Data Shift in Legal AI Systems

Venkata Nagaraju Buddarapu
LexisNexis
Raleigh, USA
venkatanagaraju.buddarapu@lexisnexis.com

Arunprasath Shankar
LexisNexis
Raleigh, USA
arunprasath.shankar@lexisnexis.com

ABSTRACT

One of the fundamental assumptions with any machine learning (ML) system is that training data comes from the same distribution as the real world data. However, in many real-world applications, this important assumption is often violated including legal research. A scenario where training and test samples follow different *input* distributions is known as **covariate shift**. This shift in data is often responsible for the deterioration in predictive performance of machine learning systems. The motivation of this research is to study the effect of covariate shift on deep learning systems used in legal research. In this paper, we propose a unified framework to detect covariate shift impacting AI systems and formulate a strategy to adapt to this shift on a periodic basis. To our knowledge, our work is the first to apply data shift detection and adaption techniques to deep learning systems involving high dimensional word embeddings. Through experiments and evaluations, we demonstrate that our framework can accurately detect data (covariate) shift on legal AI systems involving deep neural architectures.

1. INTRODUCTION

CALR stands for **C**omputer **A**ssisted **L**egal **R**esearch and is a mode of legal research that uses electronic databases that comprises court documents, statutes, secondary materials etc. Professional lawyers and paralegals rely on CALR applications for the precise understanding of the law and to serve the client's best interest. Search engines are a crucial component of legal research technology today, and its primary goal is to identify and retrieve information needed to support legal decision making. When a user types the query "*most cited cases by judge john d roberts*", he strives to understand the most often cited cases by a judge and tries to anticipate the judge's behavior. This plays a crucial role in uplifting his legal research experience. Consequently, understanding a query intent is essential for providing better search results, thus improving customers' overall satisfaction.

Understanding a query intent requires classifying legal queries and identifying domain-specific legal entities, which is a complex problem [1]. E.g., in the query: "*what are the opinions by judge john doe in civil cases dealing with dog bites ?* ", the word "*judge*" can be treated as a judge search when observed along with the context phrase "*opinions by*". The phrase "*civil cases*" can be identified as a <**practice area**> when seen alongside a supporting context and similarly "*dog bites*" can be treated as keywords. However, since we also observe the interrogative phrase "what are", we can safely assume the topic of this query is about the opinions of a judge. Table 1 presents examples for legal queries with different intents.

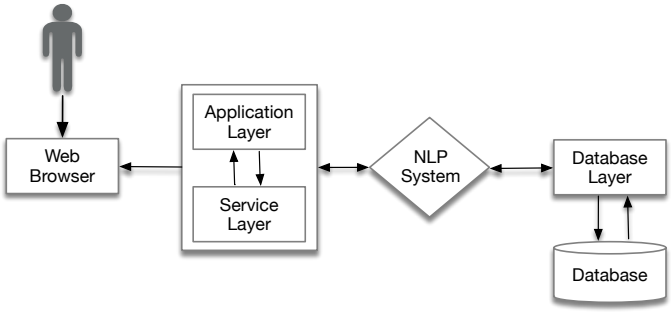

Figure 1: CALR Workflow

Intent	Example
case search	marbury v. madison, 1803
judge	judge john roberts
expert witness	expert henry lee
definitions	foreign corrupt practices act of 1977 ?
seminal cases	seminal cases on murder
burden of proof	burden of proof for hearsay statement
doctrine	what is assumed duty doctrine
elements	elements of child abuse
statutes	statute of limitations for mail fraud

Table 1: Query Intents

Identifying query intent is a classification problem, and the process of recognizing domain-specific entities is known as named entity recognition (NER), which also belong to the classification family. In general, intent and entity recognition are two primary components of any natural language processing (NLP) system. Over the past decade, the field of NLP has heavily influenced the way legal search works, shifting discovery from pure keyword-based methodologies to a more context-oriented NLP techniques.

Figure 1 depicts the workflow of a typical CALR application. The browser is a tool by which users provide input; the application layer coordinates user interactions with a service layer that triggers a search. The NLP system complements the service layer for query understanding by leveraging a database layer. The database layer usually retrieves relevant information in the form of legal documents.

NLP systems are usually built using supervised approaches and is a type of learning that uses a function to map a given input to an output. It infers learning features from labeled data consisting of training examples. For example, given a query "*what are the opinions by judge john doe in civil cases dealing with dog bites ?*", the

In: Proceedings of the Third Workshop on Automated Semantic Analysis of Information in Legal Text (ASAIL 2019), June 21, 2019, Montreal, QC, Canada.

