A Simple yet Brisk and Efficient Active Learning Platform for Text Classification

Teja Kanchinadam, Qian You, Keith Westpfahl, James Kim, Siva Gunda, Sebastian Seith, Glenn Fung
American Family Insurance, Machine Learning Research Group
{tkanchin, qyou, kwestpa, jjkim, sgunda, sseith, gfung}@amfam.com

Abstract

In this work, we propose the use of a fully managed machine learning service, which utilizes active learning to directly build models from unstructured data. With this tool, business users can quickly and easily build machine learning models and then directly deploy them into a production ready hosted environment without much involvement from data scientists. Our approach leverages state-of-the-art text representation like OpenAI's GPT2 and a fast implementation of the active learning workflow that relies on a simple construction of incremental learning using linear models, thus providing a brisk and efficient labeling experience for the users. Experiments on both publicly available and real-life insurance datasets empirically show why our choices of simple and fast classification algorithms are ideal for the task at hand.

1 Introduction

Topic classification and identification have remained a fundamental problem in industries especially when dealing with large corpora of unstructured text, which paved way for an increasing demand for labeled data. Manual labeling such data, for example, is not only a labor intensive and time demanding task but also is subject to regulations and restrictions as only certain business owners have access rights to view such data. In this study, we propose the use of a fully managed active learning service platform deployed in an enterprise level environment which allows business teams to independently curate machine learning models for topic classification.

As stated in (Settles 2009), "The key idea behind active learning is that a machine learning algorithm can achieve greater accuracy with fewer training labels if it is allowed to choose the data from which it learns." Pool-based active learning uncertainty sampling is the most classic query strategy for choosing the next unlabeled sample to be labeled by an oracle(e.g. a human annotator). A chosen base learner is then updated by the available labeled samples and continues the uncertainty sampling process. Although recent active learning research has adapted to the latest advancement in deep learning, uncertain sampling remains a practical solution that can be widely applied real world applications due to its simplicity and proven robustness. Moreover, most of existing active learning research, both in natural language processing and computer vision tasks, focused on simulations using well studied open source data sets, while few research presented how real world applications can be benefited from active learning and what is the best practice in deploying active learning processes and models. In fact, several difficulties prevent theoretical promising active learning techniques which can be adopted by practitioners, including implementing fast and cost-effective cross-validation algorithms that can be used to tune the algorithms robustly when limited labeled data is available, and in general, utilizing human annotations more efficiently. In addition to those difficulties, one must consider the active learning system response time when designing practical active learning algorithms, especially retraining the base learner involves constantly increasing labeled data. In a user interaction study (Martin and Corl 1986), it was concluded that 0.1 seconds is the appropriate response time for users to feel like their actions are directly causing something to change in the system; response times longer than 0.1 seconds but less than 1 second will make users feel the short delay but won't distract their focus and chain of thoughts; response times longer than 1 second make users feel they are waiting too long and they will grow impatient and may eventually abandon the labeling process.

In this paper, we present a robust active learning methodology that addresses some of the above difficulties commonly seen in practice. We describe how we deploy the resulting trained model in a cloud-based environment (AWS) where it can be convoked as an API to incorporate its output into other downstream insurance workflows. Specifically, the contribution of this paper is as follows:

- We propose a simple, fast, and yet robust adaptive tuning method, that we use in this work to optimize our regularization parameter. This method synergizes very well with the active learning paradigm where speed is paramount while achieving competitive performance.
- We provide an efficient implementation (Python library) of incremental regularized least squares SVM (PSVM, LS-SVM (Fung and Mangasarian 2001; Suykens, Lukas, and Vandewalle 2000)) for incremental active learning with the aforementioned fast adaptive regularization parameter tuning that instantaneous cross-validation.
- We also discuss our system architecture and deployment methodology of the proposed active learning platform in the context of two innovative insurance-related business

applications. [1]

2 Methodology

2.1 Active Learning

The typical setup for pool-based active learning for classification is as follows: a pool of unlabeled examples $\mathcal{U}$, a pool $\mathcal{L}$ of labeled example-label pairs (x, y_x), an oracle - usually a human annotator that can supply the label of any $x \in \mathcal{U}$, and a query strategy that selects which example $x^* \in \mathcal{U}$ the oracle should label such that $\mathcal{L}^* = \mathcal{L} \cup \{(x^*, y_{x^*})\}$ yields the maximum information gain versus $\mathcal{L}$.

Uncertainty Sampling is an effective and widely used query strategy. It captures the classifier's (θ) uncertainty about the class of x, and can be given as:

$$x^* = argmax_{x \in \mathcal{U}}(\mu(x)) \tag{1}$$

where $\mu(x)$ can be defined as the Shannon Entropy in a classification setting as

$$\mu(x) = -\sum_{c \in \mathcal{C}} p_c(x) \log p_c(x) \tag{2}$$

where (C) are our possible classes, and $p_c(x)$ is the probability that our classifier assigns to x having class c.

