3-2022

Evaluating Semantic Matching Techniques for Technical Documents

Rain F. Dartt

EVALUATING SEMANTIC MATCHING TECHNIQUES FOR TECHNICAL DOCUMENTS

THESIS

Rain F. Dartt, Captain, USSF

AFIT-ENG-MS-22-M-021

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENG-MS-22-M-021

EVALUATING SEMANTIC MATCHING TECHNIQUES FOR TECHNICAL DOCUMENTS

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Master of Science in Electrical Engineering

Rain F. Dartt, B.S.E.E.

Captain, USSF

March 24, 2022

AFIT-ENG-MS-22-M-021

EVALUATING SEMANTIC MATCHING TECHNIQUES FOR TECHNICAL DOCUMENTS

THESIS

Rain F. Dartt, B.S.E.E.
Captain, USSF

Committee Membership:

Gilbert L. Peterson, Ph.D
Chair

George E. Noel, Ph.D
Member

Richard Dill, Ph.D
Member

AFIT-ENG-MS-22-M-021

Abstract

Machine learning models that employ natural language processing (NLP) techniques have become more widely accessible, making them an attractive solution for text and document classification tasks traditionally accomplished by humans. Two such use cases are matching the specialized experience required for a job to statements in applicant resumes, and finding and labelling clauses in legal contracts The Air Force Materiel Command (AFMC) has an immediate need for solutions to civilian hiring. However, there is currently no truth data to validate against. A similar task is contract understanding for which there is the Contract Understanding Atticus Database (CUAD), a recently published repository of 510 contracts manually labelled by legal experts. The presented semantic matching approach first extracts, preprocesses and embeds contract clauses into a 512-dimesnion term frequency-inverse document frequency (TF-IDF) feature vector. Four logistic models are trained on a subset of these vectors. Then, the models are tuned to accept the contracts as text documents split into sliding windows of words. Next, the model performances are measured on a previously isolated test set and compared against the transformer models employed in the original CUAD research. The multinomial classifier with an L2 penalty term yielded the highest accuracy of 45.08% on the test set. Finally, this research closes with suggestions on how to apply these techniques to the domain of resume reviewing.

Table of Contents

Page

List of Figures

List of Tables

EVALUATING SEMANTIC MATCHING TECHNIQUES FOR TECHNICAL DOCUMENTS

I. Introduction

1.1 Problem Background

Recruiting qualified individuals is a key human resources (HR) function for an organization to survive and succeed. In large organizations that have a centralized hiring process, HR departments must be able to handle a high volume and variety of job applications. Some postings have role-specific requirements or technical terminology that can be challenging for HR professionals to manage. Recently, developments in artificial intelligence (AI) and natural language processing (NLP) enabled the development of recruiting software tools and services that can assist in the process. For example, an Applicant Tracking System (ATS) can collect, organize, and rank résumés according to filters or keywords inputted by recruiters and hiring managers; as of 2018, over 98% of Fortune 500 companies were using an ATS such as Oracle's *Taleo* or Workday's *Human Capital Management* system [1]. On average, it takes 7 days to screen applicants for each job posting, accounting for about 20% of the 36-day time-to-fill for an opening [2].

The Department of the Air Force (DAF) employs about 170,000 civilians alongside its active duty, guard, and reserves members. The Air Force Personnel Center (AFPC) and dedicated individuals within Air Force Materiel Command (AFMC) provide many HR services for the DAF; their personnel servicing ratio of 1-to-325 (0.3%) is about one-third the standard ratio of 1.4% [3][4]. In 2021, AFMC servicing teams

completed nearly 12,500 hiring actions in an average of 63 days, a great improvement over previous years yet still lagging behind the private sector [5]. There are many on-going initiatives within AFMC to further improve the recruitment process, one of which is developing a better ATS since federal regulations limit the availability of commercial off-the-shelf options. Currently, the hiring process relies on human technicians who are not trained or familiar with field-specific jargon that often appears in these résumés, so misclassification is too common, resulting in time wasted on false positives and accusations of unfairness from false negatives. This thesis intended to focus only on the specialized experience open response text field, AFMC is investigating the process as a whole to find opportunities for increased accuracy or timeliness through automation in an ATS. However, due to constraints regarding availability of data, an analogous dataset of legal contracts is utilized in lieu of résumés.

1.2 Motivation

The motivation behind this research is to improve the hiring process of AFPC by developing AI tools that identify whether candidates for a job posting are qualified or not based on the free text responses for specialized experience. This is modeled as a machine learning classification problem where each sample of text can be labeled according to which, if any, requirement is satisfied by the text. The thesis will compare the performance of different multiclass classification techniques on a specialized, technical corpus of annotated text samples. Since truth data is not available for the primary problem of job postings and résumés, the thesis will use the recently published Contract Understanding Atticus Database (CUAD) of legal contracts [6].

The CUAD and the supervised learning models tested on this dataset are representative of those that will support the sponsor's goals of classifying text samples from résumés compared to experience requirements found in job postings. The dataset

serves as an analogy of job résumés submitted to AFPC for technical positions. Models that are effective at classifying these contract clauses (laden with legal terminology) will hopefully also be effective at matching sentences within résumés to job requirements listed in the posting. These models will serve as a proof-of-concept to eventually be adapted as part of an automated ATS that assists technicians involved with the AFPC hiring process. By reducing the manpower burden of the technicians while simultaneously improving the ability to identify qualified candidates, AFPC will shorten the on-boarding process in support of recruitment efforts.

1.3 Hypothesis

Text classification models have a difficult time with a large number of classes. While the CUAD is a great analog for technical text compared to other common language repositories, the small total number of samples (7,000) across over 40 classes must be split into training and test sets. The best transformer model in the original paper had an Area Under Precision-Recall curve (AUPR) score of 47.% with a precision of 44% at 80% recall. Since this thesis will explore much simpler and less powerful models than transformers, having a precision of 40% would be sufficient for this proof-of-concept. The overall intent is not to be a perfect classifier that acts independently but to be a tool that assists a human HR professional by suggesting labels. Additionally, these models need to train quickly to account for projected updates that would be necessary to keep pace with changes in job requirements and technical jargon that matches. A Python-based *sklearn* logistic regression model should be capable of meeting both of these goals.

1.4 Research Goals

The model selected must meet these research goals:

- Logistic regression model classifies the test set with at least 40% accuracy
- Logistic regression model trains in less than 30 minutes

1.5 Approach

Four candidate logistic regression models will be trained on the truth data provided in the CUAD: a comma-separated values (CSV) containing labeled clauses extracted from the contracts. Then, they will have hyperparameters tuned to accommodate ingesting the contracts themselves. Lastly, the test set of contracts will be used to compare classification performance of these models to each other as well as to the models in the original CUAD paper using accuracy of model predictions and confusion matrices. The *sklearn* models are summarized below:

- Logistic regression with cross-validation and a liblinear solver, one-vs-rest
- Logistic regression with an L1 penalty and saga solver, multinomial
- Logistic regression with an L2 penalty and saga solver, multinomial
- Logistic regression with an L2 penalty and saga solver, one-vs-rest

1.6 Results

The L2 multinomial model was able to train and tune hyperparameters in about 20 minutes and achieve an accuracy of 45.06% on sliding windows of the contracts from the test set. However, variances in the methods used to produce a modified 'ground truth' for the sliding window representation obscure a direct comparison to the transformers from the CUAD paper.

1.7 Document Overview

This chapter introduced the problem and proposed a hypothesis and goals to address it. Chapter II details some of the applicable concepts and research related to

machine learning and text classification. Chapter III explains the pipeline for training the models and preparing them for testing. Chapter IV shows the performance of the models and demonstrates that the L2 multinomial model was superior. Finally, Chapter IV contains conclusions about the current work and suggestions for how this research can be applied to the ongoing AFPC efforts to improve their hiring process.

II. Background and Literature Review

This chapter outlines the prior works that serve as the foundation for research and experimentation described in the following methodology and analysis chapters. Ideas presented in literature inspired the model and parameter choices taken.

First, there is a summary of natural language processing (NLP) and machine learning supervised classification techniques employed in this research. Next, the chapter includes a literature review of recent research similar to this thesis as well as an explanation of the Contract Understanding Atticus Database (CUAD) utilized.

2.1 Natural Language Processing

In linguistics, natural languages or ordinary languages are languages that have evolved naturally for humans to communicate with each other using words that are spoken, sung, written, or signed. They are distinct from constructed languages (e.g., Esperanto, Klingon) and formal languages (e.g., programming languages). NLP can be broadly described as "any kind of computer manipulation of natural language" that includes applications such as predictive text, machine translations, and word frequency counts [7].

NLP techniques often take a sample of text and convert the words, characters, and other symbols into a numerical format that serves as the input for machine learning models such as logistic regression or neural networks. NLP can also be used to refer to practices such as automatic part-of-speech tagging, named-entity recognition, or sentence diagramming.

2.1.1 Text Preparation and Normalization

Text is typically parsed from files and stored as strings (either in whole or line-by-line which may be concatenated later). Some models (e.g., the universal sentence coder) process paragraph- or document-sized strings. However, many models expect individual words as inputs. Regular expressions are usually effective at isolating words from each other; additionally, they can perform other normalization functions such as removing special characters, converting to lowercase, and expanding contractions.

Other preprocessing steps that are often applied include normalizing words to their base stems or lemmas and removing stop words. A lemmatizer compares words to established dictionary entries while a stemmer follows an algorithm to chop off suffixes and endings. Both methods transform variants of words into a single form so subsequent techniques treat them the same. Stop words are words which are so common (e.g., linking verbs and articles) that they contain little distinctive information, and their inclusion would crowd out other words. The samples are filtered by comparing words against a library of stop words appropriate for the dataset.

2.1.2 Feature Extraction

Feature extraction methods transform text into numerical representations. They are an integral part of solving NLP problems. Basic methods include generating sparse matrices with techniques such as simple word counts and term frequency-inverse document frequency (TF-IDF). These sparse matrices make use of the bag-of-words approach, whereas dense embeddings are designed to capture complex contextual relationships. Once the words are encoded into matrices, it is easier to perform mathematical operations on the text or to use the numbers as features for text classification, semantic similarity, clustering, or other NLP tasks.

2.1.2.1 Word Counts

One of the simplest forms of feature extraction comprises making a vocabulary of unique words found in the training set and counting the occurrence of each word found in each document. Thus, each document becomes an array of integers whose length is the size of the training vocabulary. The array indices correspond to words, and the sum of each array is the total word count. Classification models use this co-occurrence matrix to learn which words are associated with which classes.[8]

Word order and context information are lost in this unigram bag-of-words approach. However, some information can be restored by counting bigrams, trigrams, or any other size N-gram instead of individual words. This comes at the expense of making the model more sensitive to training data since it can only recognize N-grams it has seen before (and N-grams are rarer as N increases).

2.1.2.2 TF-IDF

Another form of encoding is TF-IDF. This modifies the co-occurrence matrix by attaching weights such that words that are unique or heavily associated with a specific class or label have a higher real value representation than words that are more common and have less discriminatory power [9]. An example implementation that assigns weight w_i to term i in sample D is shown in Equation (1) below.

$$w_i = tf_i * idf_i = tf_i * log\frac{N}{df_i} \tag{1}$$

where tf_i is the number of times vocabulary term i occurs in D, idf_i is the inverse document frequency of term i, N is the total number of documents and df_i is the number of documents that contain vocabulary term i.[10]

2.1.2.3 Word Embeddings

Word embeddings are more advanced techniques that preserve semantic and syntactic meaning of words in context. They typically utilize neural networks or transformers and train on a very large corpus of data to develop a distributed representation of words. Depending on model design, this may occur separately and independently, or it can be hidden within the first layers of deeper network model [9]. For example, Google's universal sentence encoder (USE) was pre-trained on sources such as Wikipedia, web news, web question-answer pages and discussion forums; one implementation of the USE averages individual word vectors and passes them through additional neural network layers to achieve an overall vector for the sentence or document [11]. Architectures such as word2vec are tailored to predict words based on surrounding context words or else predict surrounding words given the current word [12]. The Global Vectors (GloVe) word representation model uses ratios of word-word co-occurrence probabilities to determine the similarity among words and capture semantic and syntactic regularities [13]

Similarities can be detected between words by calculating the distance between word embedding vectors. For example, embeddings can preserve the semantic relationships among the words King, Man, Woman, and Queen. With embedded representations, users can perform operations such as King – Man + Woman = Queen, or Queen + Man - Woman = King. Once the data has been extracted to arrays of lemmatized tokens (strings), word embedding techniques convert the strings into numerical representations that are appropriate inputs for machine learning models.

2.2 Supervised Classification

Once words are converted into features, models can use the vectors to predict appropriate labels. Supervised classification models use trusted external information

(labels or targets) about a training dataset to influence updates to weights of model algorithms [14].

2.2.1 Multinomial Naïve Bayes

Naïve Bayes classifiers are a type of probabilistic classifiers that assume independence between features. That is, the model determines the likelihood of a class label given a set of features by multiplying the independent probabilities (determined in the training phase) of the class label per individual feature. In order to prevent a previously-unseen feature from zeroing out the entire product, a smoothing adjustment is sometimes added so that every probability has a minimum representation; using a minimum of one is called Laplace smoothing. The equation for the naïve Bayes classifier is given in Equation (2).

$$f_i^{NB}(x) = \Pi_{j=1}^{n} P(X_j = x_j | C = i) P(C = i) \tag{2}$$

Where f_i^{NB} is a discriminant function for each class; a maximization function would then determine the label based on the highest conditional probability [15].

2.2.2 Random Forest Classifier

A random forest classifier (RFC) tree makes use of a number of regressive decision trees. At each decision point in each tree, a random sample of predictors is selected as candidates for the binary split point. Data separated along split points until either a maximum depth is reached or if further splitting would introduce more errors than answers. BThe randomness prevents the collection of trees from becoming too similar to each other or fixating on only a few strong predictors. An example tree

model function is shown in Equation (3) [16].

$$f(X) = \sum_{m=1}^{M} c_m * 1_{(X \in R)} \tag{3}$$

Where c_m is the node purity and R_m represents a partition of the feature space [16].

2.2.3 Logistic Regression

Logistic regression models are based on finding conditional probabilities that the labels apply to inputs using weights tuned on a training dataset using stochastic gradient descent. The logistic function is given in Equation (4) below [16].

$$p(X) = \frac{e^{\beta_0 + \beta_1 X}}{1 + e^{\beta_0 + \beta_1 X}} \tag{4}$$

The coefficients β_0 and β_1 are found in training by making predictions based on estimated values, calculating the errors, and then adjusting the coefficients accordingly [16].