Published at http:// ceur-ws.org.

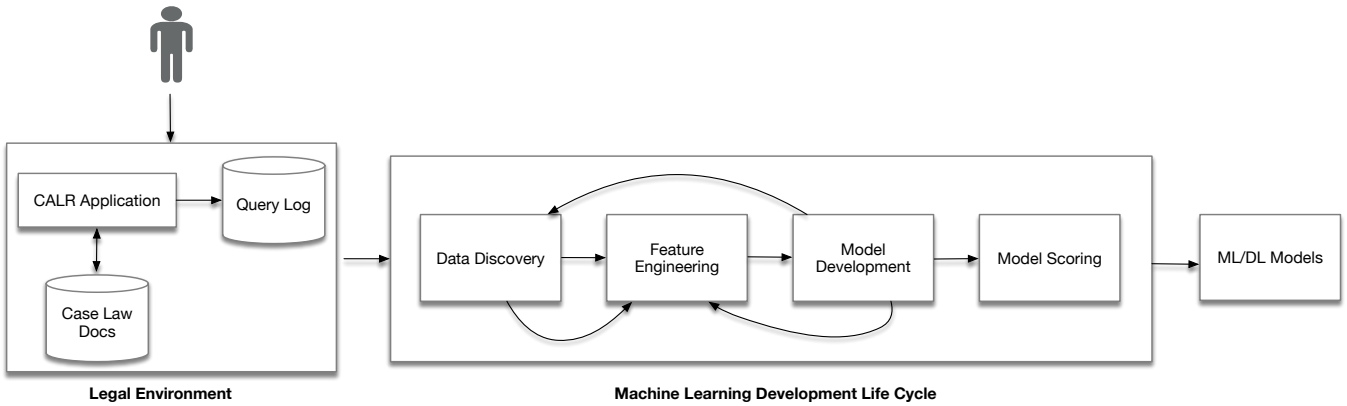

Figure 2: QIC and LER Model Development Lifecycle

output label for intent classification is "**judge**". On the other hand, we need to construct two output labels "john doe" as <**judge entity**> and "civil cases" as <**practice area**> for the task of legal entity recognition.

Our NLP system mainly consists of two models: (i) a model for identifying legal query intent which we call as Query Intent Classifier (QIC), and (ii) a model for recognizing legal entities called as Legal Entity Recognition (LER). QIC and LER model development cycles follow a standard machine learning development life cycle as shown in Figure 2. These cycles usually require application data as a prerequisite. In our case, the data is derived from user logs as mentioned previously. NLP systems usually reside within a learning environment and learn from the data collected within this environment.

In general, any legal learning environment is comprised of users (lawyers and paralegals), continually changing legal corpus and reformulating legal queries. Most machine learning methods assume the learning environments to be static, which is not the case with real-world applications such as CALR, email spam filters, stock market prediction systems etc. Real world applications including legal systems are mostly dynamic in nature and often incur distribution changes to its underlying data. This phenomenon is known as *data shift* in the machine learning arena. These data shifts usually results in performance degradation of NLP systems deployed as real-world applications.

The evolving nature of legal environment demands continuous monitoring and adaption to data shifts, in order to alleviate the issue of performance degradation in NLP systems. Data shift has been receiving significant attention in recent years amongst the machine learning community. Dataset shift refers to the problem where training and real-world datasets follow different distributions. The section 2. contains formal definitions of these shifts. Since this problem may occur in many real-life scenarios, detecting and adapting to dataset shift becomes a vital research aspect in machine learning. This research aims to observe, detect and adapt covariate shift on deep learning models using high-dimensional word embeddings, derived from a corpus of legal queries. We demonstrate the usefulness of adapting covariate shift with incremental learning on deep neural models as a necessary step to ensure consistent quality amongst AI applications deployed for legal CALR.

2. BACKGROUND KNOWLEDGE

Dataset shift research on machine learning classification algorithms is interesting and foreseen to become a more difficult problem to solve in non-stationary environments. In section 2.1, we introduce the dataset shift definitions concerning classification problems from [2]'s extensive literature survey, section 2.2 discusses the causes of dataset shift in general and 2.3 the various analysis methods for covariate shifts.