For the rest of the paper, we will use *uncertainty sampling*, which is a simple widely-used proven query strategy and that usually produces competitive learning rates in most real-life applications. In the *uncertainty sampling* strategy $\mu(x)$ captures the classifier's uncertainty about the class of x (Lewis and Gale 1994). The intuition is that not much information is gained if a classifier gets a new label with which it already agreed with high certainty. Note for binary classification ($|\mathcal{C}| = 2$), $\mu(x)$ is maximized when $p_c(x) = .5$ for both classes. For a linear classifier, this is equivalent to finding unlabeled samples that lie closest to the decision hyperplane and can be implemented in $\mathcal{O}(|\mathcal{U}| \log |\mathcal{U}|)$ time.

2.2 Incremental Learning

One of the biggest challenges in any active learning framework is that the model has to be re-trained at every iteration of the process. Typically, a model retrains quickly at the starting iterations as only a few labeled samples are available to train the model. However, as more samples come by, training time usually increases considerably since most learning algorithms complexity depends on the number of training samples. In order to make training time grow slowly across all the iterations, we propose the use of an incremental learning mechanism that takes advantages of the simplicity of a PSVM / LS-SVM classifier to make efficient incremental updates of the model instead of calculating the model form scratch at each iteration.

least-squares-based SVM classifiers are a modified version of standard SVM where a least squares cost function is proposed so as to obtain a linear set of equations in the either the primal (PSVM) or the dual space (LS-SVM). (Suykens, Lukas, and Vandewalle 2000; Fung and Mangasarian 2001).

More succinctly, the goal is to learn a linear classifier $f(x) = \theta' x$, such that $|||f(X) - Y||^2 + ||\theta||^2$ is minimized. The closed form PSVM solution with a linear kernel, is similar to a Ridge regression solution and it can be written as:

$$\theta = (X^T X + \lambda I)^{-1} X^T Y \tag{3}$$

where $X \in R^{m \times n}$ with m rows (datapoints), n is the number of features and Y is a vector of binary label with values in $\{+1, -1\}$. θ are the coefficients of the separating hyperplane; λ is the regularization parameter.

Let $M_n^{-1} = (X^T X + \lambda I)^{-1}$, such that $\theta = M_n^{-1} \sum_{i=0}^{n-1} x_i y_i$.

Note that when a new training point (x_n, y_n) is added, the linear classifier weights can be updated incrementally as follows,

$$\hat{\theta} = \theta + M_{n+1}^{-1} x_n (y_n - x_n^T \theta) \tag{4}$$

Where the new inverse M_{n+1}^{-1} is a rank-1 update of the previous one and can be calculated very efficiently using the Sherman-Morrison-Woodbury (Deng 2011) formula as follows:

$$M_{n+1}^{-1} = M_n^{-1} - M_n^{-1} x_n (1 + x_n^T M_n^{-1} x_n)^{-1} x_n^T M_n^{-1} \tag{5}$$

where M_n^{-1} is the inverse at previous iteration and (x_n, y_n) is the new point.The inverse can be updated with a low computational cost since the most expensive operation is a matrix-vector multiplication. We will call this incremental process **LS-SVM SMW**.

Based on some ideas from the batch least squares, equation (3) can be written as

$$\hat{\theta} = \theta + M_{n+1}^{-1} \frac{1}{t} \sum_t x_t (y_t - x_t^T \theta) \tag{6}$$

where t are the number of iterations in the incremental learning process.

If we were to consider not storing and updating the inverse each time, assign $M_{n+1}^{-1} = \gamma_n$, now equation (6) can be implemented in $O(1)$ time as it is only requires a vector-vector multiplication. Note that γ_n is a scalar and it can chosen by $\gamma_n = n^{-\alpha}$ where $\alpha \in (0.5, 1.0)$. This heuristic was proposed in (Polyak and Juditsky 1992) where the idea is to average trajectories at each step, such that the solution can be obtained by averaging over all previous solutions. We call this process **LS-SVM Poly**

For both incremental methods described above, during each iteration of the active learning process, the regularization parameter λ is fixed and thus cannot be changed when new training points are added for incremental learning. Since the training set is constantly and drastically changing (especially in early iterations when the labeled training dataset is small) the regularization parameter λ has to be tuned at every iteration to optimize performance. This process can add additional computing costs that could result in slower iterations of the active learning process.

In order to address this problem, we consider two approaches.

[1] Code is available at https://github.com/tkanchin/adaptive_reg_active_learning/

1. To use Singular Value Decomposition (SVD) to factorize X. This factorization of X allows to find close-form solutions for θ as a function of λ. However, we also have to update incrementally the SVD factorization in each iteration.

2. We propose the use a simple greedy adaptive tuning regularization approach that uses properties of equation 3 to quickly find values of λ that optimize our leave-one-out cross-validation performance over a finite set of candidate λ's.