2.3 Recent Related Research

NLP techniques have a wide range of applications, and research in the area is fact-paced. New techniques and models are constantly being developed, but older algorithms remain relevant and useful with novel modifications or combinations.

2.3.1 Classic Models

Naïve Bayes, random forest, logistic regression, and other older algorithms still see new applications and continued interest. For example, Pranckevičius & Marcinkevičius [17] used these and other models to classify Amazon product reviews according to the 5-star ranking system. The relatively low resource footprint and runtime help

these models remain attractive for many simpler NLP tasks or ones where large-scale training or external information is not feasible.

2.3.2 Neural Networks and Transformers

Deep neural networks and transformers are a popular choice for working with certain NLP contexts. These models can be resource intensive and require large amounts of data to properly train. However, the embedding relationships learned by these models can be extremely powerful in NLP tasks. If possible, integrating external data further improves performance [18]. Transformers are a special kind of network model that uses attention functions instead of convolution or recurrence relationships [19]. The HuggingFace Transformers library is one open-source API easily accessible for Python programmers [20]. LEGAL-BERT is an example of a domain-specific family of transformer models developed to work with legal tasks [21].

Question-answering models in particular have been applied to résumé extraction and job application matching. Researchers at LinkedIn developed a Screening Question Generation task, deployed through their Job2Questions model, to assist employers improving their job postings and searching for qualified applicants [22].

2.3.3 Automated Résumé Analysis

Recent research dealing with résumés or Applicant Tracking Systems (ATSs) spans from keyword searching and named entity recognition (NER) to evaluation and ranking systems. Usually, résumés are formally unstructured but still have predictable patterns, so models can learn how to extract fields like name, address, degree, and contact information; however, more advanced techniques can also look up keywords or sum up years of experience for an industry. These models can range from random forest classifiers and logistic regression to neural networks and transformers.

Most document review models in general can, in practice, also apply to résumés [23][24][25][26].

Some job recruitment and posting platforms such as LinkedIn have integrated automatic résumé and job posting parsing, matching, and ranking. This can be combined with having users answer screening questions or rebuild their résumé with drop-down menus or labeled text boxes. A neural network or transformer on the backend can analyze the data from multiple users and postings in order to refine their models [27][22][28][29].

2.3.4 Published Datasets

A major challenge for this thesis was finding a large enough dataset with ground truth available for supervised learning models. Résumés would be ideal; however, none of the available sources came with reliable and relevant labels. For example, one paper described manually parsing 5,000 résumés into 5 categories with 2,000 samples each, but it lacked sufficient detail in describing this process and the dataset itself was not available [29]. Roy, Chowdhary, and Bhatia [24] reference résumés downloaded from Kaggle, but only one label category, industry sector to which each résumé belonged, was included; however, it was still helpful to view the pipeline for the four models they investigated. Similarly, Tikhonova and Gavrishchuk [30] used a dataset of 5,000 CVs that had a small number of categories (and was also in Russian). The candidate dataset with the highest confidence and relevance was unfortunately not published in English. The authors published a Chinese-language corpus of over 150,000 résumés with 6 label categories manually applied by four annotators [31].

2.4 Contract Understanding Atticus Database

The Contract Understanding Atticus Database (CUAD) is a corpus of legal contracts consisting of text clauses annotated with label categories that was published on 10 March 2021. According to the authors, "The dataset includes more than 500 contracts and more than 13,000 expert annotations that span 41 label categories" [6]. These contracts were collected from the publicly available Electronic Data Gathering, Analysis, and Retrieval system maintained by the U.S. Securities and Exchange Commission. The CUAD most closely matched Air Force Personnel Center (AFPC) interests for this research due to the domain-specific vocabulary of contracts and the level of expert annotation available compared to other potential sources (e.g., unlabeled and unsorted résumés from a too-wide variety of job postings).

The reason the number of annotations is not equal to the product of contracts and label categories is that not every contract includes an entry for every label. The master clauses comma-separated values (CSV) file (manually curated by experts; Appendix A contains a version reformatted for this research effort) contains a row for each contract and two columns for each label category: the first instance contains the phrase or clause from the contract that meets that label criteria while the second either condenses and standardizes it into a format that makes it easier to present (9/41 categories) or states "Yes" or "No" depending on whether such a clause is present (32/41 categories). The existence of any non-blank data demonstrates that the category did apply to the contract in question.

The authors of the CUAD paper employ a variety of language models from the HuggingFace Transformers library [20]. These transformer models use hundreds of millions of parameters and require many hours of training. The models are trained to identify the start and end token positions of substrings within a contract that should be classified as matching one of the 41 label categories previously mentioned.

2.5 Summary

The field of machine learning remains active and challenging. Techniques and models are steadily being developed and improved. With the sheer volume of data streams to manage, automation and artificial intelligence (AI) systems are quickly transitioning from competitive advantages to core practices. AI can streamline or assist tasks previously performed by humans such as image recognition and text translation, but they can also find new applications like automatically suggesting predictive maintenance and performing fault detection, isolation, and resolution.

Specific NLP developments deal with text recognition and classification. Feature extraction methods including TF-IDF and the USE convert raw text into machine-readable formats while preserving different levels of semantic and syntactic context of words. These embedding spaces can use co-occurrence and other learned statistics to assign meaning to a sample text, so classifier models can label it appropriately. While these methods are not perfect or foolproof, they ease the burden of human users to provide a more robust total system.

III. Methodology

This chapter details the process of preparing data, training the models, and investigating hyperparameter values that provide the optimal scores on the data. The types of models chosen and the ranges of parameters were inspired by research conducted and summarized in the previous chapter. Results of the model are discussed in the Chapter IV.

Data preprocessing includes retrieving the documents from the Contract Understanding Atticus Database (CUAD) website and loading them into Python structures like Pandas DataFrames for ease of use. The data is then split into a 80% training and 20% test set, and the test set is isolated from the model until the end. Training data was embedded using a 512-feature term frequency-inverse document frequency (TF-IDF) vectorizer to serve as the input for the models. After the models were trained on the truth data from clauses, another round of tuning occurred based on sliding windows from the text files of the training contracts. Finally, predictions of the test contract windows are made with the optimized models.

3.1 Semantic Match Pipeline

Figure 1 shows an overview pipeline for this experiment. There were three main phases: model training, hyperparameter tuning, and testing. Each utilized a different subset of the CUAD dataset but all shared the same preprocessing, feature extraction, and models. The results of each phase were used by the successive phase(s).

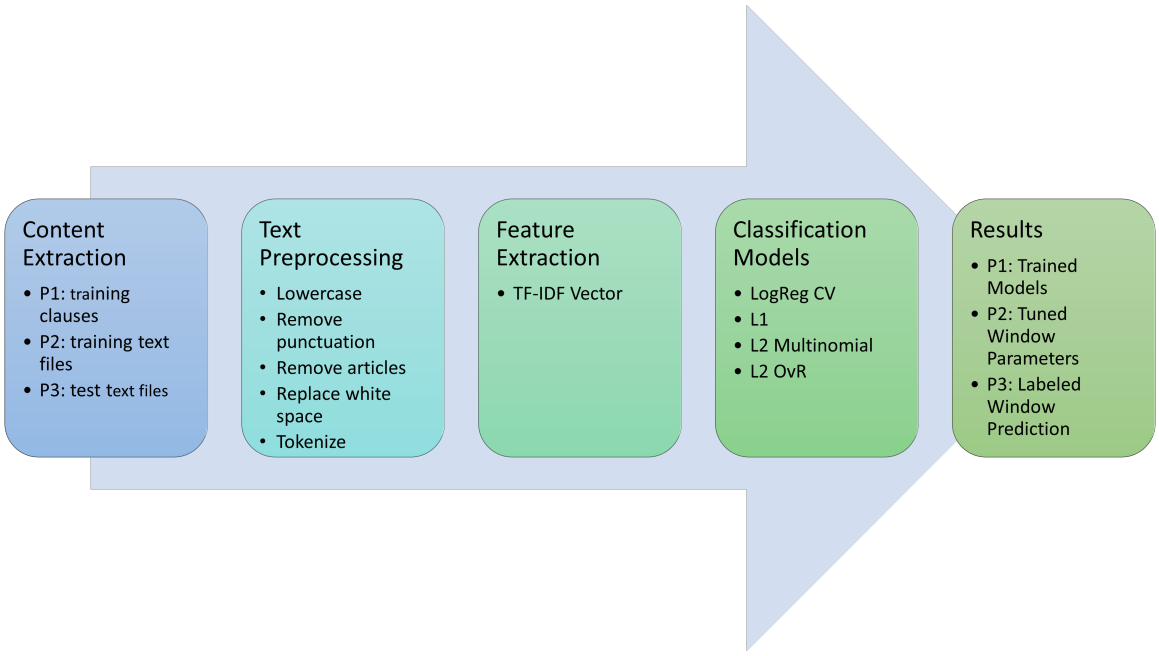

Figure 1: Semantic Match Pipeline

3.2 Preliminary Experimental Models

Preliminary experimental models used a reduced version of the dataset that included only 32 classes, eliminating the 'None' category whose entries were very long, the dates which were very short and too similar to each other, and categories which were predominantly unique proper nouns (and also very short)[32][33]. These models were useful to quickly compare the performance of different feature extraction techniques and classification models before applying the pipeline to the whole dataset. Specifically, a bag-of-words counter, TF-IDF, and the universal sentence encoder (USE) were investigated as feature vector possibilities, and Naïve Bayes, random forest, logistic regression, and a feedforward neural network (FFNN) were considered as candidate classification models. The highest performers were TF-IDF and logistic regression, respectively, so this thesis focuses on their combination.

3.3 CUAD Content Extraction

Data for the models come in two forms. The first is a folder full of .txt files, one for each of the 510 contracts. Each contract is a little different, but the formatting is mostly limited to line breaks and outline structure with all caps for the major headings. The second source of data is the *master_clauses.csv* file which is organized where each of the 510 contracts is represented by one row and each of the 41 label categories comprises two columns: one filled with text extracted from the contracts and the other is a normalized version of the former (e.g., a simple "Yes" or "No" indicating whether such a clause is present for 32 of the categories and dates in a standard format for 3 of the categories). There are a total of 6,701 clauses found among the 510 contracts. The class distribution and other information about the CUAD can be found in Appendix A. The original CUAD paper detailed manual labelling of contracts and clauses by a team of legal professionals; however, the master clauses file seems to be the only labeling publicly available, and the formatting of the extracted portion (e.g., inconsistent N/A symbols and the presence of array brackets and newline characters) indicates machine extraction, but it may have been skimmed over before release.

3.4 Text Preprocessing

The semantic match pipeline leverages the same preprocessing steps as the models presented in the CUAD paper [6]. According to the *utils.py* provided on the CUAD github, text normalization occurs in the following order:

1. Convert the text to lowercase using Python's lower() function
2. Remove punctuation as defined by set(string.punctuation) by replacing it with an empty string

3. Remove articles using a regular expression
4. Replace extra white space with a single space
5. Split the result into tokens

In the spirit of the original paper, the pipeline utilized in this thesis applies the same function to both sets of data as they are ingested into *Pandas DataFrame* structures.

3.4.1 Description of Pandas DataFrame Structures

The *df_contracts* table comprises one row for each of the 510 contracts, with each row containing a normalized version of the contract name, the string for the entire contract, a string for the contract minus all of the clauses found in that contract (used for the 'None' category), and two columns for the tokenized versions of those respective strings.

The *df_clauses* table contains every clause in its raw, normalized, and tokenized forms along with labels for the source documents, the category of clause (column in the original comma-separated values (CSV)), and a numerical representation of the category (ranging from 0 to 40 for the given clause and 41 for the leftover portions of the contract which do not belong to any other category). There were 7,211 total clauses.

An additional DataFrame, *df_01_ans*, contains a simple binary depiction of which clauses are present in each contract; this was useful for identifying category distribution, as shown in Figure 2, and other statistical properties of the dataset.

3.4.2 Splitting into Training and Testing Sets

After the data was ingested and pre-processed, 102 (20%) of the contracts and their 1,474 associated clauses were randomly sampled to split into a testing group;

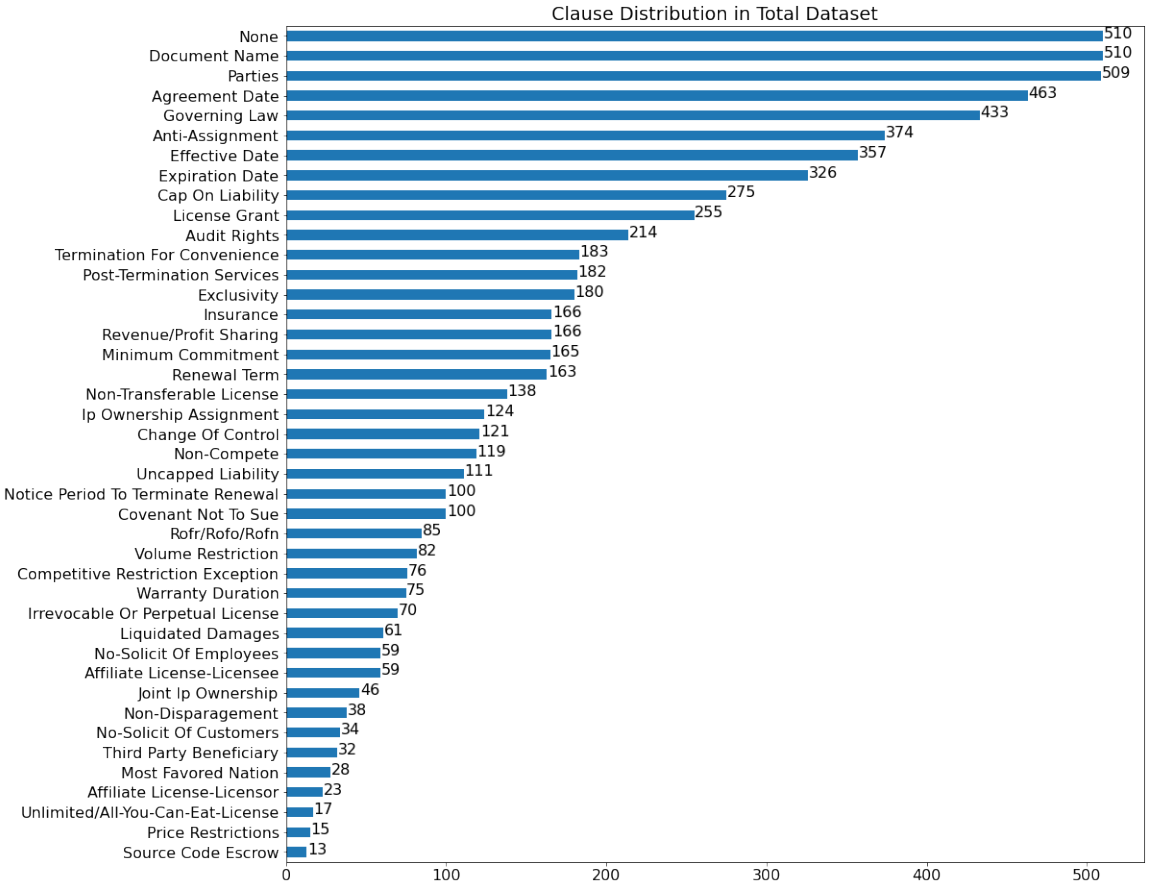

Figure 2: Class distribution in the total dataset, organized by count

the remainder become the training dataset. The class distributions of each set were proportional with respect to the whole dataset. Additionally, this ratio of training/testing matches how the authors of the CUAD paper split the data.

