Sentinel: In-House Active Learning Platform

Akshay Bhatia
Research Scientist - NLP, Knorex
akshay.bhatia@knorex.com

Vishakha Kadam
Research Scientist - NLP, Knorex
vishakha.kadam@knorex.com

Jin Yiping
Senior Research Scientist, Knorex
jinyiping@knorex.com

Tho Nguyen
Research Scientist - NLP, Knorex
tho.nguyen@knorex.com

Abstract

Modern applications require large datasets to train a classifier to achieve respectable results. But in practice, this incurs a significant cost since humans have to manually label data points. Active Learning(AL) is useful for reducing the amount of supervision needed for performing a task, by having the model select which datapoints should be labelled. We present Sentinel, an AL platform capable of building text classifiers as well as annotating large amounts of data. Our platform lets the user interact with the system and annotate and build text classifiers up to their requirements, thus requiring less human effort and time. More importantly, we implement a novel training strategy combining active learning with tri-training that performs better than *Random*, *Uncertainty*(entropy) and *Expected Gradient Length*(EGL) sampling on 2 benchmark datasets. Through a series of experiments, we evaluate our platform, Sentinel, on various text classification benchmark datasets and a corpus of Real Time Bidding (RTB) requests essential to the task of Brand Safety at Knorex. Our results demonstrate the effectiveness of our framework and provides for a computationally efficient sampling method.

1 Introduction

Sophisticated machine learning models have demonstrated state-of-the-art performance across many different domains, such as vision, audio, and text. However, to train these models to do well on such domains one often needs access to very large amount of labelled data, which can be costly to produce. Moreover, for real world problems, it is extremely costly to obtain such quantities of labelled data due to budget and time constraints. Reducing this annotation cost is of utmost importance for practical applications. Active Learning(AL) is a data-driven technique useful to identify the data points that are the most informative for the model. Active Learning has drawn much attention in the last few decades and been exploited in Natural Language Processing (NLP), data annotation and image classification task such as Speech Recognition(Zhu, 2005), Information extraction(Settles et al, 2008) etc. The main hypothesis in active learning is that if a learning algorithm can choose the data it wants to learn from, it can perform better than traditional methods with substantially less data for training. An active learner may pose queries, usually in the form of unlabelled data instances to be labelled by an oracle (e.g., a human annotator). Recently, there has been a surge of frameworks, systems and commercial services such as LibAct (Yang et al., 2017) and DUALIST(Settles and Zhu, 2012), Prodigy, Amazon Ground Truth with a focus to solve this data annotation problem using active learning. These focus on implementing state of the art active learning strategies for both single label and multi label setting but require technical expertise to get started.

We built Sentinel, a web based annotation platform as a prototype to facilitate building classifiers with few labelled documents meeting production requirements with minimum manual effort and a reasonable cost. The main motivation for our platform is to let people with little or no technical experience, either in machine learning or more generally programming, interact with the system and build classifier of their own interests.

The main contributions of our work are as follows:

- **Interactive model building**: In addition to being a complete framework for building classifiers including data collection/importing tool, pre-processing pipeline, AL engine, annotation tool and evaluation, Sentinel offers a unique querying strategy combining tri-training with active learning that outperforms baseline strategies on 2 benchmark datasets. Additionally, our interactive platform enables users with no previous technical experience to build high quality text classifiers quickly and efficiently.
- **Annotation interface**: Keeping in mind the production use cases for the trained classifier, Sentinel's easy to use document labelling interface provides the annotator distribution of sampled labels and performance of the trained classifier during

annotation and training.

- **Knorex Brand Safety**: We present a detailed case study discussing the use of Sentinel to improve brand safety at Knorex.

The remainder of this report is organized as follows. *Section 2* provides a brief overview of the related work. *Section 3* presents details of Sentinel. *Section 4* describes a case study of using Sentinel for brand safety text classification task. *Section 5* concludes the paper and points to avenues for future work.