2.1 Data Shift Types

In this section, we explain the different classification and data shift types. In general, a classification problem is defined by:

- A set of features or covariates X
- A target (class) variable Y
- A joint distribution $P(Y, X)$

$X \rightarrow Y$ problems are those where the class labels Y is predicted based on the values of covariates X. Inversely, $Y \rightarrow X$ problems are the ones where the class label Y causally determines the values of covariates X. Thus, by analyzing the relationship between X and Y, we can define three different types of data shifts:

2.1.1 ***Covariate Shift****:* Covariate shift refers to changes in the distribution of input variable $X = \{x_1, x_2 \cdots x_n\}$. Here $x_1, x_2 \cdots x_n$ are called the covariates and any distribution changes in one or more of these covariates is termed as covariate shift.

2.1.2 ***Prior Probability Shift****:* Distribution changes to the class variable Y is referred to as prior probability shift, and it appears only in $Y \rightarrow X$ problems.

2.1.3 ***Concept Shift****:* Concept shift occurs when the relationship between input variable X and class variable Y changes. There other shifts in theory, but we are not defining them since they emerge sporadically. In general, dataset shift is a phenomenon that occurs when new data distribution leads to a change in the distribution of a single feature, a combination of features, or class boundaries.

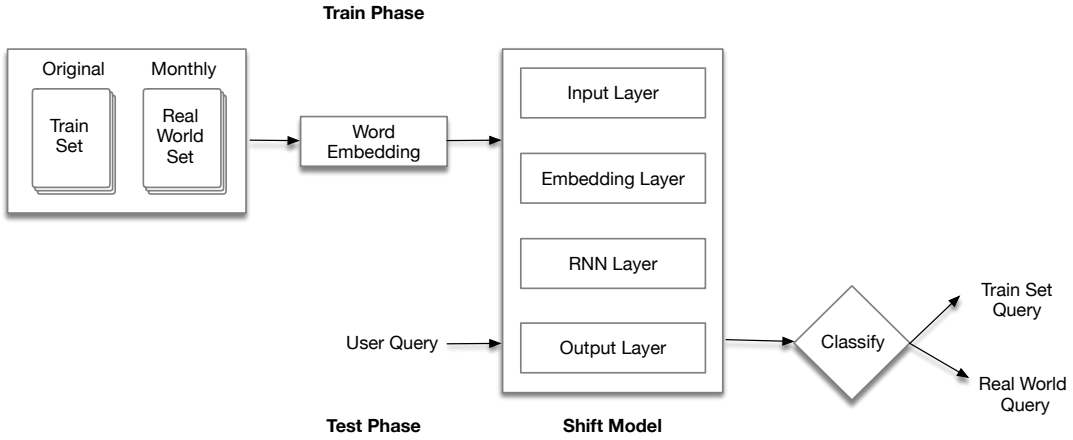

Figure 3: Shift Model

2.2 Data Shift Causes

Selection bias and non-stationary environments are the two primary reasons for data shift in general. *Selection bias* occurs when a training set does not exactly represent a real world test set. Lack of randomness in training sample selection, improper samples, and biased sampling rules often influence selection bias. On the other hand, *non-stationary* environments often must deal with dynamic nature. Hence, not handling dataset shift in real-world application creates an **overfitted** model on training samples, hence unreliable model predictions.

2.3 Covariate Shift Analysis

In this section, we introduce an overview of three well known covariate shift analysis methodologies.

2.3.1 ***Visualization****:* This methodology is the simplest, visualizing one covariate at a time. It requires humans spotting the difference in covariate distribution using histograms.

2.3.2 ***Statistical Distance****:* In this type of analysis, methods involving statistical metrics such as mean, variance, population stability index (PSI), Kullback-Leibler divergence, and Kolmogorov-Smirnov etc. are used for detecting shifts.

2.3.3 ***Uncertainty Quantification****:* This method fits a probabilistic model on the training data and every prediction on new data is associated with a confidence interval or uncertainty. Lower uncertainty on new real world data is considered no shift and higher uncertainty means a shift.

All the methodologies mentioned above have a common drawback of not being suitable for the analysis of high dimensional features. In our work, we strive to overcome this drawback by defining a shift detection algorithm to capture performance degradation in real world machine/deep learning systems, especially scoped towards legal data.

3. RELATED WORK

Data (covariate) shift is an area of machine learning that has been gaining popularity in recent years. In this section, we will discuss some of the very few existing works related to this sub-domain of AI research. First, statistics based identification methods have been widely adopted in several fields recently yielding good results. For the area of data shift, in [3][4][5], statistical methods such as exponential weighted moving average and Kolmogorov-Smirnov were used as detection methodologies especially towards time-series data, and big data online streams applications. In [6], the authors discuss hierarchical hypothesis testing techniques for concept shift detection in streaming applications.