3.5 Feature Extraction

The embedding space chosen for the pipeline was TF-IDF as implemented in sklearn's *TfidfVectorizer* class. The input arrays of string tokens ('Token Normal Clause' column of *df_clauses*) were converted to arrays of 512 floats between 0.0 and 1.0 (i.e., the vectorizer used 512 *max_features*). The numerical 'Class' column of *df_clauses* serves as the y data for training.

3.6 Phase 1: Model Instantiation and Performance on Clauses

The *LogisticRegressionCV* model from sklearn took about twenty minutes to fit with the parameters shown in Table 1, determined by results from prior work that involved a hyperparameter search and comparison to Naïve Bayes, Random Forest Classifier, and FFNN models. This performs a stratified (preserving class distribution) 10-fold cross validation during training. Other logistic regression models shown were also investigated based on parameter examples suggested in online tutorials. Each took around the same or less time to train.

Table 1: Logistic Regression Model Parameters

Model Name	**sklearn class**	**multi_class**	**solver**	**penalty**	**CV**
LogReg CV	LogisticRegressionCV	ovr	liblinear	L2	10
L1 Logistic	LogisticRegression	multinomial	saga	L1	N/A
L2 Logistic (Multinomial)	LogisticRegression	multinomial	saga	L2	N/A
L2 Logistic (OvR)	LogisticRegression	ovr	saga	L2	N/A

3.7 Setup for Whole Contract Evaluation

The next stage of experimentation utilized the training contracts in the *df_contracts* structure. Each contract had been preprocessed into sequences of words but needed to be further segmented for classification. It is standard practice for image and text classification to break streams of data into chunks of equal sizes. Depending on available resources, chunks can be operated on in parallel, and many tasks (e.g., this classification problem) are better suited on smaller samples rather than addressing the whole document at once. This was accomplished using a sliding window generator. The inputs to the generator - length of the window and the number of words to slide between windows - were among the hyperparameters tuned during this stage. Figure 3 demonstrates how the parameters sample windows from a sequence.

These parameters were tuned using the Semantic Match Pipeline with the 408

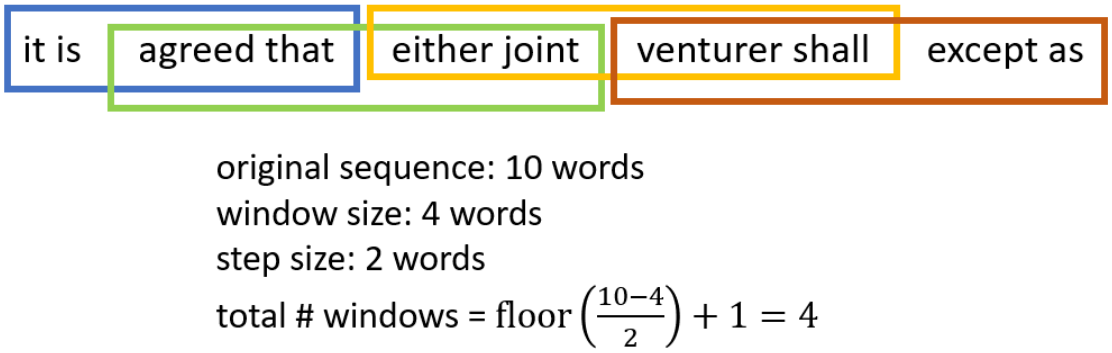

Figure 3: Sliding Window Example

contract text files that correspond to the training clauses as inputs into the sliding window generator. The ranges of hyperparameters are listed in Table 4. In Phase 3, the models were evaluated based on classification of sliding windows built from the 102 contracts in the test set.

3.8 Summary

For Phase 1, the experimental setup started with the *master_clauses.csv* file, processed it, and trained four logistic regression models. In Phase 2, these models were used with the new input of sliding windows from the raw contract text files. Different hyperparameter combinations for the sliding windows were tuned and optimized. Phase 3 in Chapter IV details using the optimizations to produce labels on sliding windows from the test set of contracts.

IV. Results and Analysis

This chapter discusses the results of the models trained per the methodology in the previous chapter and applied to the Contract Understanding Atticus Database (CUAD).

In Test 1, these models were first trained on a portion of the truth data from the CUAD *master_clauses.csv* file and then tested on the remaining 20% of the clauses. Test 2 comprised feeding the models the whole contracts as text files split into sliding windows of about 130 words. Sliding window models from two different hyperparameter searches achieved similar results to each other (and to the transformer models from the CUAD paper) but much lower than utilizing the clauses.

4.1 Precision, Recall, and Accuracy Scoring

The scoring metrics for classification models usually optimize the trade-off between recall (5), which describes the percentage of a certain class that is correctly labeled, and precision (6), which is the percentage of samples given a class that actually were that class. These metrics work well for single-label data. A simplified accuracy (7) scores how many prediction labels match the truth data. This metric is used in the pipeline for the sliding windows since the truth data for each window is no longer single-label but actually an array of acceptable labels. Since some of the windows had a 100% overlap score for multiple labels, it was not possible to single-out the 'top' of the array as the sole 'truth' for the window. While technically possible, there were no windows that were completely absent of at least one 'truth' label.

$$recall = \frac{true\ positives}{true\ positives + false\ negatives} \tag{5}$$

$$precision = \frac{true\ positives}{true\ positives + false\ positives} \tag{6}$$

$$\texttt{accuracy}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} 1(\hat{y}_i = y_i) \tag{7}$$

4.2 Test 1: Model Performance on Curated CSV

In Phase 1, the models were trained using the inputs from *df_clauses* since its truth data had the highest level of scrutiny and curation. The F1 micro-averaged accuracy of the models using the training and validation (test) clauses as input are shown in Table 2. The "test" set is comprised of those clauses from the contracts in the sequestered test set. For comparison, additional models are included from earlier examinations (mentioned in Section 3.2) that utilized only a subset of classes that excluded dates, proper nouns, and the 'None' category. This earlier examination is also what inspired the exclusive focus on logistic regression models [32][33].

Table 2: Test 1 Model Scores

Model Name	**Classes**	**Train Score**	**Test Score**
LogReg CV	42	78.2	66.4
L1 Logistic	42	90.3	73.5
L2 Logistic (Multinomial)	42	89.1	75.8
L2 Logistic (OvR)	42	87.8	75.4
Prior Model	**Classes**	**Train Score**	**Test Score**
Logistic Regression	32	82.0	73.4
Random Forest Classifier	32	85.8	65.1
Multinomial Naïve Bayes	32	71.8	60.0
Neural Network	32	96.0	64.6

Appendix B contains confusion matrices for these four 42-class models. The most common mistake in the LogReg CV model was predicting class "Renewal Term" regardless of the True Class for a clause. While the other models show some false positives among the neighboring "Expiration Date" and "Notice Period to Terminate Renewal," it is unclear why the cross-validated model so heavily favors "Renewal

Term." Its overall representation in the training clauses is a middling 133 samples (2.32%).

Each of the models were susceptible to predicting related topics (e.g., one type of date instead of another), especially in classes 25-30, which all dealt with licenses. While it is difficult to determine the exact cause that class 25 'License Grant' was favored over the rest in that range, it is likely related to its larger representation within that subset. Class 25 had 203 training samples compared to 451 total in the 25-30 range (45.0%). Notably, none of these models correctly predicted the three samples for Class 29 in the test set, which had the smallest sample size of 14 (3.10%) in the licenses range of the training set.

All models had the same 512-dimension term frequency-inverse document frequency (TF-IDF) embedding layer between the clause input and the regression. The errors indicate that some of these similar classes did not have enough exclusive tokens or a large enough sample size for higher accuracy predictions.

4.3 Test 2: Performance on Sliding Windows of Whole Contracts

After the models were fit according to the training clauses, a sliding window parser split the contract text files into input sequences as described in Section 3.7. The logistic regression models classified each of these windows as one of the 42 classes established earlier. However, the truth data from *df_clauses* did not directly translate to this windowed representation for several reasons. First, the manually curated entries from the *master_clauses.csv* file would sometimes standardize the responses for a given category (e.g., dates in MM/DD/YYYY format) into a form not actually present in the original document or any of its windows. This meant that expecting any window (or even the entire document) to contain 100% of the "truth clause" was unfeasible. Second, the categories had different average lengths, so small win-

dow sizes would contain only portions of larger clauses while larger windows might contain multiple different clauses. Third, some of the answers comprised multiple, non-consecutive phrases or sentences from the contract. The sizes of the gaps between answer segments were not easily predictable, especially across different answer categories.

Taking into account these challenges, "correct answers" for the windowed representation of contracts were formulated using a bag-of-words approach with the *df_clauses* truth data, shown in Algorithm 1. The words in each truth clause were compared to the words in each window; the size of the intersection of these two lists was divided by the size of whichever list was smaller to determine an overlap percentage. If this overlap percentage exceeded the hyperparameter *window_accuracy*, then that class category was added to that window's list of "correct answers" (*y_Actual*) for later comparison to model outputs (*y_Predicted*). It was often the case that windows would have multiple categories labeled as correct. In particular the 'None' category #41 would almost always exceed the *window_accuracy* threshold given the sheer size and range of its vocabulary in each contract. Unfortunately, this did not solve the problem that some *df_clauses* samples had been standardized into forms found nowhere in the actual document. Additionally, the disproportionate concentration of clauses in the beginning of contracts meant that earlier windows would invariably contain multiple clauses yet receive only one prediction. For example, while dates and renewal terms represent nearly 15% of the clauses, they were only predicted as labels for about 5% of the windows.

4.3.1 Hyperparameter Search with Bayesian Optimization

Using the *hyperopt* package, a Bayesian optimization routine was used to find the optimal hyperparameters among different options for logistic regression model,

Algorithm 1 Creating New Ground Truth Labels

```
function OVERLAP_SCORE(window_words, clause_words)
    words_both = intersection(window_words, clause_words)
    return |words_both| / min(|window_words|,|clause_words|)
end function
  ▷ Apply overlap_score to each window and clause combination for each contract
new_truths = []
for contract in contract_list do
    for window in contract do
        for clause in clause_list[contract] do
            if overlap_score(window, clause) > threshold then
                new_truths.append(clause.label())
            end if
        end for
    end for
end for
```

window size, step size, and window accuracy. The objective function (minimized through successive trials) is listed in (8). A true positive occurred when the model predicted one of the classes from the *y_Actual* array for that window. A false positive was recorded for any other class prediction. This precision score was the basis for model scoring. However, an alternative scoring metric known as the Hamming score, shown in Equation (9), would penalize the additional entries in *y_Actual*, yielding the results shown in Table 3. A different search would need to be conducted to optimize this alternative scoring option.

$$f(x) = 1 - precision \tag{8}$$

$$Hamming\ score = \frac{1}{n}\sum_{i=1}^{n}\frac{|y_i \bigcap \hat{y}_i|}{|y_i \bigcup \hat{y}_i|} \tag{9}$$

During the optimization, the experimental domains of the hyperparameters were as follows:

- The *window_size* (in words) was chosen among the integers [1, 133), representing

Table 3: Hamming scores for sliding window Semantic Match Pipeline

Model	Hamming Train %	Test %
L2 multinomial, 130 word window/step	27.1	27.2
L2 OVR, 132 word window/step	25.6	25.9

up to the length of seven average sentences

- The *step_size* was determined with floor division (i.e., rounding down to a whole number) of the window size and an integer from the range [1, 5), so the windows would slide proportional to their size;
- The *window_accuracy* threshold was chosen from a uniform distribution in the interval [0.5, 0.99].

Lower thresholds and larger windows yielded more entries in *y_Actual* which the model would almost always prefer. Two sets of 1,000 trials were run. The first set used the Tree Parzen Estimator optimization algorithm and the other used a random search algorithm, each using the default *hyperopt* configurations. Each trial took between 30 and 600 seconds to run; incremental progress was saved in a *hyperopt Trials* object. The hyperparameter combination for the best-scoring trial from each search algorithm is shown in Table 4.

Table 4: Hyperparameter Bayesian Optimization Search Results

Search Algorithm	Classifier	Window Size	Step Size	Window Threshold	Training Score
Tree Parzen Estimator	L2 multinomial	130 words	window/1 = 130	0.503733	0.45081
Random Search	L2 OVR	132 words	window/1 = 132	0.513973	0.42914

Both search algorithms selected large window sizes and step sizes with window thresholds on the lower end of the range. This meant that the model performed best when the *y_Actual* table contained more entries per window. The larger vocabulary in the window added to the numerator of the *window_accuracy* calculation but rarely

affected the denominator (the minimum of length of truth clause or window). Having a step size increment of an entire window produced fewer windows overall. This helped reduce fitting time for the models.

4.3.2 Model Performance on Windowed Set

The two hyperparameter searches yielded the models shown in Table 4. Additional tables and pie charts displaying the specific counts for *y_Actual* and *y_Predicted* for each model and dataset are in Appendix C. As expected, the 'None' clause dominated the *overlap_score* threshold and appeared in almost 100% of the *y_Actual* lists; however, the models only predicted 'None' about 22-25% of the time. Still, this dwarfs the next most-predicted class of 'Cap on Liability' by a factor of about 4.

4.3.2.1 Hyperparameters from Tree Parzen Estimator Algorithm

The first hyperparameter search determined the L2 multinomial classifier with a window size and step size of 130 words and a window threshold of 0.504 to be optimal. Out of 22,262 total windows from the training contracts, it correctly classified 10,036 yielding an accuracy of 45.08%. For the test set, the model classified 2,526 of 5,606 correctly for 45.06% accuracy.