2 Related Work

In this section we briefly review the active learning problem formulation. We also provide an overview of existing libraries and available services that leverage Active learning.

2.1 Problem definition

Active learning provides a framework to reducing data annotation cost and still achieve the target results. The generic goal of an active learning system is to provide the best prediction on a task, using the fewer amount of labels as possible. The system has to choose the most relevant instances to label in order to learn accurately. It is usually considered that the model has access to an annotator/oracle, which provides the labels for these instances. In our case, active learning usually aims at tackling a single problem, i.e. one dataset and one task.

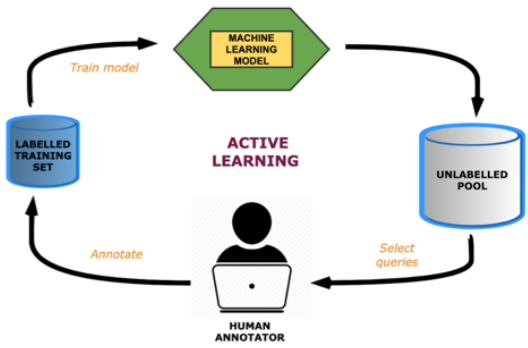

Figure 1: Active Learning Setup

Below we describe a typical AL pipeline:

- We have a dataset D and we would like to train a classifier f on a subset such that it will be efficient in predicting labels of unseen data points from the same distribution
- A data point x_i is represented by a k-dimensional feature vector and $y_i \in Y$ is its label, where there are Y possible labels. For example, we often study binary classification: $Y = \{0, 1\}$
- We consider the pool-based setting where all data points x_i are observed prior to the annotation procedure
- We choose a classifier f that is iteratively trained on some $L_i \subset D$ to map features to labels: $F_t(x_i) = \hat{y}_t$ for example, by predicting the probability $p_t(y_i{=}y|x_i)$
- Given a classifier and a pool of unlabelled data U, the goal of AL is to select which data points should be annotated next in order to learn a classification model as quickly as possible.

In any active learning scenario, a learner may pose queries under different settings. There are generally 3 different settings considered in AL literature

2.1.1 Membership Query Synthesis

In this setting, the learner request labels for samples from unlabelled pool with a input distribution including queries generated by the learner instead of sampling queries from some natural distribution. MQS has been successfully applied to problems in regression learning tasks (Cohn et al., 1996), handwritten digit classification(Lang and Baum, 1992), discovery of metabolic pathways(King et al., 2004).

2.1.2 Pool-Based Sampling

In selective sampling, each unlabelled instance is typically drawn one at a time from the data source, and the learner must decide whether or not to request its label. Here, the decision to discard or keep the instance can be evaluated in several different ways such as informativeness measure more informative samples are queried, region of uncertainty in which learner only queries samples that fall within part of the sample space that is still ambiguous to the learner. Selective sampling has been used in part-of-speech tagging (Dagan and Engelson, 1995), sensor scheduling (Krishnamurthy, 2002), learning ranking functions for information retrieval (Yu, 2005) and word sense disambiguation (Fujii et al., 1998).

2.1.3 Stream-Based Selective Sampling

Most active learning problems are framed with respect to pool based sampling where there is a small set of labeled data L and a large pool of unlabeled data U available and queries are drawn from the pool, which is usually assumed to be closed in a greedy fashion using some information metric evaluated for all instances in the unlabelled pool. Pool based sampling has been widely used for many tasks such as text classification (Lewis and Gale, 1994; McCallum and Nigam, 1998; Tong and Koller, 2000; Hoi et al., 2006a), information extraction (Thompson et al., 1999; Settles and Craven, 2008), image classification and retrieval (Tong

and Chang, 2001; Zhang and Chen, 2002), video classification and retrieval (Yan et al., 2003; Hauptmann et al., 2006), speech recognition (Tur¨ et al., 2005), and cancer diagnosis (Liu, 2004).