2.3 Efficiently tuning λ using SVD

Let SVD on $X = USV^T$ where $X \in R^{m \times n}$; U and V are orthonormal matrices and $U \in R^{m \times m}$, $V \in R^{n \times n}$; S is a diagonal matrix $S \in R^{m \times n}$; Equation (3) can now be reformulated as follows:

$$\theta = (VSU^TUSV^T + \lambda I)^{-1}VSU^TY \tag{7}$$

Note that the properties of SVD state that, $U^TU = I$ & $V^TV = I$. Thus, the above equation can be further reduced to,

$$\theta = V(S^2 + \lambda)^{-1}SU^TY \tag{8}$$

where $(S^2 + \lambda)$ is a diagonal matrix and its inverse can be cheaply calculated.

When a new point (x_n, y_n) is added, the U, S, V matrices are updated using a SVD rank update discussed in (Brand 2006) and solve for equation (8). Note that even though this method allows us to change λ in the incremental learning process, it is still computationally expensive because SVD rank update involves decomposing a re-diagonalized matrix.

2.4 Solving for λ: Adaptive regularization Approach

We propose a simple yet effective approach to update the regularization parameter λ in this approach. The basic idea is to train several parallel incremental learners each initialized with a λ parameter. The best learner at each step is identified by performing a leave-one-out cross validation.

Generalized Cross Validation (GCV): The running time for leave-one-out cross validation increases linearly as training set grows in incremental learning. Since, the underlying learner is LS-SVM, (Golub, Heath, and Wahba 2007) proposed a simple method to estimate the LOOCV error and is given by:

$$GCV(\lambda) = \frac{1}{n}\sum_i((y_i - \hat{y}_i)/1 - tr(H)/n)^2 \tag{9}$$

where $\hat{y}_i$ are the predictions of the model, y_i is the ground truth and H is the hat matrix and is defined as

$$H = X(X^TX + \lambda I)^{-1}X^T \tag{10}$$

Algorithm 1 Greedy adaptive regularization approach

1: Let $\hat{\theta}(\lambda^*) = \{\theta^1(\lambda^1), \theta^2(\lambda^2), ..., \theta^l(\lambda^l)\}$
2: Let $\hat{g} = \{g^1, g^2, ..., g^l\}$ be the corresponding GCV
3: **if** TRAINING **then**
4: Let (x_n, y_n) be the new training point
5: Update all learners in $\hat{\theta}(\lambda^*)$
6: Update $\hat{g}$
7: **else**
8: return best model as $\hat{\theta}[argmin(\hat{g})]$
9: **end if**

Note that the set $\{\lambda_1, \ldots, \lambda_l\}$ defines the search space, however the optimal value of $lambda$ can change between iterations.

The goal of the greedy adaptive regularization method is to prevent overfitting when tuning the linear models between iterations. Since overfitting commonly occurs on small training sets which is usually the case in active learning scenarios, adaptive regularization can enable the active-learning-produced models to achieve better generalization, especially at early iterations, and hence to produce better queries.

3 Offline Validation

3.1 Dataset

To simulate the active learning process and to test the design choices, we have used the IMDB movie reviews dataset which consists of 1000 positive and 1000 negative movie reviews. These reviews were labeled and annotated by human annotators and are best described in (Zaidan, Eisner, and Piatko 2007).

3.2 Feature Representation

To test the performance of our proposed algorithm against various feature representations, we have considered the following:

1. **W2V:** Unweighted average of the word vectors that comprise the sentence or document (Wieting et al. 2015). We we have used the publicly available Glove model (Pennington, Socher, and Manning 2014) trained on news articles.
2. **USE:** Pre-trained transformer based Universal Sentence Encoder described in (Cer et al. 2018) to map input to vector representation.
3. **BERT:** Pre-trained transformer based model described in (Devlin et al. 2018) to map input to vector representation.
4. **GPT2:** Pre-trained transformer based model described in (Radford et al. 2019) to map input to vector representation.

3.3 Incremental Active Learning Results

We have benchmarked the speed and performance of various formulations against IMDB movie reviews dataset discussed in Section 3.1 and the results are shown in Table 1. The table compares incremental learning to that of the traditional learning where the learner is re-trained with all the points at each iteration of the learning process. It is evident from the

Table 1: Table showing the performance comparison between various learning approaches. *LS-SVM SVD* refers to section 2.3, *LS-SVM SMW* and *LS-SVM Poly* refers to section 2.2

Model	Incremental	AUC	Speed (s)
LS-SVM	No	0.8588	27.32
LS-SVM SMW	**Yes**	**0.8588**	**1.629**
LS-SVM Poly	Yes	0.8289	0.072
LS-SVM SVD	Yes	0.811	628.42

table that the SVD methods are computationally expensive as they involve some heavy operations in the decomposition of matrices. The other LS-SVM methods are computationally cheap, it is important to observe that the LS-SVM Poly is fastest when compared to all the other methods as here we approximating the inverse update operations. However, since the approximation is stochastic in nature, the performance wouldn't be exact when compared to the baseline LS-SVM. Note that the LS-SVM SMW method achieves the best AUC performance while performing around 1000 updates in 1.6 seconds. This roughly translates to negligible computation times when updating the models during the labeling process.