4.3.2.2 Hyperparameters from Random Search Algorithm

The second hyperparameter search determined the L2 one-vs-rest classifier with a window size and step size of 132 words and a window threshold of 0.514 to be optimal. Out of 21,930 total windows from the training contracts, it correctly classified 9,411 yielding an accuracy of 42.91%. For the test set, the model classified 2,379 of 5,518 correctly for 43.11% accuracy.

4.4 Comparison to Original CUAD Models

The main metric used in the CUAD paper is the area under the precision recall curve (AUPR). Table 5 below shows the scores for the top two CUAD models presented. Table 6 displays the results of the sliding window logistic regression models. Those curves are not easily available for these sliding window models due to the nature of the *y_Actual* having multiple acceptable entries per window but the model is only configured to predict a single class per window. However, the accuracy scores in the low 40s are on par with the precision at 80% recall for their BERT models. The CUAD models used a Jaccard similarity coefficient to determine if the windows matched ground truth; there, the denominator is the magnitude of the union of the sets being compared (a window and a truth clause) while this thesis utilized the magnitude of the smaller set instead. Still, the threshold near 0.5 was optimal in both papers [6].

Table 5: Results of NLP Transformer Models on CUAD

CUAD Model	**AUPR**	**Precision @ 80% Recall**
RoBERTa-large	48.2	38.1
DeBERTa-xlarge	47.8	44

Table 6: Training and Test Scores for Sliding Window Models

Model	**Training Score %**	**Test Score %**
L2 multinomial, 130 word window/step	45.1	45.1
L2 OVR, 132 word window/step	42.9	43.1

4.5 Summary

This research sought to find a logistic regression model that could quickly train on the CUAD, a dataset full of technical legal texts, and perform classification tasks

without lagging too far behind much larger and slower transformer models. The logistic regression models presented were trained and tuned within thirty minutes and reached accuracies much higher than anticipated, though differences in methodology obscure direct comparisons with the transformers.

V. Conclusions

5.1 Summary and Lessons Learned

One of the hardest challenges to overcome was the lack of truth data to base scoring metrics on. Upon manual inspection, even the *master_clauses.csv* from the CUAD contained minor errors and formatting inconsistencies. Finding enough reliable data for training is paramount for building trust in the model outputs. Machine learning models really benefit from having more input. While 7,000 entries might sound like an overwhelming number of data entries to parse by hand, it becomes much less impressive once divided among 42 classes with widely varying distribution and further split into training and testing data. The fact that many categories were similar to each other further complicated the task.

Using models and packages that were already available with ample documentation proved critical to setting up and running the experiments necessary for this thesis. Some attempts were made early on at implementing alternate models such as neural networks, but they provided worse results while simultaneously being more difficult to comprehend how it worked.

5.2 Future Work

5.2.1 Improving the Models

If there were a reliable way to reduce the *y_Actual* list to a single class for each window, the scoring and results would be easier and fairer to compare among other similar models and techniques. One potential method would be to limit the list to the highest percentage of overlap instead of all classes which exceed the overlap score. However, with the current hyperparameters, multiple classes yield 100% overlap with no fair way to pick a single winner. An alternate method from the overlap score could

involve comparing the TF-IDF vector of the window to the average of TF-IDF vectors for each class in the training set. The Hamming score described in Equation (9) is another alternative score that could be optimized.

A more intensive approach would be to take the time to tune a separate classifier for each class or for classes with similar lengths. There is a wide variance in the average length of clauses among the different categories; for example, relevant dates are often one to three words while a non-compete clause can be an entire paragraph. The CUAD code available on their github contained a function that appeared to match clauses to their indices within each contract; this would have been interesting to utilize if attempts to integrate it had been successful.

With a little modification, the models could be expanded into multi-label classifiers, so the *y_Predict* arrays could have more than one entry. It would then be possible to score based on how many labels match for each window. This would also translate better for the actual intended purpose of this thesis: resumé classification.

The best (but also least likely) long-term solution is reforming the legal industry standards so that contracts do not contain useless walls of text. A modular approach where each clause is clearly labeled and follows a standard phrasing to begin with would help both people and machines understand the contract more easily.

5.2.2 Transitioning to Resumé Dataset

The original purpose of this thesis was to support efforts to assist clerical staff with reviewing resumés. Legal contracts were used as an analogue due to the unavailability of truth data for technical resumés. In theory, a model's ability to classify windows of a contract is correlated to its ability to interpret the 'Specialized Experience' free text field in job applications. This field is where applicants demonstrate that their prior education and experiences meets or exceeds the requirements listed in the job

posting.

The tasks are not exactly identical, but some of the differences may contribute to higher accuracy in the resumé domain once truth data is acquired. For example, the 'Specialized Experience' free text field will be much shorter than the average legal contract. Additionally, if each job requirement is considered a class, there will likely be far fewer than 42 classes to consider for any given position. This means that models can be trained and tuned much more quickly, depending on certain design choices. Some particularly impactful decisions include whether to use online or offline learning and whether to use a separate classification model for each requirement, a separate model for each job posting, or one mega-model that must be updated with each new requirement or posting. Separate models for each requirement would be an efficient way to share among different job postings and also incorporate multi-label output; that is, a single sentence (or window) may fulfill more than one requirement of the job posting. The individual models for each requirement could scan the paragraph to determine which portion, if any, might fulfill their requirement.

There are three major downsides immediately apparent to this sort of solution compared to alternatives such as holding the hiring staff to higher standards. The first is that any software tool that manages these models will need to be periodically updated to incorporate newer techniques and remain compatible with whatever suite of anti-user and anti-capability restrictions put in place on government computers. Clerical staff will also require training on using the tool, likely on a recurring basis. Sustainment costs for the tool will undoubtedly eclipse the amount of money it would take to hire more qualified and effective recruiters.

Secondly, users of the tool will gradually become less proficient in using their own discretion to determine whether a candidate is qualified or not. While the stated intent/extent of the hypothetical tool would be merely assisting the human by identi-

fying possible text matches, that is not the actual expected outcome. If users do not trust the tool enough to reduce their personal critical thinking effort, then it was a waste of time and resources. If users do trust the tool enough to offload mental work, then their capacity for critical thinking will diminish as it falls out of practice. This can be alleviated if the deficit in mental work is instead applied elsewhere, but care must be taken to avoid diverting them so much that they lose their focus on hiring the most qualified candidates for the job.

Finally, and perhaps most importantly, is that the hypothetical tool might enable underqualified candidates to reach the next level of hiring when they otherwise would have been eliminated. While each job posting has its own list of requirements, there is one hidden yet critical trait common among all positions: effective communication and professional writing skills. If a candidate is unable or unwilling to tailor their resumé for the intended audience of non-technical HR professionals, they should not be allowed to move forward in the selection process. There is already an overwhelming lack of communication competence present in the government workforce that rivals the need for technical competence that the tool seeks to address. If this type of hiring remains centralized rather than distributed to frontline managers or supervisors where it belongs, recruiters should not be expected to be up to date on the latest jargon; applicants should take responsibility for making their profile understandable and attractive to prospective employers. There is no net benefit from gaining new hires with the utmost technical skills if they are unable to work well within the bureaucracy and boundaries expected of them.

Appendix A. Additional Dataset Information

1.1 CUAD GitHub

The GitHub repository for the Contract Understanding Atticus Database (CUAD) can be found online at `https://github.com/TheAtticusProject/cuad`. The authors also link their website and the published paper that accompanies the dataset. The *master_clauses.csv* file was located within the 'CUAD_v1' download through the website.

1.2 Overlaid Histogram

Figure 4 shows the class distribution of clauses among the total and training datasets. The test distribution is implied in the difference.

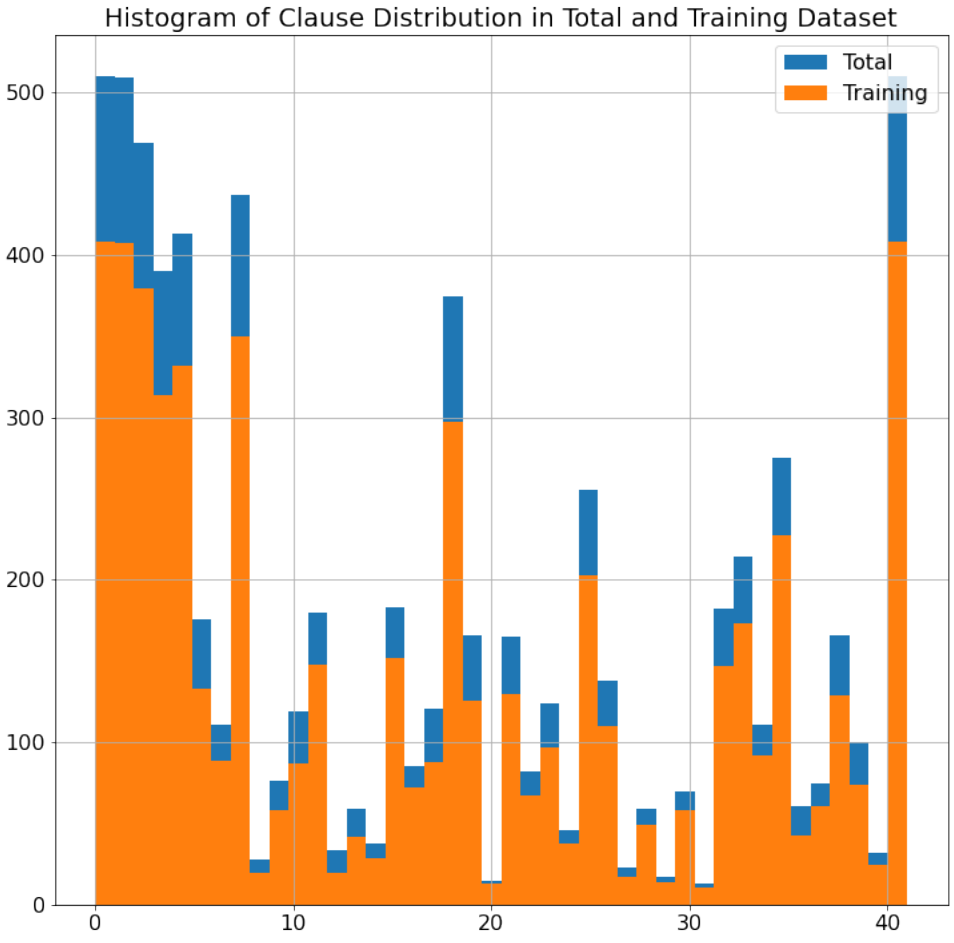

Figure 4: Histogram of Clauses

1.3 Class Names and Numbers

Table 7 displays the pairing of class names and label numbers as referenced throughout the thesis. The numbers were used as the classification label by the models. The order of classes comes from the order of their respective columns in the *master_clauses.csv* file, withh 'None' being the final class as it was added for manually.

Table 7: Matching class names to numbers

Class Name	#	Class Name	#	Class Name	#
Document Name	0	**Non-Disparagement**	14	**Affiliate License-Licensee**	28
Parties	1	**Termination For Convenience**	15	**Unlimited/All-You-Can-Eat-License**	29
Agreement Date	2	**Rofr/Rofo/Rofn**	16	**Irrevocable Or Perpetual License**	30
Effective Date	3	**Change Of Control**	17	**Source Code Escrow**	31
Expiration Date	4	**Anti-Assignment**	18	**Post-Termination Services**	32
Renewal Term	5	**Revenue/Profit Sharing**	19	**Audit Rights**	33
Notice Period To Terminate Renewal	6	**Price Restrictions**	20	**Uncapped Liability**	34
Governing Law	7	**Minimum Commitment**	21	**Cap On Liability**	35
Most Favored Nation	8	**Volume Restriction**	22	**Liquidated Damages**	36
Competitive Restriction Exception	9	**Ip Ownership Assignment**	23	**Warranty Duration**	37
Non-Compete	10	**Joint Ip Ownership**	24	**Insurance**	38
Exclusivity	11	**License Grant**	25	**Covenant Not To Sue**	39
No-Solicit Of Customers	12	**Non-Transferable License**	26	**Third Party Beneficiary**	40
No-Solicit Of Employees	13	**Affiliate License-Licensor**	27	**None**	41

Appendix B. Confusion Matrices

For each of these confusion matrices, the true class label is listed on the left side while the model's predicted class is listed across the top. The diagonal line from top left to bottom right represents true positives. These all refer to the dataset from *master_clauses.csv*, not the sliding windows.