Most real-world deep learning applications need training and the training phase usually face an internal covariate shift. In [7], Sergey et al. proposed an ensemble of batch normalized networks to detect shifts in image classification. Word embeddings are considered as the building blocks for NLP and the problem of choosing a right embedding for a particular NLP task is always a problem of trial and error. In [8], authors have discussed the various factors influencing a word embedding's stability, and one such factor is word frequency. Insufficient vocabulary affects word frequency and landing a perfect real-world sufficient vocabulary is not a one-step process. In our paper, we demonstrate that continuously updating word embeddings to represent real world data promotes the model's performance.

In [9], the researchers propose a novel minimax approach for regression problems under covariate shift. Non-stationary environments influence and change the machine learning development process. Under covariate shift, the standard model selection techniques such as cross-validation do not work as expected. Hence, an importance-based weighted cross validation strategy was proposed in [10]. However, this method necessitates the presence of covariate shift during the development phase. Sample re-weighting and active learning are well-known methods for adapting covariate shift. Sample re-weighting re-weights every training point in the learning process based on the probability of a being inside the

	QIC F_{m_0} = 0.9344				LER F_{m_0} = 0.8773			
Month	P	R	F_m	Δ	P	R	F_m	Δ
m_1	0.9821	0.9687	0.9725	+0.038	0.9613	0.8958	0.9123	+0.350
m_2	0.8194	0.7708	0.7726	−0.1618	0.6291	0.6102	0.6195	−0.2578
m_3	0.9531	0.8541	0.8669	−0.0675	0.9028	0.7149	0.7979	−0.0794
m_4	0.875	0.8041	0.8380	−0.0964	0.7821	0.6073	0.6837	−0.1936
m_5	0.9791	0.9583	0.9636	+0.0292	0.9613	0.8958	0.9123	+0.0350
m_6	0.7777	0.6999	0.7042	−0.2302	0.6962	0.6444	0.6599	−0.2174
m_7	0.8697	0.7916	0.7707	−0.1637	0.7851	0.7185	0.7289	−0.1484
m_8	0.8779	0.8291	0.8396	−0.0948	0.7248	0.6999	0.7018	−0.1755

Table 2: Current System - Real World Performance Metrics (Monthly)

training set. When adequate samples are available for the training set, active learning is adapted. Active learning selects test instances that dramatically influence the learning process and hopes to reduce the uncertainty under covariate shift. Some earlier works [11][12] have discussed these approaches.

In this paper, we scope our research to handling covariate shift in word embeddings acquired from legal search queries. It also discusses an incremental learning approach for adapting covariate shift to legal AI systems. To our knowledge, this work is the first of its kind to apply data shift on word embeddings focussing on deep learning applications. It is also the first to apply it to legal domain space.

4. THE PROPOSED FRAMEWORK

4.1 Current System

Deep learning (DL) systems learn representations of data with multiple levels of abstraction and are composed of several processing layers. These methods have dramatically improved the state-of-the-art in NLP empowered by word embeddings. Learning a high dimensional dense representation for vocabulary terms, also known as a word embedding, has recently attracted much attention in NLP and information retrieval tasks. The embedding vectors are typically learned based on term proximity from a large corpus and are used to accurately predict adjacent word(s), given a word or context.

For the purpose of this study, we consider two NLP models which we had developed earlier: (i) a model for identifying legal query intent namely Query Intent Classifier (QIC), and (ii) a model for recognizing legal entities termed as Legal Entity Recognition (LER). Our DL models follow a similar architecture as described in Figure 3. They consist of four layers - input, embedding, recurrent neural network (RNN) and an output layer. The input layer receives a dense representation of the word vocabulary derived from legal queries contained in user logs. The vocabulary is a diverse mixture of legal query types (intent), e.g., judge queries, case search, legal definitions and others shown in Table 1.

For the embedding layer, we use pre-trained word embeddings trained via a word2vec[13] model using ∼1M queries derived from user logs. The RNN layer consists of bi-directional Long Short Term Memory (LSTM) units primarily used for sequence to sequence learning. In the output layer, IOB tags [14] were used to generate labels for LER. For intent classification, since the task is a multi-class problem, we grouped and labeled all of the queries into 4 classes - judge, expert witness, seminal cases, and other.

