURITY” was clearly not C-Level; the model is correct
5) ”SENIOR MANAGER, INFORMATION TECHNOLOGY INFRAST...” was likely a manager; the model appears to be correct
6) ”ASSISTANT VP NETWORK SECURITY OPERATIONS” was most likely an executive; the model appears to be incorrect and is likely thrown off by the ”ASSISTANT” token, which is the highest-ranking keyword for the ”CONTRIBUTOR” level
7) ”MANAGER, COMPUTING INFRASTRUCTURE” is definitely an ICP Role; the model is correct

8) Whether or not ”CTO/CSO” is Networking/Systems is unclear, but it is definitely still a C-LEVEL; the model is probably correct here
9) ”DIRECTOR, TECHNOLOGY SECURITY” is a toss-up; either label appears plausible.
10) ”ACCOUNT MANAGER, OPERATIONS” is clearly a non-technical role and thus NON-ICP; the model is correct
11) ”DIRECTOR, INFORMATION SECURITY & COMPLIANCE, GRC” has ”COMPLIANCE” in it, thus the model appears to be correct
12) ”SR.COMPLIANCE OFFICER” appears to not be C-LEVEL; the model is incorrect
13) ”VICE PRESIDENT, CYBER SECURITY & SENIOR RISK F...” is security-focused; the model is correct
14) ”SR. SOLUTIONS CONSULTANT VEC BUSINESS UNIT” is likely NON-ICP; the model is incorrect
15) ”SENIOR MANAGER, QUALITY MANAGEMENT” is likely NON-ICP; the model is correct
16) ”IT SYSTEMS & SECURITY ADMINISTRATOR” is likely higher than CONTRIBUTOR; the model is incorrect
17) ”SENIOR MANAGER, SECURITY SERVICES” is likely a manger; the model is correct
18) ”DATA SCIENCE MANAGER, INFORMATION TECHNOLOGY” is probably more of a DEVELOPMENT Rolee, but neither the model nor the data is actually correct here
19) ”SENIOR DIRECTOR, CORPORATE TECHNOLOGY” could be either INFORMATION SECURITY or NETWORKING for its role, this seems to be a toss-up

APPENDIX H
TOP 5 KEYWORD TOKENS BY PREDICTED CATEGORY

Role: DEVELOPMENT

Token	C	$P(I > 1)$	σ_I	μ_I
EXPERIENCE	342	0.6754	1.9229	2.3867
ENGINEERING	5555	0.8185	1.2887	2.3008
CONTINUITY	146	0.9521	0.8019	2.2415
SC	121	0.8347	1.3948	2.1759
ENGINEER	39117	0.9291	0.7801	2.0992

Role: GOVERNANCE RISK COMPLIANCE

Token	C	$P(I > 1)$	σ_I	μ_I
COMPLIANCE	10479	0.7732	1.6837	2.5297
CIS	162	0.5802	2.1752	2.1348
/	1049	0.7731	0.8835	1.7443
REGULATORY	142	0.5915	1.6323	1.7066
GOVERNANCE	1901	0.5886	1.4753	1.6895

Role: INFORMATION SECURITY

Token	C	$P(I > 1)$	σ_I	μ_I
SECURITY	78803	0.8831	1.0418	2.1422
SEC	871	0.8599	1.0119	2.1316
ADVISORY	445	0.7079	1.4526	1.9459
MIS	105	0.7714	1.0363	1.8400
STRATEGYCURITY	104	0.6346	1.5682	1.8231

Role: IT GENERAL

Token	C	$P(I > 1)$	σ_I	μ_I
FACILITIES	237	0.6540	1.6641	2.0193
TECHNOLOGIES	211	0.8578	0.9675	2.0138
TECHNICIAN	701	0.8531	0.7244	1.8552
DIGITAL	972	0.7356	1.1589	1.8206
CLOUD	612	0.7696	0.9442	1.7335

Role: NETWORKING

Token	C	$P(I > 1)$	σ_I	μ_I
INSPECTOR	102	0.6961	2.2769	2.7104
NETWORK	51458	0.9082	1.2321	2.5156
NETWORKS	1026	0.7173	1.9112	2.5054
MOBILE	548	0.6624	1.7687	2.1379
ARCHITECT	32578	0.8752	1.0342	2.1133