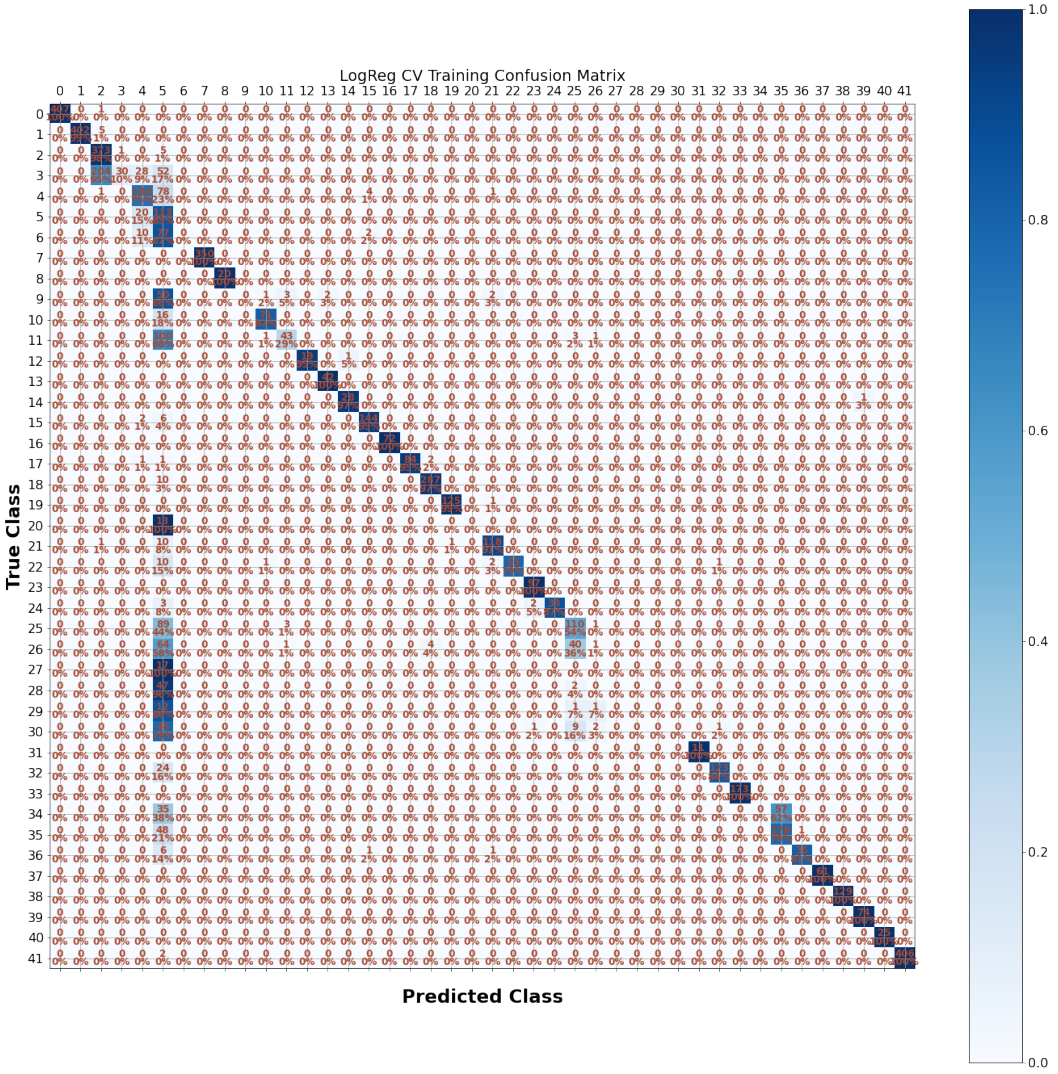

Figure 5: LogReg CV Training Confusion Matrix

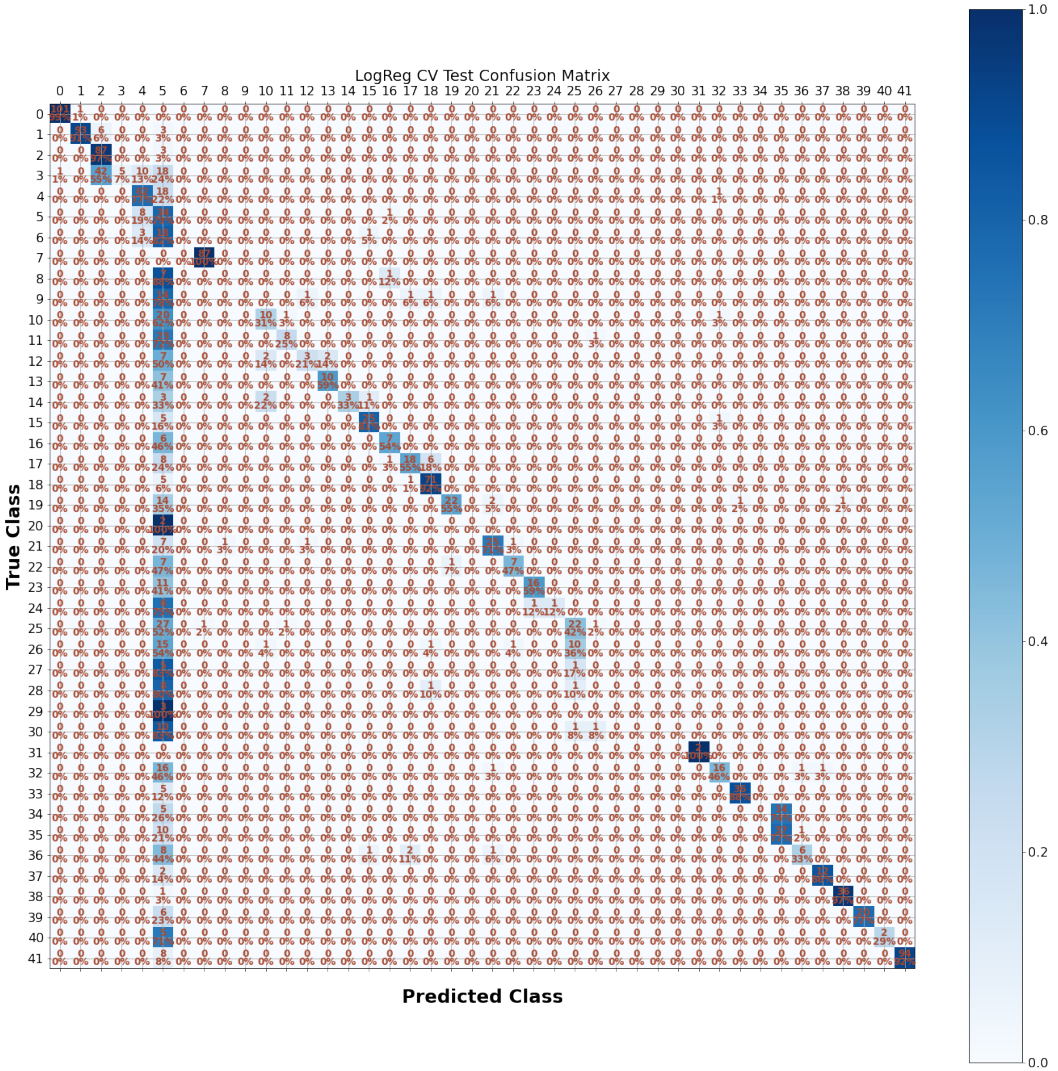

Figure 6: LogReg CV Test Confusion Matrix

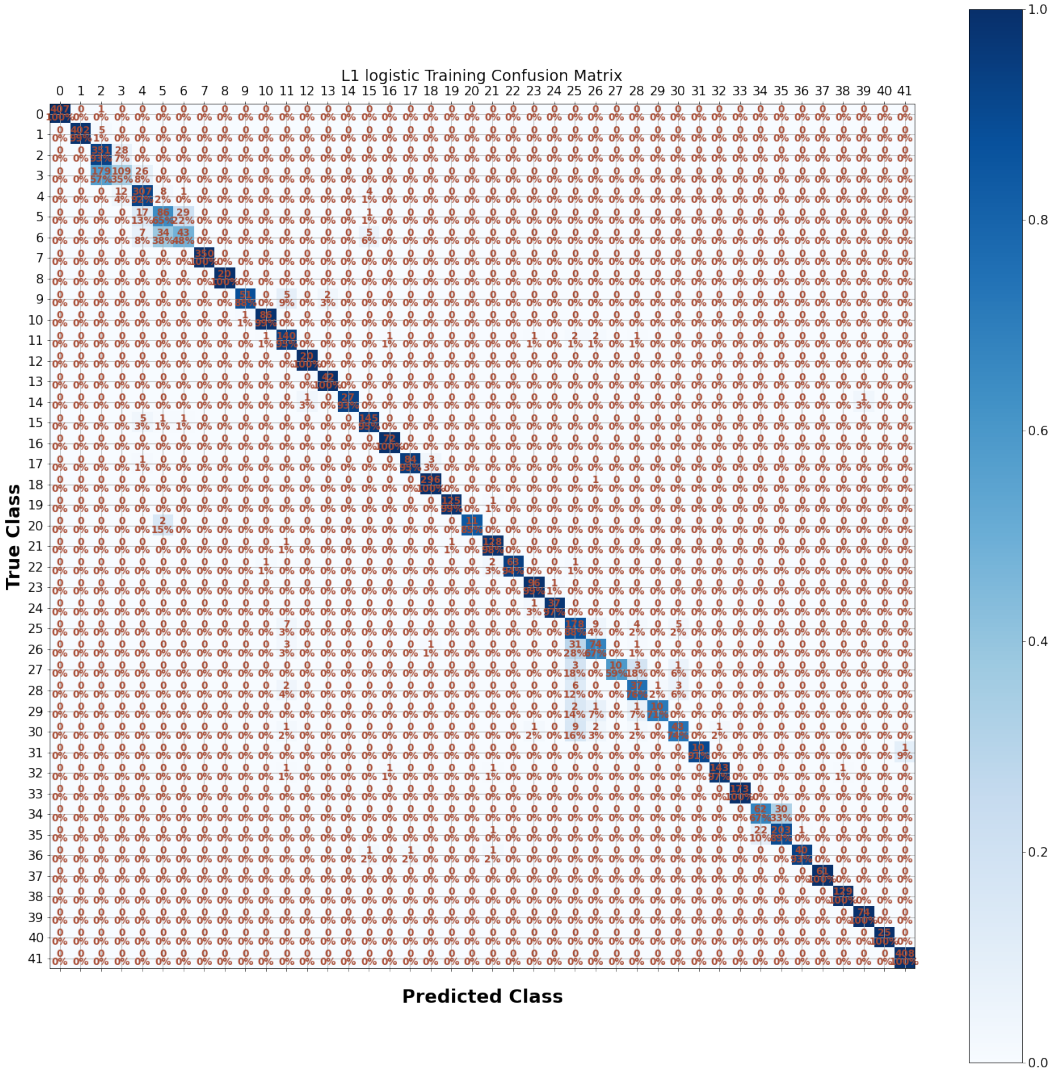

Figure 7: L1 Training Confusion Matrix

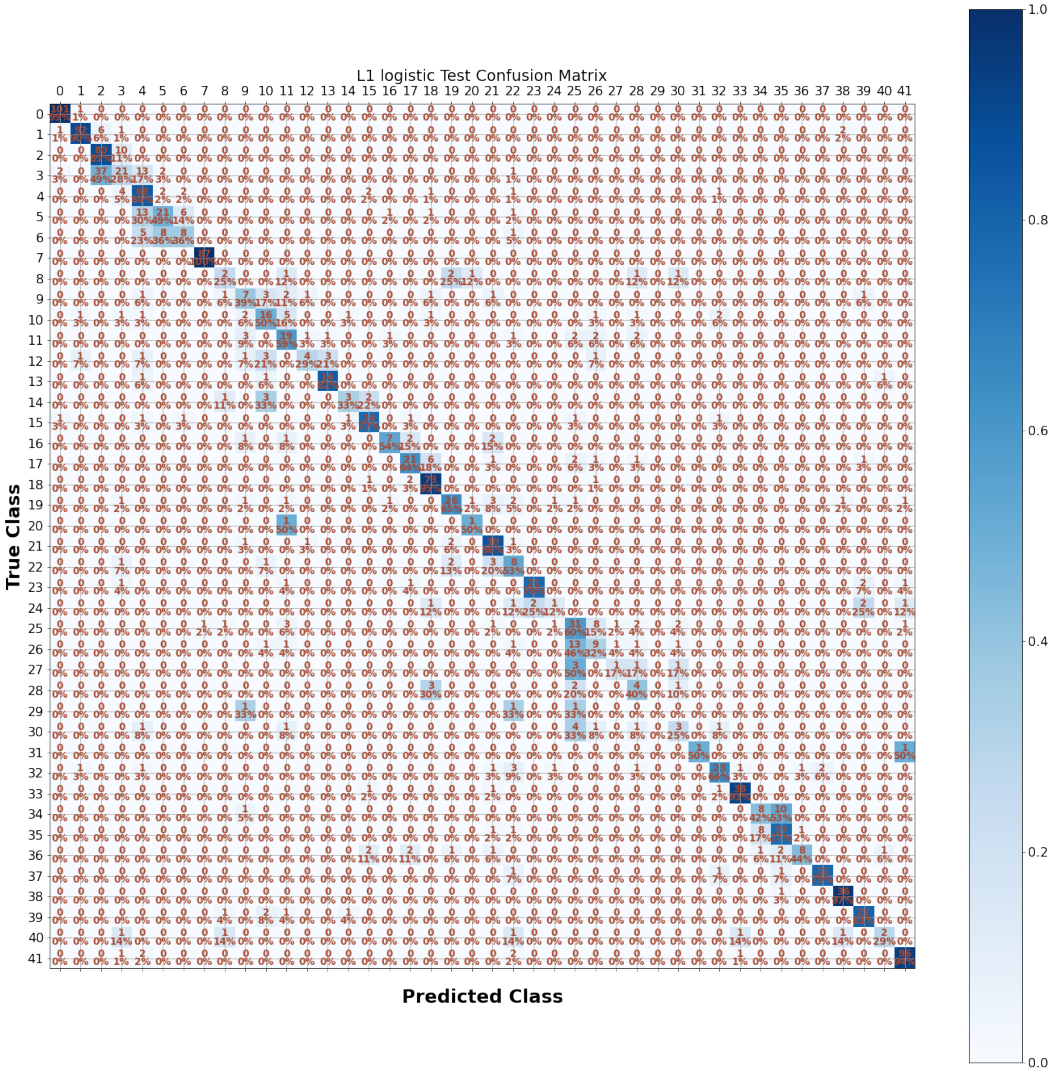

Figure 8: L1 Test Confusion Matrix

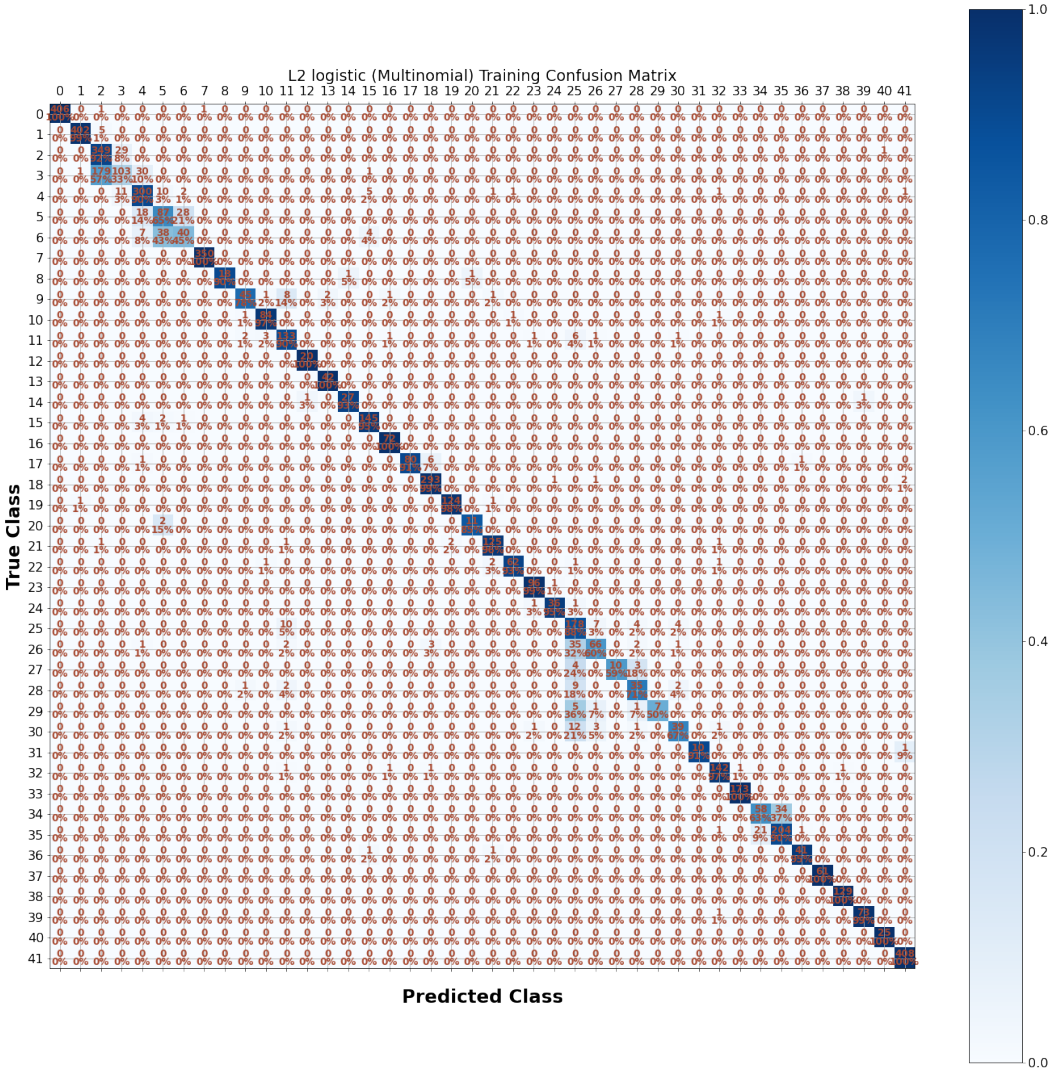

Figure 9: L2 Multinomial Training Confusion Matrix

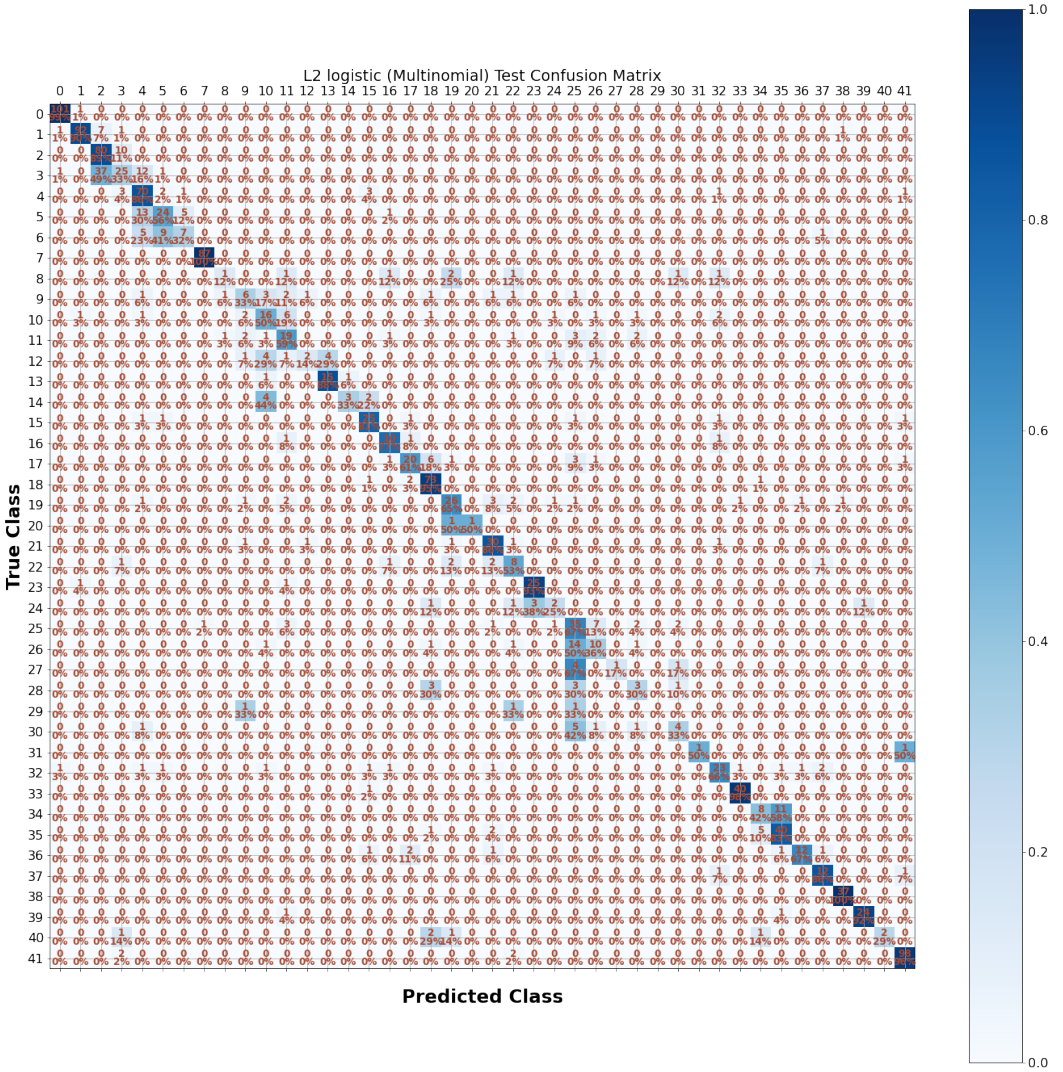

Figure 10: L2 Multinomial Test Confusion Matrix

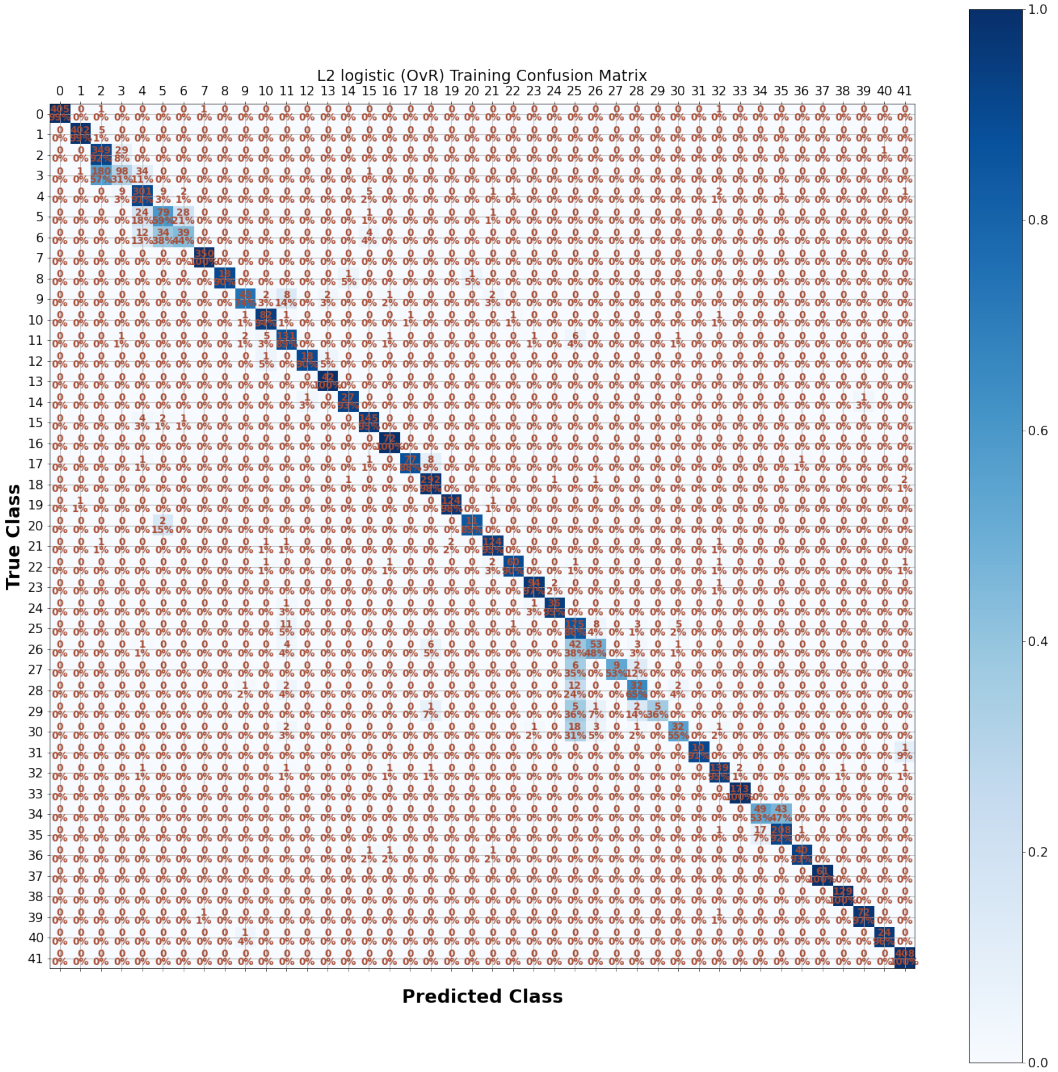

Figure 11: L2 OvR Training Confusion Matrix