Active Learning Process We have used a incremental LS-SVM classifier as the underlying classification model. We initialize the model with 2 randomly selected labeled examples to start the active learning simulation. In each iteration, we include one example from the unlabeled set to update the linear model until the unlabeled set is completely exhausted. We also calculate the receiver-operating-characteristic (ROC), or an AUC score of the model against a hold out test set. Metrics are reported by averaging the results from 10 runs and for the simplicity of display, we have not reported standard deviations.
In Figure 1(a), we show that the proposed incremental LS-SVM model with adaptive regularization and GPT2 features clearly outperform other text representations in an active learning setting using uncertainty sampling.

Similarly, in Figure 1(b), we show that uncertainty sampling clearly outperforms random sampling using adaptive regularized LS-SVM model.

In Figure 1(c), we show the performance of adaptive regularization in an active learning setting using uncertainty sampling. It is evident from the figure, the best model at every iteration of the training process is yielding the optimal performance.

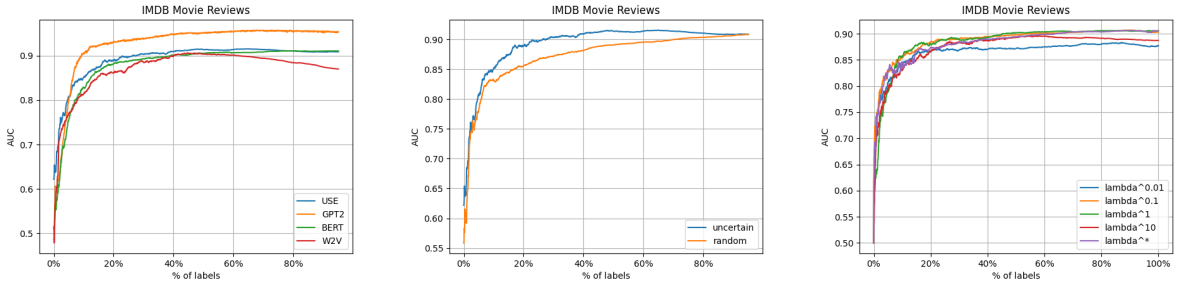
4 Platform And Applications

Based on the learning methodology discussed in Section 2, we have developed an active learning platform for users to upload and label unstructured documents (see Figure 2(a)). Once the documents are labeled and models are curated, they are directly deployed into a production ready hosted environment (see Figure 2(b)). The rest of this section is divided as follows: a) In Section 4.1, we introduce our active learning platform. b) In Section 4.2, we discuss about two innovative insurance domain specific applications. c) In Section 4.3, we briefly discuss our deployment methodology.

4.1 Active Learning Platform

We have developed an active learning platform with an interactive web-based user interface (UI) and all the services supported by this UI are orchestrated by a back-end server, designed using a light-weight and independent micro-services architecture (Thönes 2015).

1. In Figure 3(a), once the user uploads a dataset of interest to be analyzed, we automatically process this dataset using *FICO* which is a series of steps namely $\{Featurization, Indexing, Cleaning, Organization\}$.
   (a) *Organization:* There are several different types of insurance data that can flow into our system. In this step, we identify the source of the data and store them in a database.
   (b) *Cleaning:* In order to deal with the potential exposure of personal identifiable information (PII), we use regular expressions to filter out the email addresses, phone numbers, ids, people names and other characters. In addition to this, we use regular expressions to identify the numerical entities such as time, date, dollar signs, etc. and assign special tokens to them.
   (c) *Indexing:* In order for us to query our corpus for any documents containing a set of tokens/keywords, we index our data using Elasticsearch (Gormley and Tong 2015), which provides the flexibility and scalability necessary to perform a token-based search over millions of documents in near real-time.
   (d) *Featurization:* In this step, our goal is map our input to a text representation that is well suited for learning algorithms with limited labeled data. We employ a K80 GPU Amazon Web Services (AWS) service to extract features from a GPT2 model discussed in Section 3.2.
2. In Figure 3(b), the user defines a concept they wish to extract from the documents of interest using a short list of keywords that are related to that concept. The active learning platforms offers the ability to enrich this set of keywords by supplementing semantically similar words using a word embedding based model. We have trained a FastText model described in (Joulin et al. 2017) using many insurance-related text datasets available within the company to learn a rich word embedding that captures the language commonly used in the insurance domain.
3. In Figure 3(c), users can label documents as positive or negative for the defined concept. The UI will also allow users to highlight certain phrases/keywords so as to explain the rationale behind their decision. Note that the documents presented to the user for labeling are provided by an active learning algorithm powered by incremental adaptive regularized method. This makes the model update and document query to negligible computation time providing a smooth labeling experience to the users.
4. In Figure 3(c), users can track the progress and performance of their labeling tasks. It would enable users to a)