4.2 The Problem

Legal data, in general, is both complex and diverse. User queries and word vocabularies extracted from these queries change over time. This in turn leads to changes in the underlying word embeddings which are usually the core components behind AI system(s). A word's embeddings are vectors that represent some aspect of its meaning and are generally trained on large, unlabeled corpora (in our work - legal queries). Any change in word embeddings results in complications and inconsistencies within feature weights that are part of the embedding matrix. Furthermore this change makes it harder to accomplish a consistent prediction model whose behavior does not change frequently in production.

Our legal queries were both natural and un-natural, meaning, natural queries are mostly synthesized (augmented by us) and the un-natural queries are those derived from user logs. Users tend to type queries in more un-natural format. E.g., the query, *"justice marshall abortion law 2017"* is a very un-natural representation of language having multiple intents. Also, more structured queries like boolean queries are also un-natural in its representation.

Word embeddings are almost universally useful across a wide range of tasks, but there were many limitations to this method. Word embeddings are generally used for shallow language modeling tasks, so there is a limitation to what the word embeddings can capture. Unlike RNNs and other complex architectures, language models like word2vec have trouble capturing the meaning of combinations of words, negation, etc. On the other hand, instead of training a model to map a single vector for each word, RNNs learn to map a vector to each word based on the entire sentence/surrounding context.

Another key limitation is that word embedding models do not take context into account. For instance, the word "*lynch*" has different meanings. According to California Penal Code 405a, "*Lynch*" is defined as, "the taking by means of a riot of any person from the lawful custody of any police officer." It also refers to killing someone without legal authority, usually by hanging; and "*lynch law*"

refers to the punishment of presumed crimes or offenses, usually by death, without due process of law. The above example illustrates word embeddings built on non-stationary legal vocabulary are susceptible to data shift. Therefore, it is a necessity to develop strategies and techniques to overcome the issue. In the next section, we discuss the performance degradation of our DL models observed over 8 months of experimentation showing a covariate shift.

4.3 Performance Degradation

To evaluate our DL models, we chose $F1$ score as a metric. In Table 2, m_0 denotes the initial 0^{th} month, and F_{m_0} denotes the corresponding $F1$ score for models - QIC and LER for that month (m_0). After development, the models were deployed to production in month m_0. At m_0, QIC's $F_{m_0} = 0.9344$ and LER 's $F_{m_0} = 0.8733$. These $F1$ scores set the baseline for performance comparisons, and Δ represents performance gain or loss correlated to these baselines.

The initial performance score observed during model development was good, significant performance degradation was observed after month m_0, QIC and LER model performance scores and the observed shifts (delta values) for months m_1 to m_8 are shown in Table 2. Months m_1 and m_5 saw no significant performance degradation. Remaining months {m_2,m_3,m_4,m_6,m_7,m_8} are the ones with significant performance degradation. Month m_6 witnessed a maximum QIC degradation where $\Delta = -0.2302$ and month m_2 observed a maximum LER degradation whose $\Delta = -0.2578$.

Although, we achieved good baseline performance results, the degradation after deployment prompted us to research on the cause of degradation. Our analysis discovered that covariate shift in legal user queries consequently influences legal word embeddings causing distribution changes. These changes include observing vocabulary differences such as new words or part of speech patterns. There is not enough research to identify word embedding distribution changes or covariate shift in legal (or any) environment. In this work, we propose a unique algorithm to detect covariate shift in the legal queries as explained in the upcoming section.

4.4 Algorithm

Previously, we talked about how our system suffers from covariate shift and Table 2 showed changes in $F1$score clearly showing performance degradation. Our proposed algorithm detects covariate shift in legal user queries that impacts high dimensional word embeddings that derive from it. The degradation part was discussed earlier. This section first introduces the intuition behind the algorithm. Next, it discusses notations used for defining the algorithm, followed by explanation and results.

4.4.1 ***Intuition:*** The core intuition behind the proposed algorithm is to detect covariate shift by classifying "**new**" real-world data (legal queries) as similar or different to the training data ("**old**"). The algorithm starts with building a binary classifier over the combined dataset (proprietary to LexisNexis) of current training and monthly user queries, and then predicts a probability that a user query is a member of a training set. We assign membership to both training (old) and real-world (new or test) data with output labels - **train** and **test** to create input-output pairs for the shift classifier (binary). Inconsiderable training error (covariate shift) in new user queries and limited accuracy indicates that new real-world data and training data are similar. If significant word distributions shift in the real-world test queries, then the classifier correctly classifies test queries from training queries, hence proving a distribution change in the covariates or word embeddings.