2.2 Frameworks, systems and commercial services

Several recent works propose an end-to- end active learning based annotation system. In this section, we discuss various tools and libraries as well as some commercial platforms along with their motivation and limitations within our problem scope which leverage active learning for data annotation and different NLP tasks. These include *LibAct*(Yang et al., 2017) and *DUALIST*(Settles and Zhu, 2012). We also discuss some commercially available services such as Amazon Sagemaker Ground Truth and Prodigy.

2.2.1 LibAct

LibAct(Yang et al., 2016) is a python package with implementations of diverse active learning strategies. It considers a pool-based active learning problem setup with a set of labelled examples, a set of unlabelled examples, a supervised learning model, and an annotator/oracle. In each iteration of active learning, the algorithm queries the oracle to label an unlabelled example for the model. *LibAct* also provides algorithm/parameter selection during active learning by implementing the active-learning-by-learning (ALBL) meta-algorithm. ALBL can smartly validate several different active learning algorithms on the fly, and matches the best of those algorithms in performance, facilitating the users in terms of automatic algorithm/parameter selection.

But unlike Sentinel, *LibAct*, however, does not have a rich UI interface to annotate documents and requires a prior understanding of its API interfaces and python classes which could be a setback for users to get started with it.

2.2.2 DUALIST

DUALIST(Settles et al., 2012) is an interactive machine learning system for quickly building classifiers for text processing tasks. It does so by asking ”questions” of a human ”teacher” in the form of both data instances (e.g., text documents) and features (e.g., words or phrases). It uses active learning and semi-supervised learning to build text-based classifiers at interactive speed.

DUALIST is limited to only supporting few hundred thousand instances during its AL pipeline which limits its use for practical use case in real world applications such as contextual targeting and brand safety.

Figure 2: A web demo of DUALIST platform

2.2.3 Amazon Sagemaker Ground Truth

Ground Truth(GT) is a feature with *Amazon Sagemaker*, a fully managed machine learning service. GT provides users with many features such as:

- **Automatic Labeling** : This feature uses machine learning to decide which data needs to be labelled by humans.
- **Workflow** : Users have the option of using ready made workflows or creating custom ones based on their needs.
- **Workforce**: This involves either using *Amazon Mechanical Turk*(AMT), a pool of 500k independent contractors, or a private workforce of employees or some vendor companies.

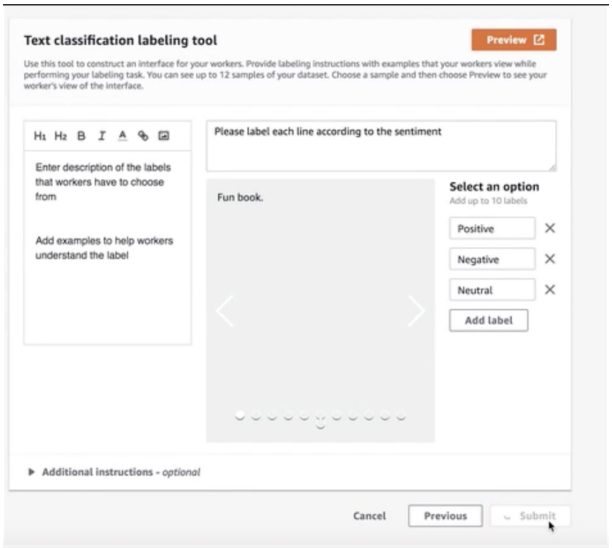

Figure 3: GT text classification annotation view

Although it is backed by a huge *AWS* eco-system with vast support, the main limitation of GT is that it initially requires a few thousand labelled instances before any of its features such as automatic labelling or

training the model could be fully utilized, thus making it extremely inefficient to to cover any real world uses cases such as brand safety or fraud detection that generally require large amount of human effort and time to annotate data.

2.2.4 Prodigy

Prodigy is a machine annotation tool to create end-to-end prototypes for training and evaluating text classification, named entity recognition, image classification and word vector models.