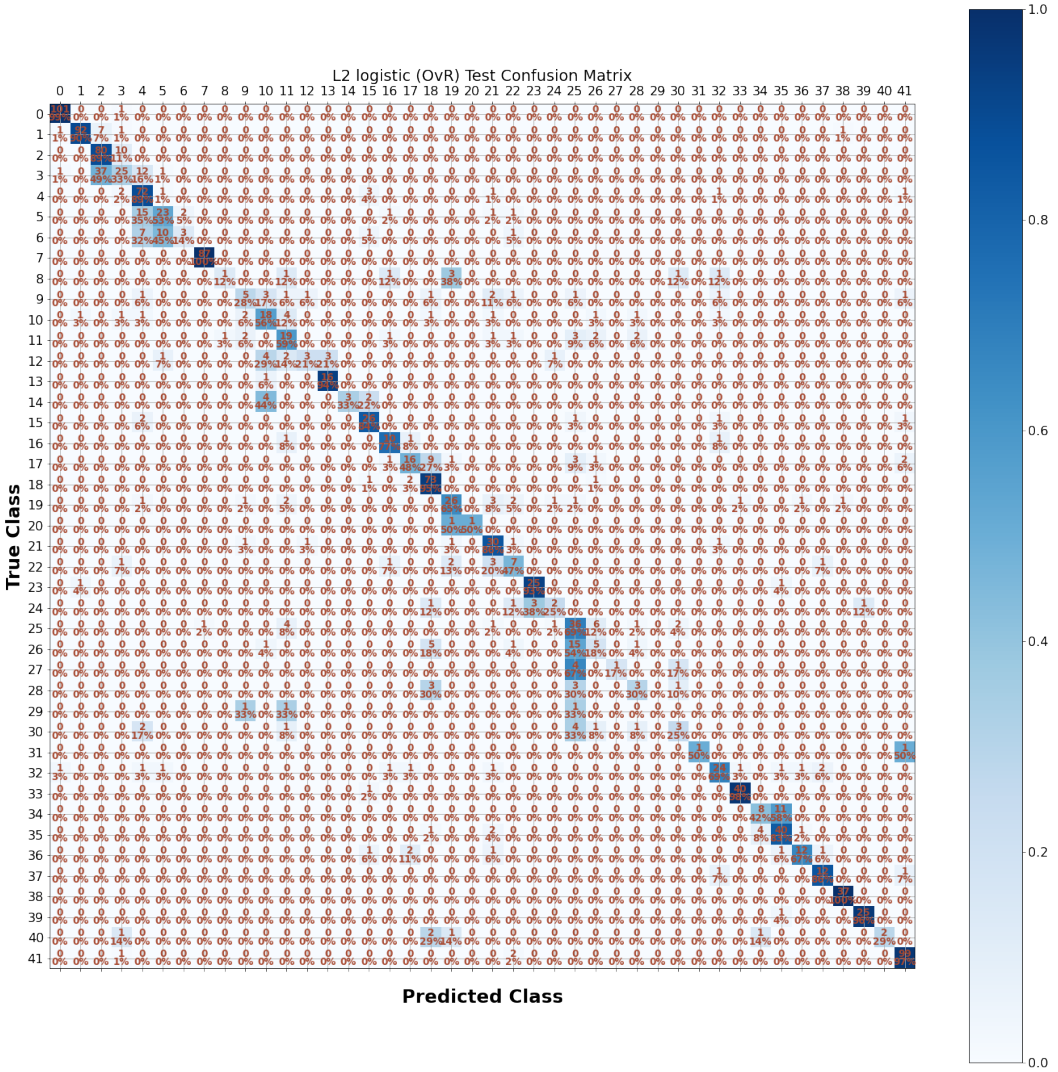

Figure 12: L2 OvR Test Confusion Matrix

Appendix C. Sliding Window Counts

The following tables display the class counts for the training and test datasets for the two sliding windows models. The 'Clause' columns represent the number of times each clause was present in *master_clauses.csv*. The 'Actual' counts show how many of the sliding windows had an overlap score that exceeded the threshold. Since windows could have multiple 'Actual' entries, the sum of this column is not equal to the total number of sliding windows; however, the opposite is true for the 'Predicted' counts.