Symbol	Usage
X_t	Variable denoting current training set of user queries
Y_t	Target variable denoting output labels for X_t
$(X, Y)_t$	Variable denoting input-output pairs for X_t and Y_t
X_m	Variable denoting current month's user queries
Y_m	Target variable denoting output labels for X_m
$(X, Y)_m$	Variable denoting input-output pairs for X_m and Y_m
X_{tm}	Variable denoting combined queries of X_t and X_m
Y_{tm}	Target variable output labels for X_{tm}
$(X, Y)_{tm}$	Variable denoting input-output pairs for X_{tm} and Y_{tm}
M^{Δ}	Binary classifier for the shift model
F^{Δ}	$F1$-score of M^{Δ}
Ψ	Matthews Correlation Coefficient of M^{Δ}

Table 3: Algorithm Notations

4.4.2 ***Notations:*** Table 3 displays the list of symbols used for defining the algorithm. In general, X as input variables and Y as output variables are the X and Y to the binary classifier M^{Δ}(shift model). Along with the standard $F1$ score, an additional qualitative measure called Matthews correlation coefficient Ψ is also used for assessing M^{Δ}. For this work, we utilized a supplementary Δ symbol for all shift model related symbols. Also suffixes t and m serve as current training and current month respectively.

Algorithm 1: CSD Algorithm

Input: X_t and X_m
Output: **YES** if covariate shift, **NO** otherwise

1. Assign target labels 0 to Y_t and 1 to Y_m
2. Combine inputs X_t and X_m along with their respective output labels Y_t and Y_m to create dataset $(X, Y)_{tm}$
3. Perform the classic train-test split on dataset $(X, Y)_{tm}$ to create train set $(X, Y)^{\Delta}_{\text{train}}$ and test set $(X, Y)^{\Delta}_{\text{test}}$
4. Train a Word2Vec model using both X_t and X_m
5. Using word embeddings as features from Step 4, create a shift detection model M^{Δ} trained on $(X, Y)^{\Delta}_{\text{train}}$ and tested it on $(X, Y)^{\Delta}_{\text{test}}$
6. Compute $F1$ Score F^{Δ} and Ψ using the expected labels from Step 1 and the predicted labels from Step 5.
7. If $F^{\Delta} > 0.7$ and $\Psi > 0.2$ then return **YES** else return **NO**

4.4.3 ***Steps:*** Our proposed covariate shift algorithm(CSD) is illustrated in Algorithm 1 and contains 7 steps in total. Assignment of target (output) variables is performed in step 1 where 0 indicates a query from the current training set and 1 meaning it originated from a real-world test. Step 2 creates input-output pairs $(X, Y)_{tm}$ from (X_t, Y_t) and (X_m, Y_m). We then perform a standard train-test split of 80% train queries $(X, Y)^{\Delta}_{\text{train}}$ and 20% test queries $(X, Y)^{\Delta}_{\text{test}}$. In order to create word embeddings, a word2cec model was trained on combined dataset X_{tm} .

The architecture of our proposed shift model is shown in Figure 3 and it resembles the architecture of the previously discussed DL models (QIC and LER), (i) an input layer, (ii) an embedding layer in the form of pre-trained word2vec embedding, (iii) RNN layer consisting of LSTM units, and (iv) an output layer with sigmoid function that outputs a binary membership (0 or 1). Tokenized input queries are passed to the input layer along with its output labels, using which the shift model predicts membership (0 or 1) for the given query. The next step in the algorithm is to train a classifier M^{Δ} on $(X,Y)^{\Delta}_{\text{train}}$ and test it on $(X,Y)^{\Delta}_{\text{test}}$. To determine covariate shift, we calculate standard $F1$ test score F^{Δ} and shift score Ψ on the test set using expected labels assigned initially. If $F^{\Delta} > 0.7$ and $\Psi > 0.2$, then algorithm returns "**YES**" as an indication of covariate shift and "**NO**" otherwise. With trial and error, we arrive at the requirement that F^{Δ} should be ≥ 0.7 and also Ψ should exceed 0.2 to result in a covariate shift, and these values provide excellent coverage detecting the covariate shift for our application.

Month	F^{Δ}	Shift Score (Ψ)	Is Covariate Shift ?
m_1	0.5	0.01	No
m_2	0.96	0.22	Yes
m_3	0.53	0.17	No
m_4	0.87	0.32	Yes
m_5	0.51	0.02	No
m_6	0.92	0.38	Yes
m_7	0.89	0.26	Yes
m_8	0.98	0.54	Yes

Table 4: Detection Test Results

4.4.4 ***Results****:* Using our proposed algorithm, we conducted detection tests for 8 months as mentioned in Table 2, and Table 4 outlines the results of the tests. It comprises a shift score Ψ based on Matthews Correlation Coefficient, F^{Δ}_{m} denoting shift model's $F1$ score on current training X_{tm} and the column "Is Covariate Shift" indicating "Yes" if covariate shift and "No" otherwise. Results from Table 4 shows the algorithm was able to detect covariate shift accurately for all months except month m_3. Although the performance for month m_3 has decreased, shift detection failed to detect this with a shift score Ψ=1.7 that did not meet our threshold.