Role: NON-ICP

Token	C	$P(I > 1)$	σ_I	μ_I
SUPPORT	7964	0.8650	1.2285	2.4125
SOX	152	0.7105	1.6291	2.3351
REGULATORY	272	0.8125	1.2351	2.1729
HR	528	0.6723	1.6236	2.1579
AUDITOR	2179	0.7581	1.0753	2.1158

Role: SYSTEMS

Token	C	$P(I > 1)$	σ_I	μ_I
COMPUTING	184	0.8587	1.6771	3.1620
SAP	688	0.8517	1.3136	2.3870
DATABASE	161	0.8634	1.1666	2.1404
CHAIN	220	0.8955	0.9582	2.1247
SHARED	195	0.6821	1.4386	2.0550

Function: ENGINEERING

Token	C	$P(I > 1)$	σ_I	μ_I
ENGINEERING	8302	0.8055	1.8191	3.1884
USER	550	0.7782	1.8084	2.9094
SC	103	0.9029	1.5281	2.8184
ACQUISITION	143	0.7832	1.9257	2.8169
ENGINEER	36543	0.9307	1.0052	2.7128

Function: IT

Token	C	$P(I > 1)$	σ_I	μ_I
CLOUDOPS	148	0.9122	1.7494	3.4616
COMPUTING	788	0.6942	2.3129	2.7683
NETWORKS	1161	0.6787	2.1205	2.6626
NETWORK	55276	0.8741	1.4328	2.5110
CLOUD	9174	0.7141	1.8551	2.3470

Function: NON-ICP

Token	C	$P(I > 1)$	σ_I	μ_I
SOX	174	0.7759	1.6984	2.8556
WORKFORCE	104	0.7596	1.6203	2.5124
HR	537	0.6760	1.8565	2.4508
FEDERAL	897	0.7492	1.6728	2.3373
REGULATORY	207	0.8019	1.3736	2.2757

Function: PROCUREMENT

Token	C	$P(I > 1)$	σ_I	μ_I
VENDOR	301	0.8306	1.8140	3.1192
PROCUREMENT	634	0.7965	1.4775	2.4946
PURCHASING	193	0.8290	0.8402	2.0923
STRATEGIC	135	0.5926	1.7337	2.0413
CONTRACTS	139	0.7770	0.9129	1.8423

Function: RISK LEGAL COMPLIANCE; note that this category only has 4 keywords due to sparse occurrence in the data

Token	C	$P(I > 1)$	σ_I	μ_I
COUNSEL	612	0.7582	1.8677	2.5337
LEGAL	190	0.6579	1.6137	2.2000
ATTORNEY	100	0.6300	1.4923	2.0250
AFFAIRS	129	0.6434	1.0544	1.4424

Level: C-LEVEL

Token	C	$P(I > 1)$	σ_I	μ_I
MANAGING	2608	0.8232	1.4067	1.9321
CHIEF	33243	0.5455	1.9126	1.8665
DC	110	0.4636	2.0206	1.6889
BOARD	273	0.7692	1.1379	1.5691
FINANCIAL	862	0.6821	1.4002	1.4915

Level: CONTRIBUTOR

Token	C	$P(I > 1)$	σ_I	μ_I
ASSISTANT	4480	0.7196	1.9783	2.5618
ADVISORY	433	0.7760	1.6962	2.4613
SENIORSECU	109	0.9358	1.2885	2.2809
ADMINISTRATIVE	514	0.7782	1.5843	2.2622
DH	162	0.4198	2.8010	2.0324

Level: DIRECTOR

Token	C	$P(I > 1)$	σ_I	μ_I
DIRECTORY	169	0.9053	1.2659	2.9705
DIRECTOR	105562	0.7906	1.4671	2.6474
DIR	1374	0.7096	1.5266	2.3240
DIRECTORSECURITY	798	0.9511	0.5741	2.1445
DESIGN	219	0.4247	2.2401	2.0183