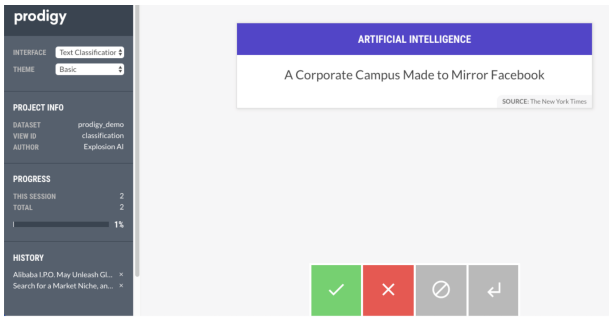

Figure 4: Prodigy annotation web demo

It offers two types of labelling:

- **Manual labelling**: This is used when training corpus is very small
- **Binary labelling**: This lets the user verify whether a label is correct or not after the model predict it

Prodigy is mostly designed as a developer tool in the form of APIs and config calls requiring some technical expertise of the same which could be a barrier to entry for users with no or little experience. Also, since prodigy is more focused on labelling sequences of shorter length, it is somewhat impractical for real time bidding(RTB) documents, used in services such as contextual advertising and brand safety, which are often very long sequences.

3 Sentinel

In this section, we discuss the technical details of our AL platform *Sentinel*. We first describe the fundamental design and system architecture in detail. We then explain the different querying strategies for sampling and ML algorithms available in within platform.

Sentinel is offered as a web based user interface implemented using the tornado web framework. We also use the open source active learning framework *modAL* to build a wrapper and implement various sampling strategies. The complete system architecture is depicted in *Figure 5*. Currently, *Sentinel* only supports text classification.

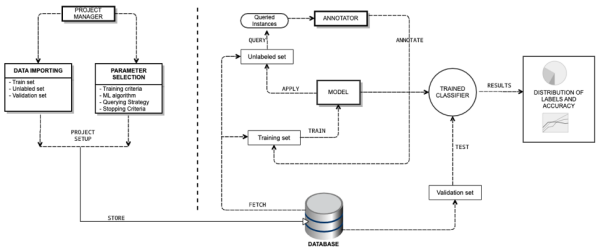

Figure 5: Sentinel system architecture

Our platform comprises of a clean and easy to use UI interface from importing data to annotating documents and training a classifier and can be used by anyone without any specific technical expertise. To make it easier to build classifiers and annotate documents at the same time, we provide two separate interfaces: a project setup interface and a annotation/training interface that constantly interact with third component - a shared database.

3.1 Project Manager

The project setup is an essential step to create and customize annotation projects. This complete workflow of includes the following steps:

- **Creating a project** : Provide a unique dataset name along with the number of categories to be classified into
- **Import datasets** : Import/upload the training, evaluation and unlabelled sets
- **Sampling strategies and type of learner**: Choose additional parameters in the form of ML algorithms and sampling strategies

Below we quickly discuss the above requirements in some detail before moving to the querying strategies including tri-training with active learning.

The platform requires three types of data that the project manager needs to upload. Figure 6a show the data upload interface. *Sentinel* currently supports data in CSV format. First, we require the train set, which will be used to train the machine learning model. The second is the test set, which will be used to test the performance of the system after each annotation step. The last one is the unlabelled set, on which the annotators will work.

Furthermore, our pre-processing pipeline accesses the imported datasets and performs additional pre-processing steps such as cleaning and vectorising the documents and stores them in local database making it readily available for the annotation process.