(a) (b) (c)

Figure 1: Active Learning results on IMBD movie reviews dataset (a) text representation (b) uncertainty sampling versus random sampling (c) performance of adaptive regularization

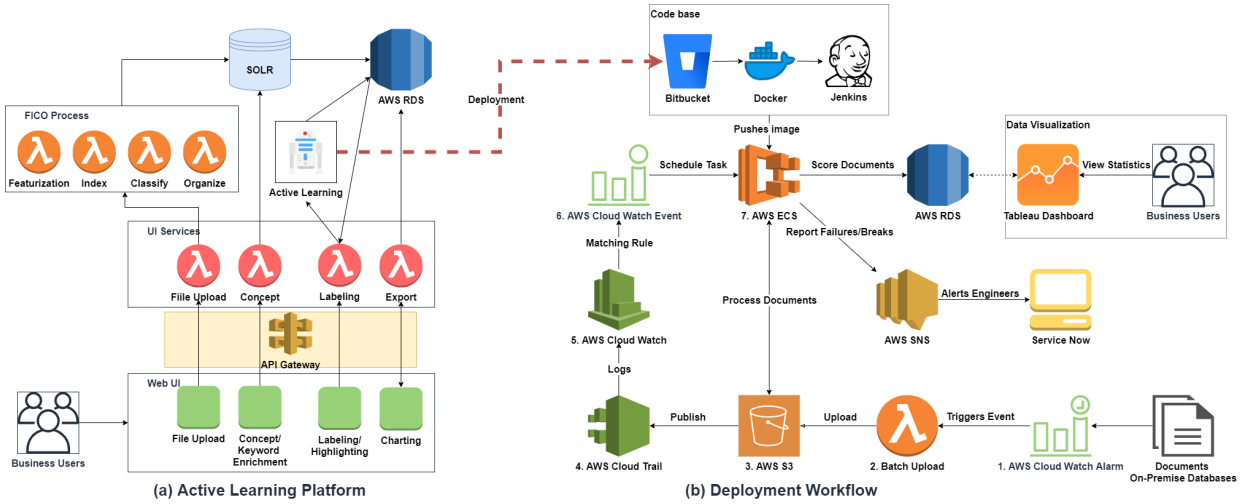


Figure 2: Active Learning Platform and Model Deployment Workflow

track the labeling progress b) model predictions on the remaining documents c) AUC performance at different steps in the labeling process. Note that every 3^{rd} labeled document (randomly selected) is used as a hold-out set to calculate AUC.

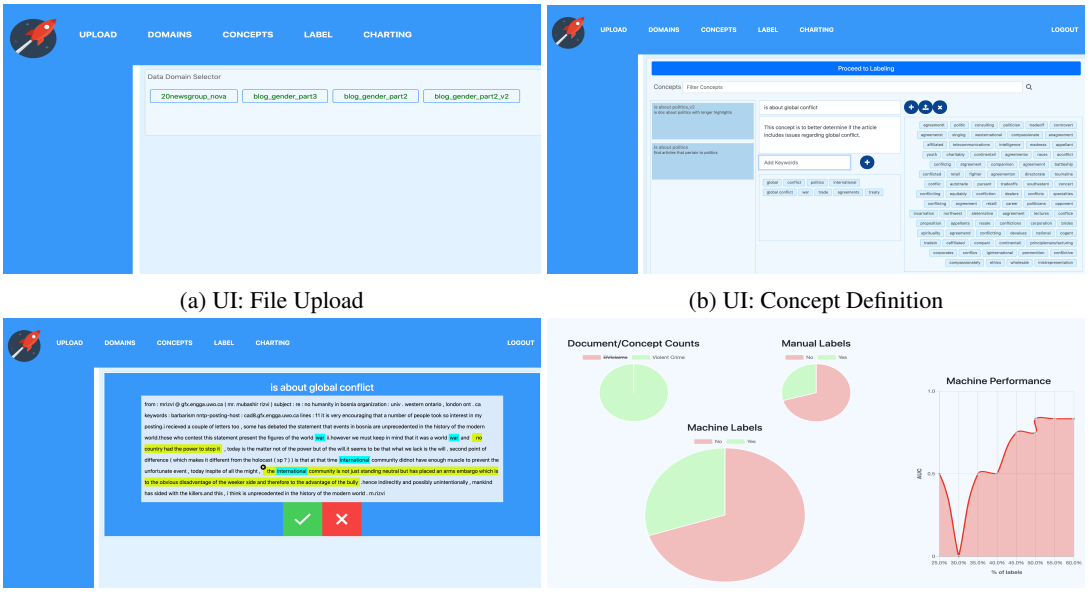
4.2 Applications for Insurance

Auto Claims Loss Causes Insurance business processes generate large amounts of unstructured text which are packed with relevant information. For example, every time we receive a claim, the request and a brief hand-typed description of the incident is stored in our internal database systems for business related purposes. Here is an example of an auto claim:

"*The front right tire on the trailer blew out and broke apart causing damage to the wheel well and possibly to the wiring on the electronic brakes for that wheel.*"

In order to produce metrics related to the different claim causes, product analysts within our company manually categorize these claims using simple heuristic rules like keywords and regular expressions. This is not only a time consuming and labor intensive job but it is also prone to human subjection and variance. This problem can be addressed by the proposed active learning platform where a) an analyst can upload documents of interest related to claims b) define a concept for the desired loss cause c) label a few documents to train models to automatically classify incoming claims into the appropriate loss cause d) deploy these models into production and automate the categorization process.

Table 2 shows the results of this application on identifying loss causes i.e. *Tire Blowout*, *Water Damage*, and *Wind Damage*, and curating machine learning models using the proposed active learning platform. From a total of 7287 auto insurance claims, the loss cause *Wind Damage* has 4350 documents retrieved using simple heuristics like keywords; analysts labeled a total of 1804 samples from these retrieved documents indicating whether the claim belongs to loss cause or not. Since every 3^{rd} labeled document is used as a hold-out set, the machine learning model curated using active learning achieved an AUC of 94.3. **model** $+^{ve}$ is obtained by applying the machine learning model to the total number of retrieved candidates. This gives an insight that the the model was able to effectively prune a good amount of the retrieved documents and only retain those that are more



(a) UI: File Upload

(b) UI: Concept Definition

(c) UI: Labeling Interface

(d) UI: Labeling Statistics

Figure 3: Interactive Web UI interface of the proposed Active Learning Platform

likely belonging to the desired loss cause.

Property Claims Audit Property claims are unstructured textual narratives containing information about building characteristics, utilities and surrounding environment. Auditors would like to audit these claims periodically to verify that claims have been operated using standard procedures. Currently, the auditing process is conducted by randomly selecting a subset of claims. This can be addressed more efficiently using our proposed active learning platform. For example, if the auditors want to audit the claims related to *"structural damage"*, they can curate a model using our proposed active learning platform and deploy the model to production. The model is then applied to a huge corpus of claims generating predictions, the auditors can rank the predictions of the model and audit the claims which are more likely related to *"structural damage"*.

Table 3 shows the results of our application on identifying topics for property claims auditing. The topics are developed from a corpus of around 140,000 property claims. From the table, most of these audit concepts are relatively rare. For example, there are only eight *"Pollution"* claims which is a small percentage. Instead of combing through hundreds of thousands of claims to find *"Pollution"* claims, the auditor can identify such claims quickly and efficiently with the proposed framework.

4.3 Deployment

The architecture of our deployment workflow is shown in Figure 2(b). Our workflow is a highly scalable event-driven system using AWS managed services operating at minimum costs. Our deployment workflow is explained as below:

1. First, we package the models along with the application code in a docker container and push the image to a container registry on the cloud. We then use Elastic Container Service (ECS) which provides a pay-per-use container orchestration engine without the maintenance of computing infrastructure to run the packaged docker image as an ECS task. This setup enables the system to vertically scale by increasing the memory and vCPUs as needed.

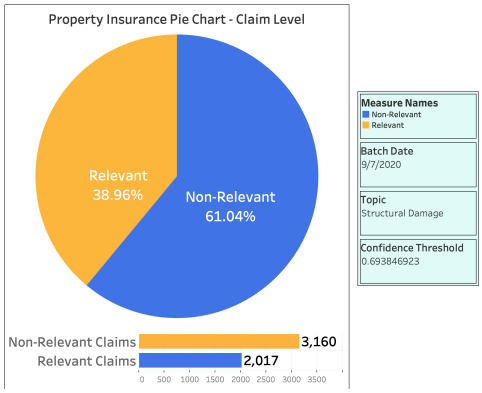
2. Second, we create batches of documents and store them on S3 using a Lambda service, which provides a pay-per-use serverless computing infrastructure.

3. Third, a CloudWatch is triggered as an ECS task once the batched documents are uploaded to S3. The ECS task reads the documents from S3, runs it against the application code and store the results in an AWS Managed Relational Database Service (RDS).

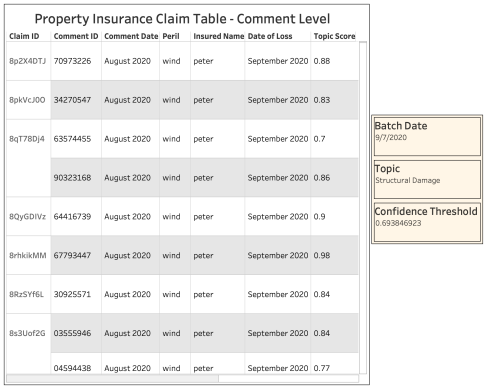
Business users can interactively visualize the results via dashboards (see Figure 4) in real-time. For example, in Figure 4(a), the dashboard provides percentage breakdown of non-relevant claims versus relevant claims to be audited. In Figure 4(b), users are able to view the relevant claims at a detailed level as well as export the contents for further investigation.