Level: EXECUTIVE

Token	C	$P(I > 1)$	σ_I	μ_I
HEAD	6044	0.8104	2.0013	3.4082
VP	21153	0.7602	1.4800	2.5287
EXECUTIVE	10217	0.8185	1.7190	2.5132
EX	128	0.7422	1.9307	2.4044
VICE	27538	0.8067	1.1352	1.8782

Level: MANAGER

Token	C	$P(I > 1)$	σ_I	μ_I
MANAGERCING	175	0.8343	1.5635	2.8635
LEADER	2284	0.8266	1.7588	2.7828
LEAD	12859	0.9034	1.5055	2.7384
SUPERVISOR	4767	0.8393	1.4132	2.7321
MANAGER	100159	0.8407	1.4538	2.6779

Level: UNKNOWN; note that this category only has 4 keywords due to sparse occurrence in the data

Token	C	$P(I > 1)$	σ_I	μ_I
SUPPORT	119	0.7731	0.8544	1.5572
CLOUD	146	0.8082	0.4475	1.2369
SPECIALIST	119	0.6218	0.6842	1.1207
IT	165	0.6242	0.5145	1.0090

APPENDIX I
TOP 5 ANTI-KEYWORD TOKENS BY PREDICTED CATEGORY

Role: DEVELOPMENT

Token	C	$P(I > 1)$	σ_I	μ_I
NETWORKING	1689	0.7324	1.5150	2.2326
PRESIDENTSE	227	0.8326	0.9383	2.1117
LIAISON	171	0.8363	1.1678	2.0265
OPERATING	723	0.7109	1.3408	2.0245
.SE	209	0.8086	0.9745	2.0080

Role: GOVERNANCE RISK COMPLIANCE

Token	C	$P(I > 1)$	σ_I	μ_I
OPIONSERT	240	1.0000	0.0000	3.0000
OPERATIONSSECURITY	185	0.8000	1.6871	2.8752
OPPORTUNITY	111	0.8108	1.3489	2.3951
MASTER	296	0.5878	2.2590	2.1860
OPERATOR	271	0.7675	1.8345	2.1455

Role: INFORMATION SECURITY

Token	C	$P(I > 1)$	σ_I	μ_I
ENGINEERPOINT	101	0.8515	1.5465	2.7461
SOX	178	0.7472	1.8198	2.6130
CHAIN	559	0.8318	1.4841	2.3174
ENGINEER	62146	0.8665	1.0859	2.2825
EQUAL	107	0.8692	0.8194	2.2458

Role: IT GENERAL

Token	C	$P(I > 1)$	σ_I	μ_I
SOX	178	0.8371	1.3377	2.6250
CHANCELLOR	143	0.8951	1.4265	2.4548
PARTNERSHIPS	398	0.7764	1.5447	2.3803
NETWORKING	1704	0.7735	1.5266	2.1816
SUPERVISORY	177	0.7966	1.2608	2.0719

Role: NETWORKING

Token	C	$P(I > 1)$	σ_I	μ_I
ELECTRONIC	134	0.7910	2.0705	2.8737
ENGINEERPOINT	110	0.9273	1.3385	2.8320
EXPERIENCE	541	0.7375	1.9470	2.7203
INVENTORY	378	0.7143	1.7393	2.5450
COMPLIANCE	11795	0.6903	1.9485	2.4149

Role: NON-ICP

Token	C	$P(I > 1)$	σ_I	μ_I
.SE	209	0.9665	0.9113	3.4588
##SECU	689	0.9695	0.4555	2.9091
CLOUDOPS	143	0.7902	1.7670	2.7428
INSPECTOR	117	0.5812	2.7335	2.4802
PROGRAMMER	1092	0.8059	1.2512	2.3907

Role: SYSTEMS

Token	C	$P(I > 1)$	σ_I	μ_I
PRIVACY	2649	0.7599	2.4418	3.0348
CHANCELLOR	146	0.7466	1.9972	2.6303
OPPORTUNITY	111	0.8559	1.2042	2.4773
HEADSE	112	0.9554	0.8472	2.3668
DIRECTORSE	970	0.7258	1.5858	2.1353

Function: ENGINEERING

Token	C	$P(I > 1)$	σ_I	μ_I
CLOUDOPS	148	0.8919	1.7701	3.4832
CLOUD	9182	0.8086	1.7894	2.6526
COMPUTING	792	0.6717	2.2614	2.5349
NETWORK	55427	0.7996	1.4337	2.2344
PRESIDENTSE	230	0.8261	1.1616	2.1455