Month	Shift (Ψ)	TP	FN
m_1	0.01	✓	✗
m_2	0.22	✓	✗
m_3	0.1	✗	✓
m_4	0.32	✓	✗
m_5	0.02	✓	✗
m_6	0.38	✓	✗
m_7	0.26	✓	✗
m_8	0.54	✓	✗

Table 5: CSD Algorithm - Evaluation

To evaluate the performance of our algorithm, we considered recall R (in %) as a measuring metric. False positives trigger false alarms and result in no effect on system's performance, whereas false negatives result in system's performance degradation.

$$R = \frac{\text{Number of Shifts Detected}}{\text{Total Number of Shifts}} X100 \quad (1)$$

For our shift detection, we achieved a recall of ~80% as show in Table 5. Thus, CSD acts as an alarm and it is the first process in the two-step procedure to learn under covariate shift. In the next section, we explain how we adapt to covariate shift through incremental learning.

4.5 Incremental learning

This is the second stage of our proposed framework that facilitates adapting to covariate shift through incremental learning. Incremental learning is a method where the model's knowledge base is updated continuously.

Month	**Train Size** (X_t)
m_0	1,026,883
m_1	1,026,883
m_2	1,176,640
m_3	2,081,806
m_4	2,081,806
m_5	2,327,126
m_6	2,364,142
m_7	2,466,806
m_8	2,629,006

Table 6: Cumulative Training Size

For incremental learning, at each iteration of CSD, X_t is updated on a monthly basis. Table 6 shows the cumulative X_t size of training data after each incremental learning. The baseline training set holds $1,026,883$ queries at first month m_0 and through incremental learning, the training set is accumulated to $2,629,006$ at last month m_8. We address covariate shift in legal queries by incremental re-training at each detection. To retrain our QIC and LER models, we used an expert system which is a Java-based rule engine. Subject Matter Experts (SMEs) produced the desired output labels Y_m. The performance improvements after re-training is displayed in Table 7. It contains two $F1$ scores, (i) an original $F1$ score F_m from Table 2 before retraining and, (ii) a new latest $F1$ score $\widehat{LF_m}$ after retraining. Original performance degradation is Δ from Table 2 and the result of incremental learning is measured by subtracting the latest $F1$ score from the base $F1$ score i.e., δ=$\widehat{F_m}$ - F_m.

Apart from overcoming the performance degradation, incremental learning improved the $F1$ score over the baseline. QIC has the highest improvement $\delta = +0.144$ on m_8 and LER has $\delta = 0.0191$ on month m_2. Our experimental results demonstrate the effectiveness of the proposed covariate shift-detection and incremental learning strategy.

4.6 Environment

For training our DL and shift models, we used AWS ml.$p3.8x$large instance with 4 NVIDIA Tesla V100 GPUs. Average time taken for all models is ~94 minutes for 100 epochs, and training time for

	QIC F_{m_0} = 0.9344				LER F_{m_0} = 0.8773			
Month	F_m	Δ	LF_m	δ	F_m	Δ	LF_m	δ
m_2	0.7726	−0.1618	0.9493	+0.0149	0.6195	−0.2578	0.8964	+0.0191
m_3	0.8669	−0.0675	0.9312	−0.0032	0.7979	−0.0794	0.8032	−0.0741
m_5	0.8380	−0.0964	0.9589	+0.0245	0.6837	−0.1936	0.8790	+0.0017
m_6	0.7042	−0.2302	0.9200	−0.0144	0.6599	−0.21742	0.8529	−0.0244
m_7	0.7707	−0.1637	0.9221	−0.0123	0.7289	−0.1484	0.8400	+0.0373
m_8	0.8396	−0.0948	0.9488	+0.144	0.7018	−0.1755	0.8928	+0.0155

Table 7: Results after incremental learning

word2vec is ~84 minutes for 10 epochs. For implementing our shift and DL models, we used TensorFlow [15]. For our DL models, we fixed a batch size of 512.