Table 2: Results for Auto Claims Loss Cause

RV topics	No. retrieved candidates	No. human labels	AUC	model $+^{ve}$
Wind Damage	4350	1804	94.3	701
Tire Blowout	1490	768	89.4	264
Water Damage	1453	520	91.3	197



(a) View of claims by topic with respective confidence threshold



(b) Detailed view of claims by topic with respective confidence threshold

Figure 4: Interactive Business Dashboard for Property Claims Audit

Table 3: Results for Property Claims Audit

Property claim topics	No. retrieved candidates	No. human labels	AUC	model $+^{ve}$
Structural Damage	1326	231	81.54	239
Pollution	1000	200	72.85	8
Asbestos	341	67	84.61	71
Smoke Type	224	69	81.25	24
Water Intrusion	798	161	72.85	82

5 Related Work

5.1 Active Learning

Active learning aims to develop label-efficient algorithms by sampling the most representative queries to be labeled by an oracle, which usually is a human annotator. Many sampling strategies have been developed over the past decades (MacKay 1992; McCallum and Nigam 1998; Freund et al. 1997; Bloodgood 2018; Wu et al. 2017). The most effective and commonly used pool-based active learning is probably uncertainty sampling (Zhu et al. 2008; Ramirez-Loaiza et al. 2017; Culotta and McCallum 2005). Recently developed deep active learning also research on how to adapt new model architectures to uncertainty sampling (Yoo and Kweon 2019; Gal, Islam, and Ghahramani 2017; Siddhant and Lipton 2018). Although deep models can out-perform classic uncertainty sampling, they are usually computationally inefficient.

5.2 Incremental Active Learning

A deployed active learning system needs to retrain and respond in real time for users to keep focused and efficient. Therefore we based our LSSVMS formulation on least square SVM in (Fung and Mangasarian 2001) and (Suykens and Vandewalle 1999a; 1999b) which produces the solutions by solving a set of linear equations of solving a more costly constrained quadratic programming. Under the formulation of LSSVMs, (Deng 2011) proposes a generalized Sherman–Morrison–Woodbury formula which can be represented in Moore–Penrose inverse and the generalized Drazin inverse forms. On the other hand, more research (Zhou et al. 2015; Brand 2003) proposed to incrementally update Singular Value Decomposition (SVD) based on incoming data. In this paper, we explored using SVD to factorize our least-square solutions to allow for the incremental update of the weights as well to optimally tune the regularization parameter in every step of the updates.

6 Conclusion and Future Work

We have designed, implemented and deployed a system that solves a problem relevant to the insurance domain, although the system could be use in in any other domains where concepts need to be asserted or classified from an unstructured text source. Our system optimally incorporates feedback form human labeler following an active learning strategy. The system is designed such that the human feedback is incorporated into model training in an instantaneous way so it seamlessly enhances the user experience. The machine learning methodology behind this implementation is simple yet elegant and relies on classical results form the least-squares literature. In particular, numerical empirical evidence has shown us that the greedy adaptive regularization approach is of great value to the system and the reason it is possible to use it is a consequence of relying on simple linear models rather than state-of-the-art more complex architectures where performing cross validation would be prohibitive because of both the time constraints and the small size of the available training data characteristic in active learning scenarios. Infrastructure for large-scale deployment of our system is also explained in detail and parts of the code will be shared for reproducibility purposes. As future work we want to explore other more ambitious schemes that take advantage of the speed and simplicity of our approach. One interesting approach would be applications in crowdsourcing platforms where calculations could be performed locally (light weighted edge computing) and then sync sporadically with a master node that would act as a global active learner.

References

Bloodgood, M. 2018. Support vector machine active learning algorithms with query-by-committee versus closest-to-hyperplane selection. *CoRR* abs/1801.07875.

Brand, M. 2003. Fast online svd revisions for lightweight recommender systems. In *Proceedings of the 2003 SIAM international conference on data mining*, 37–46. SIAM.

Brand, M. 2006. Fast low-rank modifications of the thin singular value decomposition. *Linear algebra and its applications* 415(1):20–30.

Cer, D.; Yang, Y.; yi Kong, S.; Hua, N.; Limtiaco, N.; John, R. S.; Constant, N.; Guajardo-Cespedes, M.; Yuan, S.; Tar, C.; Sung, Y.-H.; Strope, B.; and Kurzweil, R. 2018. Universal sentence encoder.

Culotta, A., and McCallum, A. 2005. Reducing labeling effort for structured prediction tasks. In *Proceedings, The Twentieth National Conference on Artificial Intelligence and the Seventeenth Innovative Applications of Artificial Intelligence Conference, July 9-13, 2005, Pittsburgh, Pennsylvania, USA*, 746–751.