In addition to the this, the platform also requires some additional parameters (as depicted in figure 6):

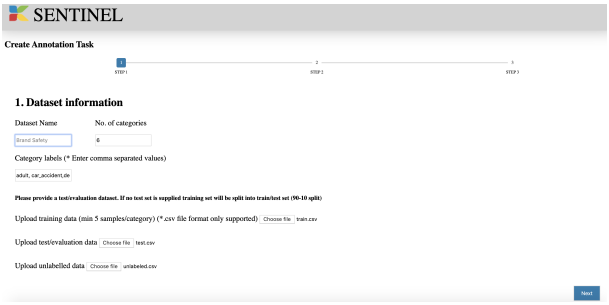

Figure 6: Task setup - I

- **Querying strategy**: Sampling strategy to query samples in the AL loop – *uncertainty*, *margin*, *entropy* or *tri-training with active learning*
- **ML algorithm**: Type of classifier/learner to be used – *SVM*, *MNB* or *CNN*
- **Training criteria**: How to retrain at every iteration – batch or instance
- **Stopping criteria**: When to stop training and/or stop requesting labels – no. of queries or target accuracy

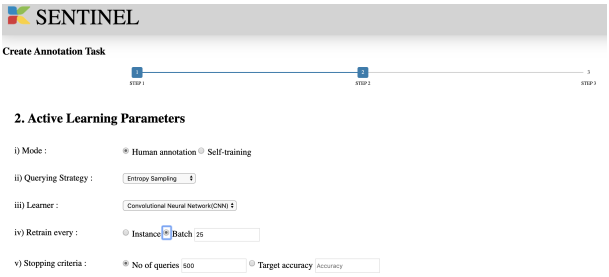

Figure 7: Task setup - II

We aggregate all these parameters with the imported datasets and updates them in our database before moving to the annotation stage.

3.2 Annotator

For annotation and visualization of results, we provide a separate user interface. Since the goal of the platform is to annotate as little documents with minimal effort sufficient enough to achieve a desired performance, we provide the user with a single document and ask them to select the most relevant label from the list provided by the project manager during setup. The resulting annotated document is then first moved to the train set from the unlabeled set in our database. In case of the training criteria being batch mode, this step will be repeated *batch_size* times i.e. number of samples until the classifier is retrained for that iteration whereas for instance mode we simply retrain the classifier with that sample after adding it to the train set. After each batch or instance is manually annotated, we query a new set of samples(either a batch or instance) according to the querying strategy selected and repeat the process again until the stopping criteria.

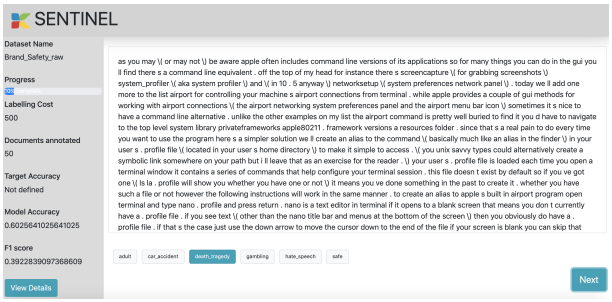

Figure 8: Sentinel annotator interface

The annotation ends when stopping criteria are met. The stopping criteria can either be : a) a predefined number of querying samples, or b) the result on the validation set reaches a target accuracy.

The end goal for our platform is to build production ready classifiers to be used throughout *Knorex*. With this in mind, *Sentinel's* workflow is designed in such a way that an oracle/annotator is able to evaluate the performance of the classifier at each iteration of labelling during the annotation process in terms of accuracy over a fixed validation set and distribution of queried labels.

3.3 Active Learning

Sentinel comprises of an active learning setup that is specifically designed to solve the problem of reducing the annotation load from two viewpoints, by answering the questions "What to annotate?" and "How to annotate?". The question "What to annotate?" becomes particularly compelling if we can select data to be annotated in an iterative and adaptive way. On the other hand, "How to annotate?" deals with large cost savings which can be achieved by labelling each datapoint more efficiently. This can be done with intelligent interfaces that interact with a human annotator.

We now discuss the types of sampling strategies available in Sentinel.