Function: IT

Token	C	$P(I > 1)$	σ_I	μ_I
SOX	174	0.8103	1.6391	3.0096
ENGINEERING	8863	0.7901	1.7499	2.9090
VENDOR	334	0.8293	1.6092	2.6739
OPPORTUNITY	111	0.8739	1.1014	2.5006
ENGINEER	38703	0.8958	1.0620	2.4954

Function: NON-ICP

Token	C	$P(I > 1)$	σ_I	μ_I
PROGRAMMER	1093	0.8509	1.1529	2.3947
ENGINEERINGSECURITY	109	0.8532	1.4673	2.3074
COUNSEL	712	0.6994	1.8629	2.2846
HEADSE	112	0.7946	1.4564	2.2152
ENGINEERING	13484	0.7165	1.7304	2.2047

Function: PROCUREMENT

Token	C	$P(I > 1)$	σ_I	μ_I
)O	454	0.8238	2.8208	3.6797
CONTROL	1575	0.5537	2.8754	2.6856
COMPUTING	807	0.6741	2.3624	2.6768
OVERSIGHT	230	0.5130	3.1570	2.6150
NETWORKS	1187	0.6580	2.1569	2.5877

Function: RISK LEGAL COMPLIANCE

Token	C	$P(I > 1)$	σ_I	μ_I
OVERSIGHT	233	0.6223	2.6342	2.5562
NETWORKS	1187	0.6672	2.0631	2.5465
INVENTORY	391	0.6675	1.8980	2.4691
INFORMATICS	317	0.6814	1.9289	2.3926
ENGINEEROPS	659	0.9712	0.7295	2.3203

Level: C-LEVEL

Token	C	$P(I > 1)$	σ_I	μ_I
OPIONSERT	240	1.0000	0.0000	3.0000
SENIORSECURITY	149	0.7987	1.4989	2.7944
VICE	29097	0.8194	1.7709	2.7724
CONSULTANTCURITY	313	0.7668	1.3439	2.4059
EMERGENCY	185	0.5946	2.3388	2.2778

Level: CONTRIBUTOR

Token	C	$P(I > 1)$	σ_I	μ_I
HEAD	7306	0.7974	1.8513	2.8498
HEADSE	112	0.8571	1.4162	2.7833
LEAD	13493	0.8871	1.3461	2.7146
DEAN	136	0.6397	2.2040	2.5394
LEADER	2673	0.8100	1.5911	2.5258

Level: DIRECTOR

Token	C	$P(I > 1)$	σ_I	μ_I
EXECUTIVESE	102	0.7353	2.8832	3.8853
MANAGING	3557	0.8513	1.8939	2.6451
MANAGERCING	179	0.7430	1.6743	2.3409
##SECU	684	0.9766	0.3359	2.1760
SENIORSECU	110	0.7273	1.2775	2.1198

Level: EXECUTIVE

Token	C	$P(I > 1)$	σ_I	μ_I
OPERATIONSSECURITY	163	0.8405	1.4858	2.6755
MANAGERCING	177	0.8192	1.5500	2.6297
SENIORSECURITY	139	0.7986	1.5451	2.5048
COACH	144	0.6667	1.9010	2.3884
CONTROL	1398	0.4077	2.9870	2.3089

Level: MANAGER

Token	C	$P(I > 1)$	σ_I	μ_I
HEAD	7373	0.7532	2.2180	3.1483
CHIEFSE	101	0.9010	1.4093	3.1414
EXECUTIVESE	103	0.7864	1.9285	2.9120
HEADSE	112	0.6250	2.1030	2.4986
JUNIOR	226	0.8496	1.1032	2.4227

Level: UNKNOWN

Token	C	$P(I > 1)$	σ_I	μ_I
HEADSE	112	0.9732	2.0046	3.8974
EXECUTIVESE	103	0.7864	2.5330	3.1981
CHIEFSE	101	0.8812	1.4551	2.8404
HEAD	7403	0.7764	1.8401	2.5726
OH	105	0.5714	2.1646	2.3836