Deng, C. Y. 2011. A generalization of the sherman–morrison–woodbury formula. *Applied Mathematics Letters* 24(9):1561 – 1564.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Freund, Y.; Seung, H. S.; Shamir, E.; and Tishby, N. 1997. Selective sampling using the query by committee algorithm. *Machine Learning* 28(2-3):133–168.

Fung, G., and Mangasarian, O. L. 2001. Proximal support vector machine classifiers. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '01, 77–86. New York, NY, USA: Association for Computing Machinery.

Gal, Y.; Islam, R.; and Ghahramani, Z. 2017. Deep bayesian active learning with image data. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, 1183–1192.

Golub, G. H.; Heath, M.; and Wahba, G. 2007. Generalized cross-validation as a method for choosing a good ridge parameter. In Chan, R. H.; Greif, C.; and O'Leary, D. P., eds., *Milestones in Matrix Computation - Selected Works of Gene H. Golub, with Commentaries*. Oxford University Press. 202–212.

Gormley, C., and Tong, Z. 2015. *Elasticsearch: the definitive guide: a distributed real-time search and analytics engine*. " O'Reilly Media, Inc.".

Joulin, A.; Grave, E.; Bojanowski, P.; and Mikolov, T. 2017. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, 427–431. Association for Computational Linguistics.

Lewis, D. D., and Gale, W. A. 1994. A sequential algorithm for training text classifiers.

MacKay, D. J. C. 1992. Information-based objective functions for active data selection. *Neural Computation* 4(4):590–604.

Martin, G. L., and Corl, K. G. 1986. System response time effects on user productivity. *Behaviour & Information Technology* 5(1):3–13.

McCallum, A., and Nigam, K. 1998. Employing EM and pool-based active learning for text classification. In *Proceedings of the Fifteenth International Conference on Machine Learning (ICML 1998), Madison, Wisconsin, USA, July 24-27, 1998*, 350–358.

Pennington, J.; Socher, R.; and Manning, C. D. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 1532–1543.

Polyak, B. T., and Juditsky, A. B. 1992. Acceleration of stochastic approximation by averaging. *SIAM journal on control and optimization* 30(4):838–855.

Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; and Sutskever, I. 2019. Language models are unsupervised multitask learners. *OpenAI Blog* 1(8):9.

Ramirez-Loaiza, M. E.; Sharma, M.; Kumar, G.; and Bilgic, M. 2017. Active learning: an empirical study of common baselines. *Data Min. Knowl. Discov.* 31(2):287–313.

Settles, B. 2009. Active learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences.

Siddhant, A., and Lipton, Z. C. 2018. Deep bayesian active learning for natural language processing: Results of a large-scale empirical study. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, 2904–2909.

Suykens, J. A. K., and Vandewalle, J. 1999a. Least squares support vector machine classifiers. *Neural Process. Lett.* 9(3):293–300.

Suykens, J. A. K., and Vandewalle, J. 1999b. Multiclass least squares support vector machines. In *International Joint Conference Neural Networks, IJCNN 1999, Washington, DC, USA, July 10-16, 1999*, 900–903. IEEE.

Suykens, J. A.; Lukas, L.; and Vandewalle, J. 2000. Sparse approximation using least squares support vector machines. In *2000 IEEE International Symposium on Circuits and Systems (ISCAS)*, volume 2, 757–760. IEEE.

Thönes, J. 2015. Microservices. *IEEE software* 32(1):116–116.

Wieting, J.; Bansal, M.; Gimpel, K.; and Livescu, K. 2015. Towards universal paraphrastic sentence embeddings. *CoRR* abs/1511.08198.

Wu, J.; Guo, A.; Sheng, V. S.; Zhao, P.; Cui, Z.; and Li, H. 2017. Adaptive low-rank multi-label active learning for image classification. In *Proceedings of the 2017 ACM on Multimedia Conference, MM 2017, Mountain View, CA, USA, October 23-27, 2017*, 1336–1344.

Yoo, D., and Kweon, I. S. 2019. Learning loss for active learning. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, 93–102.

Zaidan, O.; Eisner, J.; and Piatko, C. 2007. Using "annotator rationales" to improve machine learning for text categorization. In *Human language technologies 2007: The conference of the North American chapter of the association for computational linguistics; proceedings of the main conference*, 260–267.

Zhou, X.; He, J.; Huang, G.; and Zhang, Y. 2015. Svd-based incremental approaches for recommender systems. *Journal of Computer and System Sciences* 81(4):717 – 733.

Zhu, J.; Wang, H.; Yao, T.; and Tsou, B. K. 2008. Active learning with sampling by uncertainty and density for word sense disambiguation and text classification. In *COLING 2008, 22nd International Conference on Computational Linguistics, Proceedings of the Conference, 18-22 August 2008, Manchester, UK*, 1137–1144.