Table 8: TPE Model Training Counts

Class Name	Class	Clause	Clause %	Actual	AC %	Predicted	PC %
Document Name	0	408	7.11	4817	21.64	101	0.45
Parties	1	407	7.09	2724	12.24	348	1.56
Agreement Date	2	379	6.61	952	4.28	120	0.54
Effective Date	3	314	5.47	927	4.16	522	2.34
Expiration Date	4	332	5.79	1043	4.69	436	1.96
Renewal Term	5	133	2.32	260	1.17	141	0.63
Notice Period To Terminate Renewal	6	89	1.55	204	0.92	24	0.11
Governing Law	7	350	6.10	1812	8.14	595	2.67
Most Favored Nation	8	20	0.35	58	0.26	41	0.18
Competitive Restriction Exception	9	58	1.01	188	0.84	360	1.62
Non-Compete	10	87	1.52	420	1.89	526	2.36
Exclusivity	11	148	2.58	690	3.10	895	4.02
No-Solicit Of Customers	12	20	0.35	72	0.32	25	0.11
No-Solicit Of Employees	13	42	0.73	122	0.55	97	0.44
Non-Disparagement	14	29	0.51	175	0.79	258	1.16
Termination For Convenience	15	152	2.65	823	3.70	275	1.24
Rofr/Rofo/Rofn	16	72	1.26	528	2.37	589	2.65
Change Of Control	17	88	1.53	268	1.20	599	2.69
Anti-Assignment	18	297	5.18	1411	6.34	632	2.84
Revenue/Profit Sharing	19	126	2.20	433	1.95	1109	4.98
Price Restrictions	20	13	0.23	37	0.17	23	0.10
Minimum Commitment	21	130	2.27	517	2.32	647	2.91
Volume Restriction	22	67	1.17	179	0.80	1045	4.69
Ip Ownership Assignment	23	97	1.69	372	1.67	662	2.97
Joint Ip Ownership	24	38	0.66	254	1.14	192	0.86
License Grant	25	203	3.54	1088	4.89	598	2.69
Non-Transferable License	26	110	1.92	524	2.35	160	0.72
Affiliate License-Licensor	27	17	0.30	140	0.63	40	0.18
Affiliate License-Licensee	28	49	0.85	261	1.17	156	0.70
Unlimited/All-You-Can-Eat-License	29	14	0.24	35	0.16	4	0.02
Irrevocable Or Perpetual License	30	58	1.01	261	1.17	100	0.45
Source Code Escrow	31	11	0.19	73	0.33	53	0.24
Post-Termination Services	32	147	2.56	1177	5.29	1142	5.13
Audit Rights	33	173	3.02	714	3.21	976	4.38
Uncapped Liability	34	92	1.60	177	0.80	136	0.61
Cap On Liability	35	227	3.96	793	3.56	1331	5.98
Liquidated Damages	36	43	0.75	106	0.48	235	1.06
Warranty Duration	37	61	1.06	176	0.79	447	2.01
Insurance	38	129	2.25	527	2.37	334	1.50
Covenant Not To Sue	39	74	1.29	231	1.04	702	3.15
Third Party Beneficiary	40	25	0.44	246	1.11	101	0.45
None	41	408	7.11	22262	100.00	5485	24.64

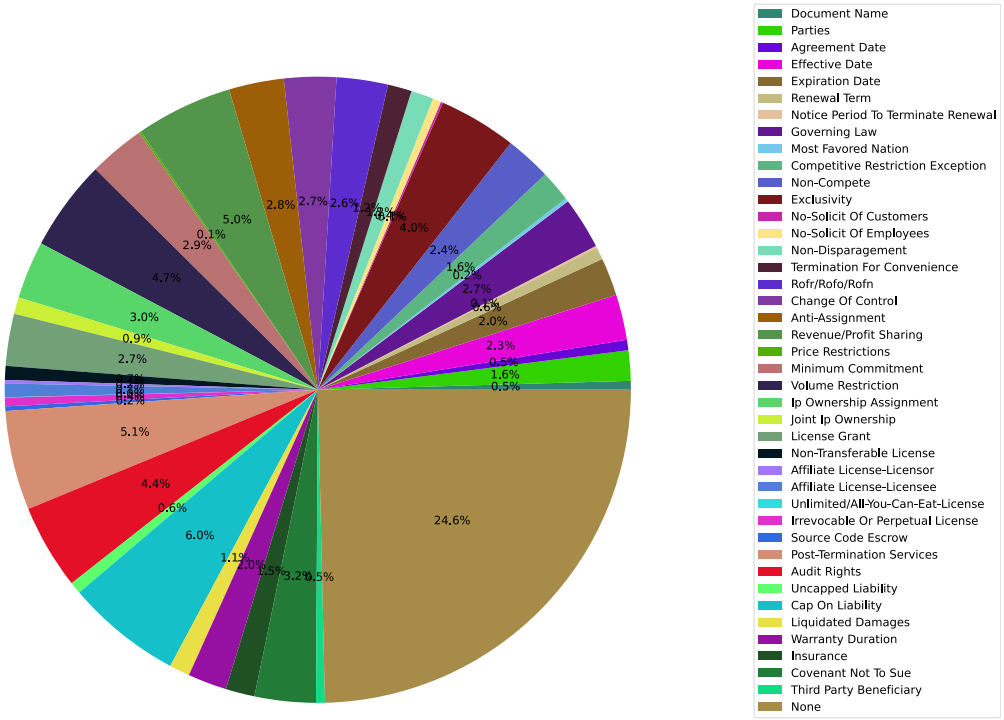

Figure 13: Pie Chart of Predicted Counts in Table 8

Table 9: TPE Model Test Counts

Class Name	Class	Clause	Clause %	Actual	AC %	Predicted	PC %
Document Name	0	102	6.92	1124	20.05	19	0.34
Parties	1	102	6.92	719	12.83	84	1.50
Agreement Date	2	90	6.11	241	4.30	14	0.25
Effective Date	3	76	5.16	316	5.64	111	1.98
Expiration Date	4	81	5.50	229	4.08	119	2.12
Renewal Term	5	43	2.92	86	1.53	51	0.91
Notice Period To Terminate Renewal	6	22	1.49	42	0.75	2	0.04
Governing Law	7	87	5.90	407	7.26	163	2.91
Most Favored Nation	8	8	0.54	13	0.23	11	0.20
Competitive Restriction Exception	9	18	1.22	53	0.95	116	2.07
Non-Compete	10	32	2.17	127	2.27	149	2.66
Exclusivity	11	32	2.17	117	2.09	196	3.50
No-Solicit Of Customers	12	14	0.95	46	0.82	3	0.05
No-Solicit Of Employees	13	17	1.15	45	0.80	28	0.50
Non-Disparagement	14	9	0.61	49	0.87	86	1.53
Termination For Convenience	15	31	2.10	131	2.34	49	0.87
Rofr/Rofo/Rofn	16	13	0.88	158	2.82	141	2.52
Change Of Control	17	33	2.24	201	3.59	155	2.76
Anti-Assignment	18	77	5.22	358	6.39	170	3.03
Revenue/Profit Sharing	19	40	2.71	146	2.60	229	4.08
Price Restrictions	20	2	0.14	4	0.07	3	0.05
Minimum Commitment	21	35	2.37	228	4.07	150	2.68
Volume Restriction	22	15	1.02	42	0.75	221	3.94
Ip Ownership Assignment	23	27	1.83	247	4.41	197	3.51
Joint Ip Ownership	24	8	0.54	43	0.77	48	0.86
License Grant	25	52	3.53	404	7.21	197	3.51
Non-Transferable License	26	28	1.90	189	3.37	30	0.54
Affiliate License-Licensor	27	6	0.41	99	1.77	15	0.27
Affiliate License-Licensee	28	10	0.68	93	1.66	44	0.78
Unlimited/All-You-Can-Eat-License	29	3	0.20	5	0.09	2	0.04
Irrevocable Or Perpetual License	30	12	0.81	62	1.11	22	0.39
Source Code Escrow	31	2	0.14	38	0.68	12	0.21
Post-Termination Services	32	35	2.37	314	5.60	273	4.87
Audit Rights	33	41	2.78	121	2.16	236	4.21
Uncapped Liability	34	19	1.29	41	0.73	42	0.75
Cap On Liability	35	48	3.26	131	2.34	325	5.80
Liquidated Damages	36	18	1.22	61	1.09	84	1.50
Warranty Duration	37	14	0.95	32	0.57	70	1.25
Insurance	38	37	2.51	214	3.82	91	1.62
Covenant Not To Sue	39	26	1.76	109	1.94	215	3.84
Third Party Beneficiary	40	7	0.47	31	0.55	15	0.27
None	41	102	6.92	5605	99.98	1418	25.29

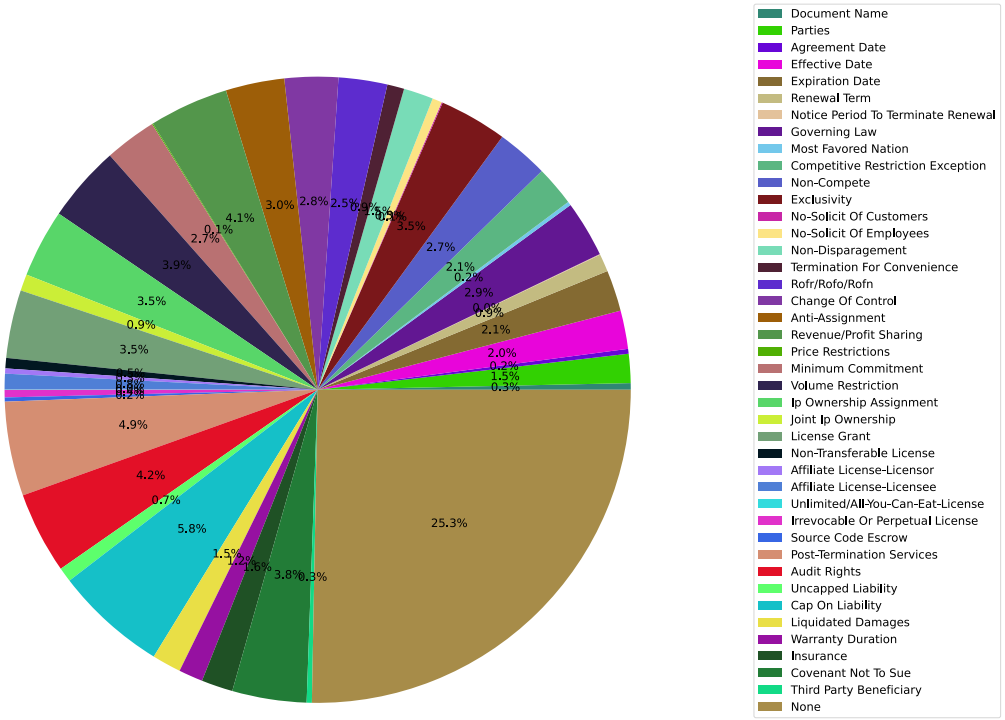

Figure 14: Pie Chart of Predicted Counts in Table 9

Table 10: Rand Model Training Counts

Class Name	Class	Clause	Clause %	Actual	AC %	Predicted	PC %
Document Name	0	408	7.11	4790	21.84	80	0.36
Parties	1	407	7.09	2712	12.37	275	1.25
Agreement Date	2	379	6.61	933	4.25	130	0.59
Effective Date	3	314	5.47	929	4.24	462	2.11
Expiration Date	4	332	5.79	1024	4.67	413	1.88
Renewal Term	5	133	2.32	255	1.16	123	0.56
Notice Period To Terminate Renewal	6	89	1.55	216	0.98	21	0.10
Governing Law	7	350	6.10	1823	8.31	496	2.26
Most Favored Nation	8	20	0.35	50	0.23	30	0.14
Competitive Restriction Exception	9	58	1.01	162	0.74	498	2.27
Non-Compete	10	87	1.52	402	1.83	605	2.76
Exclusivity	11	148	2.58	641	2.92	1034	4.72
No-Solicit Of Customers	12	20	0.35	72	0.33	33	0.15
No-Solicit Of Employees	13	42	0.73	104	0.47	93	0.42
Non-Disparagement	14	29	0.51	174	0.79	310	1.41
Termination For Convenience	15	152	2.65	808	3.68	227	1.04
Rofr/Rofo/Rofn	16	72	1.26	467	2.13	596	2.72
Change Of Control	17	88	1.53	266	1.21	590	2.69
Anti-Assignment	18	297	5.18	1355	6.18	679	3.10
Revenue/Profit Sharing	19	126	2.20	407	1.86	1115	5.08
Price Restrictions	20	13	0.23	35	0.16	16	0.07
Minimum Commitment	21	130	2.27	494	2.25	644	2.94
Volume Restriction	22	67	1.17	170	0.78	1101	5.02
Ip Ownership Assignment	23	97	1.69	372	1.70	662	3.02
Joint Ip Ownership	24	38	0.66	250	1.14	212	0.97
License Grant	25	203	3.54	979	4.46	538	2.45
Non-Transferable License	26	110	1.92	515	2.35	177	0.81
Affiliate License-Licensor	27	17	0.30	132	0.60	56	0.26
Affiliate License-Licensee	28	49	0.85	232	1.06	168	0.77
Unlimited/All-You-Can-Eat-License	29	14	0.24	34	0.16	10	0.05
Irrevocable Or Perpetual License	30	58	1.01	235	1.07	101	0.46
Source Code Escrow	31	11	0.19	66	0.30	52	0.24
Post-Termination Services	32	147	2.56	1088	4.96	1273	5.80
Audit Rights	33	173	3.02	655	2.99	900	4.10
Uncapped Liability	34	92	1.60	172	0.78	139	0.63
Cap On Liability	35	227	3.96	743	3.39	1275	5.81
Liquidated Damages	36	43	0.75	119	0.54	246	1.12
Warranty Duration	37	61	1.06	162	0.74	440	2.01
Insurance	38	129	2.25	472	2.15	304	1.39
Covenant Not To Sue	39	74	1.29	221	1.01	719	3.28
Third Party Beneficiary	40	25	0.44	269	1.23	84	0.38
None	41	408	7.11	21930	100	5003	22.81