5. PERFORMANCE METRICS

We utilize standard measures to evaluate the performance of our QIC, LER and shift model classifiers, i.e., precision (P), recall (R), and $F1$-measure. Precision (P) is the proportion of actual positive class members returned by our method among all predicted positive class members returned by our method. Recall (R) is the proportion of predicted positive members among all actual positive class members in the data. $F1 = 2PR/(P+R)$ is the harmonic average of precision and recall. We also utilized Matthews correlation coefficient (MCC) to compute the shift score and MCC is a correlation coefficient between actual and expected predictions. It varies between -1 and +1: -1 when actual and expected are entirely different, 1 when there is a perfect match and 0 when it is random. Accompanying shift $F1$ score, MCC Ψ was used as shift score for measuring the similarity between training and test set.

6. CONCLUSION AND FUTURE WORK

Learning strategies under covariate shift have been receiving significant research interest recently. In non-static environments such as legal, learning methods need to employ unique learning strategies and covariate shift monitoring systems to acquire a greater capability to generalize the learning. Our proposed framework in this work belongs to the category of incremental learning under covariate shift for legal AI systems, and its core component is a covariate shift detection algorithm which detects shift in our deep learning models - QIC and LER. The results demonstrate the benefit of building a monitor system for covariate shift detection and also its adaptation through incremental learning. For future work, we plan to extend our current research by employing better strategies to reduce false negatives. Also, we are plan to apply a similar strategy to other data shifts such as prior probability and concept shifts.

REFERENCES

[1] S. Arunprasath and B. Venkata Nagaraju, "Deep ensemble learning for legal query understanding," in *Proceedings of CIKM 2018 Workshop on Legal Data Analytics and Mining (LeDAM 2018)*, CEUR-WS.org, October 2018. To appear.

[2] J. G. Moreno-Torres, T. Raeder, R. Alaiz-RodríGuez, N. V. Chawla, and F. Herrera, "A unifying view on dataset shift in classification," *Pattern Recogn.*, vol. 45, pp. 521–530, Jan. 2012.

[3] H. Raza, G. Prasad, and Y. Li, "Dataset shift detection in non-stationary environments using ewma charts," *Proceedings - 2013 IEEE International Conference on Systems, Man, and Cybernetics, SMC 2013*, pp. 3151–3156, 10 2013.

[4] D. Zhao, L. Bu, C. Alippi, and Q. Wei, "A kolmogorov-smirnov test to detect changes in stationarity in big data," *IFAC-PapersOnLine*, vol. 50, pp. 14260 – 14265, 2017.

[5] H. Raza, H. Cecotti, Y. Li, and G. Prasad, "Adaptive learning with covariate shift-detection for motor imagery-based brain–computer interface," *Soft Comput.*, vol. 20, pp. 3085–3096, Aug. 2016.

[6] S. Yu, X. Wang, and J. C. Príncipe, "Request-and-reverify: Hierarchical hypothesis testing for concept drift detection with expensive labels," pp. 3033–3039, 07 2018.

[7] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *CoRR*, vol. abs/1502.03167, 2015.

[8] L. Wendlandt, J. K. Kummerfeld, and R. Mihalcea, "Factors influencing the surprising instability of word embeddings," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 2092–2102, Association for Computational Linguistics, 2018.

[9] X. Chen, M. Monfort, A. Liu, and B. D. Ziebart, "Robust covariate shift," in *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics* (A. Gretton and C. C. Robert, eds.), vol. 51 of *Proceedings of Machine Learning Research*, (Cadiz, Spain), pp. 1270–1279, PMLR, 09–11 May 2016.

[10] M. Sugiyama, M. Krauledat, and K.-R. Müller, "Covariate shift adaptation by importance weighted cross validation," *J. Mach. Learn. Res.*, vol. 8, pp. 985–1005, Dec. 2007.

[11] A. Liu and K. Asif, "Addressing covariate shift in active learning with adversarial prediction," *ICML 2015 Workshop of Active Learning*.

[12] M. Sugiyama, S. Nakajima, H. Kashima, P. V. Bunau, and M. Kawanabe, "Direct importance estimation for covariate shift adaptation," *Annals of the Institute of Statistical Mathematics*, 2008.

[13] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, (USA), pp. 3111–3119, Curran Associates Inc., 2013.

[14] L. Ramshaw and M. Marcus, "Text chunking using transformation-based learning," in *Third Workshop on Very Large Corpora*, 1995.

[15] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, "Tensorflow: A system for large-scale machine learning," in *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, OSDI'16, (Berkeley, CA, USA), pp. 265–283, USENIX Association, 2016.