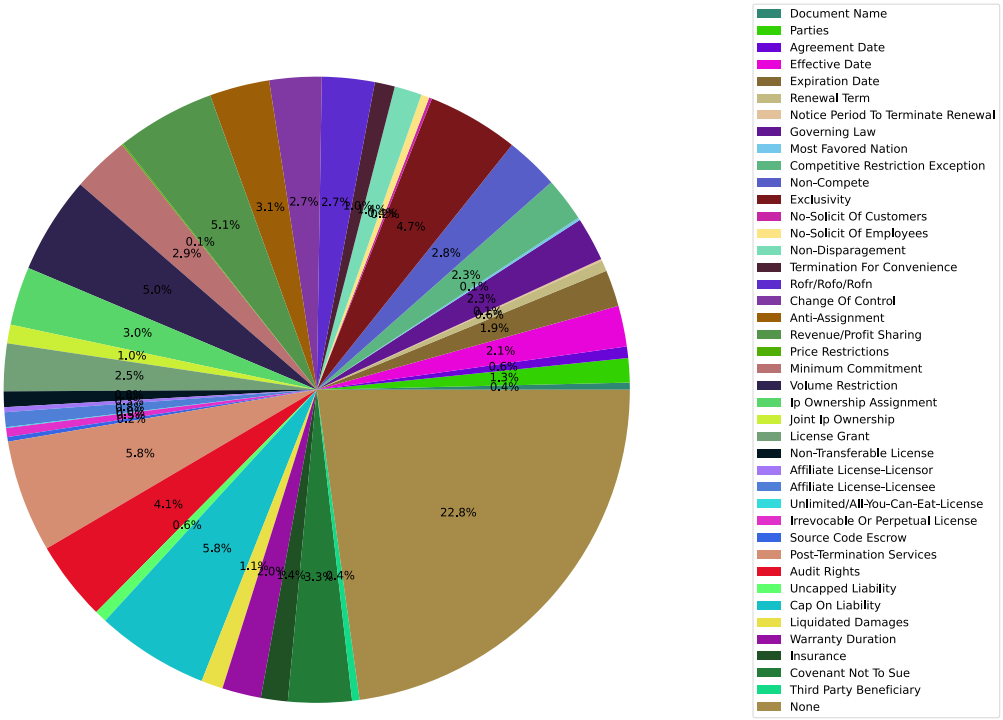

Figure 15: Pie Chart of Predicted Counts in Table 10

Table 11: Rand Model Test Counts

Class Name	Class	Clause	Clause %	Actual	AC %	Predicted	PC %
Document Name	0	102	6.92	1114	20.19	17	0.31
Parties	1	102	6.92	714	12.94	64	1.16
Agreement Date	2	90	6.11	240	4.35	13	0.24
Effective Date	3	76	5.16	311	5.64	96	1.74
Expiration Date	4	81	5.50	219	3.97	104	1.88
Renewal Term	5	43	2.92	83	1.50	40	0.72
Notice Period To Terminate Renewal	6	22	1.49	43	0.78	9	0.16
Governing Law	7	87	5.90	421	7.63	141	2.56
Most Favored Nation	8	8	0.54	14	0.25	10	0.18
Competitive Restriction Exception	9	18	1.22	50	0.91	143	2.59
Non-Compete	10	32	2.17	118	2.14	160	2.90
Exclusivity	11	32	2.17	123	2.23	229	4.15
No-Solicit Of Customers	12	14	0.95	49	0.89	3	0.05
No-Solicit Of Employees	13	17	1.15	37	0.67	21	0.38
Non-Disparagement	14	9	0.61	49	0.89	89	1.61
Termination For Convenience	15	31	2.10	127	2.30	37	0.67
Rofr/Rofo/Rofn	16	13	0.88	149	2.70	130	2.36
Change Of Control	17	33	2.24	173	3.14	145	2.63
Anti-Assignment	18	77	5.22	329	5.96	193	3.50
Revenue/Profit Sharing	19	40	2.71	146	2.65	226	4.10
Price Restrictions	20	2	0.14	4	0.07	1	0.02
Minimum Commitment	21	35	2.37	198	3.59	161	2.92
Volume Restriction	22	15	1.02	34	0.62	229	4.15
Ip Ownership Assignment	23	27	1.83	225	4.08	203	3.68
Joint Ip Ownership	24	8	0.54	41	0.74	52	0.94
License Grant	25	52	3.53	378	6.85	191	3.46
Non-Transferable License	26	28	1.90	180	3.26	32	0.58
Affiliate License-Licensor	27	6	0.41	85	1.54	13	0.24
Affiliate License-Licensee	28	10	0.68	94	1.70	52	0.94
Unlimited/All-You-Can-Eat-License	29	3	0.20	7	0.13	3	0.05
Irrevocable Or Perpetual License	30	12	0.81	57	1.03	16	0.29
Source Code Escrow	31	2	0.14	33	0.60	8	0.14
Post-Termination Services	32	35	2.37	288	5.22	307	5.56
Audit Rights	33	41	2.78	117	2.12	215	3.90
Uncapped Liability	34	19	1.29	36	0.65	47	0.85
Cap On Liability	35	48	3.26	124	2.25	331	6.00
Liquidated Damages	36	18	1.22	55	1.00	77	1.40
Warranty Duration	37	14	0.95	34	0.62	85	1.54
Insurance	38	37	2.51	180	3.26	82	1.49
Covenant Not To Sue	39	26	1.76	105	1.90	212	3.84
Third Party Beneficiary	40	7	0.47	36	0.65	18	0.33
None	41	102	6.92	5516	99.96	1313	23.79

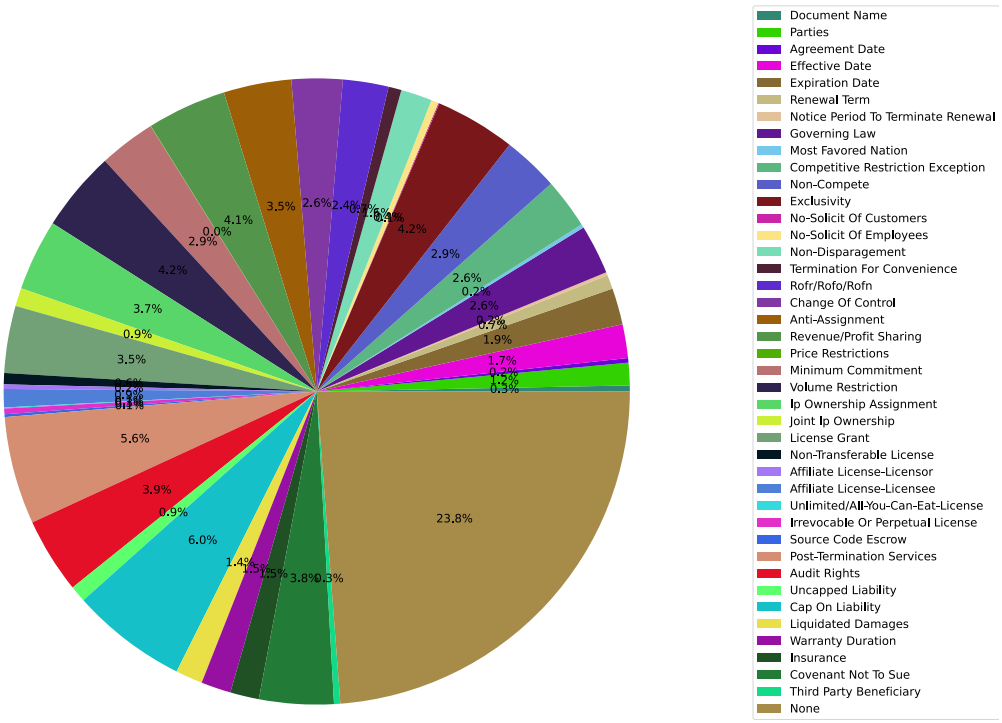

Figure 16: Pie Chart of Predicted Counts in Table 11

Bibliography

1. Joe Shields. Over 98% of Fortune 500 Companies Use Applicant Tracking Systems (ATS), 2018.

2. Society for Human Resource Management. SHRM Customized Talent Acquisition Benchmarking Report. Technical report, 2019.

3. Marisa Alia-Novobilski. AFMC civilian hiring pilot program targets efficiency, timeliness, 2018.

4. Axios HR. How Many HR People Do You Need in 2021? Technical report, 2021.

5. Marisa Alia-Novobilski. Reduced hiring timelines continue to fulfill AFMC talent needs, 2021.

6. Dan Hendrycks, Collin Burns, Anya Chen, and Spencer Ball. CUAD: An Expert-Annotated NLP Dataset for Legal Contract Review. 2021.

7. Steven Bird, Ewan Klein, and Edward Loper. *Natural Language Processing with Python*. O'Reilly Media, Inc., 2009.

8. Ngoc Giang Nguyen, Vu Anh Tran, Duc Luu Ngo, Dau Phan, Favorisen Rosyking Lumbanraja, Mohammad Reza Faisal, Bahriddin Abapihi, Mamoru Kubo, and Kenji Satou. DNA Sequence Classification by Convolutional Neural Network. *Journal of Biomedical Science and Engineering*, 09(05):280–286, 2016.

9. Felipe Almeida and Geraldo Xexéo. Word Embeddings: A Survey. (1991), 2019.

10. Georgios Paltoglou and Mike Thelwall. A study of Information Retrieval weighting schemes for sentiment analysis. *ACL 2010 - 48th Annual Meeting of*

the Association for Computational Linguistics, Proceedings of the Conference, (July):1386–1395, 2010.

11. Daniel Cer, Yinfei Yang, Sheng yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Céspedes, Steve Yuan, Chris Tar, Yun Hsuan Sung, Brian Strope, and Ray Kurzweil. Universal sentence encoder for English. *EMNLP 2018 - Conference on Empirical Methods in Natural Language Processing: System Demonstrations, Proceedings*, pages 169–174, 2018.

12. Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *1st International Conference on Learning Representations, ICLR 2013 - Workshop Track Proceedings*, pages 1–12, 2013.

13. Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

14. Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

15. Irina Rish. An Empirical Study of the Naïve Bayes Classifier An empirical study of the naive Bayes classifier. *Cc.Gatech.Edu*, (January 2001):41–46, 2014.

16. Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning*. 2013.

17. Tomas Pranckevičius and Virginijus Marcinkevičius. Comparison of Naive Bayes, Random Forest, Decision Tree, Support Vector Machines, and Logistic Regression Classifiers for Text Reviews Classification. *Baltic Journal of Modern Computing*, 5(2):221–232, 2017.

18. Meina Song, Wen Zhao, and E. HaiHong. KGAnet: a knowledge graph attention network for enhancing natural language inference. *Neural Computing and Applications*, 32(18):14963–14973, 2020.

19. Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 2017-Decem, pages 5999–6009, 2017.

20. Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. Transformers: State-of-the-art natural language processing. *arXiv*, 2019.

21. Ilias Chalkidis, Manos Fergadiotis, Prodromos Malakasiotis, Nikolaos Aletras, and Ion Androutsopoulos. LEGAL-BERT: The Muppets straight out of Law School. *arXiv*, (i), 2020.

22. Baoxu Shi, Shan Li, Jaewon Yang, Mustafa Emre Kazdagli, and Qi He. Learning to Ask Screening Questions for Job Postings. *SIGIR 2020 - Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 549–558, 2020.

23. Sunil Kumar Kopparapu. Automatic extraction of usable information from unstructured resumes to aid search. *Proceedings of the 2010 IEEE International Conference on Progress in Informatics and Computing, PIC 2010*, 1:99–103, 2010.

24. Pradeep Kumar Roy, Sarabjeet Singh Chowdhary, and Rocky Bhatia. A Machine Learning approach for automation of Resume Recommendation system. *Procedia Computer Science*, 167(2019):2318–2327, 2020.

25. Maura R. Grossman, Gordon V. Cormack, and Adam Roegiest. Automatic and semi-Automatic document selection for technology-Assisted review. *SIGIR 2017 - Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 905–908, 2017.

26. Sachin Pawar, Rajiv Srivastava, and Girish Keshav Palshikar. Automatic gazette creation for named entity recognition and application to resume processing. *5th ACM COMPUTE Conference: Intelligent and Scalable System Technologies, COMPUTE 2012*, 1(212):1–7, 2012.

27. Shuqing Bian, Xu Chen, Wayne Xin Zhao, Kun Zhou, Yupeng Hou, Yang Song, Tao Zhang, and Ji Rong Wen. Learning to match jobs with resumes from sparse interaction data using multi-view co-teaching network. *arXiv*, pages 65–74, 2020.

28. Rohini Nimbekar, Yoqesh Patil, Rahul Prabhu, and Shainila Mulla. Automated Resume Evaluation System using NLP. *2019 6th IEEE International Conference on Advances in Computing, Communication and Control, ICAC3 2019*, pages 17–20, 2019.

29. Chamila Maddumage, Dulanjaya Senevirathne, Isuru Gayashan, Tharusha Shehan, and Sagara Sumathipala. Intelligent Recruitment System. *2019 IEEE 5th International Conference for Convergence in Technology, I2CT 2019*, 2019.

30. Maria Tikhonova and Anastasia Gavrishchuk. NLP methods for automatic candidate's CV segmentation. *2019 International Conference on Engineering and Telecommunication, EnT 2019*, 2019.

31. Yanyuan Su, Jian Zhang, and Jianhao Lu. The Resume Corpus: A Large Dataset for Research in Information Extraction Systems. *Proceedings - 2019 15th Interna-*

tional Conference on Computational Intelligence and Security, CIS 2019, pages 375–378, 2019.

32. Rain Dartt. CSCE 823 Project Report, 2021.

33. Rain Dartt. CSCE 623 Project Report, 2021.

Acronyms

AFMC Air Force Materiel Command. iv, 1, 2, 1

AFPC Air Force Personnel Center. 1, 2, 3, 5, 14

AI artificial intelligence. 1, 15

ATS Applicant Tracking System. 1, 2, 3

ATSs Applicant Tracking Systems. 12

AUPR Area Under Precision-Recall curve. 3

CSV comma-separated values. 4, 14, 19

CUAD Contract Understanding Atticus Database. iv, 2, 3, 4, 6, 14, 16, 18, 20, 23, 30, 36, 1

DAF Department of the Air Force. 1

FFNN feedforward neural network. 17, 21

HR human resources. 1, 3

NER named entity recognition. 12

NLP natural language processing. iv, 1, 6, 7, 12, 15, 1

RFC random forest classifier. 10

TF-IDF term frequency-inverse document frequency. iv, 7, 8, 15, 16, 17, 20, 25, 1

USE universal sentence encoder. 9, 15, 17

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From — To)*
24–03–2022	Master's Thesis	Sept 2020 — Mar 2022

4. TITLE AND SUBTITLE

EVALUATING SEMANTIC MATCHING TECHNIQUES FOR TECHNICAL DOCUMENTS

5a. CONTRACT NUMBER

5b. GRANT NUMBER

20G353

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Rain F. Dartt

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT-ENG-MS-22-M-021

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

AFRL/ACT3
Building 620
WPAFB OH 45433-7765
DSN 713-857, COMM 937-713-8575
Email: michael.mendenhall.1@us.af.mil

10. SPONSOR/MONITOR'S ACRONYM(S)

AFRL/ACT3

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

DISTRIBUTION STATEMENT A:
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Machine learning models that employ NLP techniques have become more widely accessible, making them an attractive solution for text and document classification tasks traditionally accomplished by humans. Two such use cases are matching the specialized experience required for a job to statements in applicant resumes, and finding and labelling clauses in legal contracts The AFMC has an immediate need for solutions to civilian hiring. However, there is currently no truth data to validate against. A similar task is contract understanding for which there is the CUAD, a recently published repository of 510 contracts manually labelled by legal experts. The presented semantic matching approach first extracts, preprocesses and embeds contract clauses into a 512-dimesnion TF-IDF feature vector. Four logistic models are trained on a subset of these vectors. Then, the models are tuned to accept the contracts as text documents split into sliding windows of words. Next, the model performances are measured on a previously isolated test set and compared against the transformer models employed in the original CUAD research.

15. SUBJECT TERMS

NLP, semantic matching, machine learning, supervised learning

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Captain Rain F. Dartt, AFIT/ENG
U	U	U	UU	70	19b. TELEPHONE NUMBER *(include area code)* (931) 249-6550; rain.dartt@spaceforce.mil