- **Least confidence sampling**: *LC*(Culotta and McCallum, 2005) simply queries the instance whose posterior probability of being positive is nearest 0.5. For problems with three or more class labels, it queries the instance whose prediction is the least confident.

$$\phi_{LC}(x) = 1 - P_\theta\left(y^* \mid x\right) \quad (1)$$

- **Margin sampling**: *Margin sampling*(Scheffer et al., 2001) aims to correct for a shortcoming in least confident strategy, by incorporating the posterior

of the second most likely label(equation 2). Intuitively, instances with large margins are easy, since the classifier has little doubt in differentiating between the two most likely class labels.

$$\phi_M(x) = P_\theta\left(y_1^* \mid x\right) - P_\theta\left(y_2^* \mid x\right) \quad (2)$$

- **Entropy sampling**: *Entropy sampling*(Mann and McCallum, 2007) is an uncertainty measure which uses information-theoretic measure that represents the amount of information needed to "encode" a distribution(equation 3). It is proportional to the average number of guesses needed to find out the true class.

$$\phi_{ENT}(x) = -\sum_y P_\theta(y \mid x) \log_2 P_\theta(y \mid x) \quad (3)$$

- **Tri-training with active learning(Ours)**: *Tri-training*(Chen et al., 2018) is a multi-view bootstrap training method which leverages the agreement(or disagreement) of three independently trained models to reduce the bias of predictions on unlabelled data. The main requirement for tri-training is that the initial models are diverse. Here, an unlabelled data point is added to the training set of a model M_i if the other two models M_j and M_k agree on its label. The main algorithm is as follows:

Algorithm 1 Tri-training with AL

```
for iteration = 1, 3, ... do
    S_i = bootstrapSample(L)
    m_i = trainModel(S_i)
repeat
    for iteration = 1, 3, ... do
        L_i = ∅
        for x in U do
            if p_j(x)=p_k(x)(j, k ≠ i) then
                L_i = L_i ∪ (x, p_j(x))
        m_i = trainModel(L ∪ L_i)
until m_i does not change
Apply majority vote over m_i
```

In *Sentinel*, we implement a novel querying strategy combining AL with disagreement based tri-training. For initialisation, we follow Chen et al., 2018 to generate three accurate and diverse modules - M_i, M_j and M_k. Instead of training three networks separately, tri-net is one Deep Neural Network which is composed of a shared module M_s and three different modules M_i, M_j and M_k. Here, M_i, M_j and M_k classify the shared features generated by shared module M_s. In order to get more diversity among three modules, we use different convolution kernel sizes and different depths for M_i, M_j and M_k. Then, at each iteration we randomly sample a subset of documents from the unlabelled set and add it to the training set only if:

$$p(M_i) \neq p(M_j)\ \&\ p(M_i) \neq p(M_k)\ \&\ p(M_j) = p(M_k) \quad (4)$$

$$p(M_j) \neq p(M_i)\ \&\ p(M_j) \neq p(M_k)\ \&\ p(M_i) = p(M_k) \quad (5)$$

$$p(M_k) \neq p(M_i)\ \&\ p(M_k) \neq p(M_j)\ \&\ p(M_i) = p(M_j) \quad (6)$$

where *p(M)* = output prediction of the model *M*

During inference, given an unseen document *x*, we use the average of the posterior probability of the three modules as the posterior probability of our method. The unseen instance *x* is classified with maximum posterior probability shown in equation 4:

$$y = \underset{c \in \{1,2,\ldots,C\}}{\arg\max} \{p\left(M_i\left(M_S(x)\right) = c \mid x\right) + p\left(M_j\left(M_S(x)\right) = c \mid x\right) + p\left(M_k\left(M_S(x)\right) = c \mid x\right)\} \quad (7)$$

In order to use these querying strategies, we also provide a set of ML classifiers as described below.

- **Multinomial Naïve Bayes**: *MNB* classification algorithm belongs to class of the probabilistic algorithms based on applying Bayes' theorem with the "naive" assumption of conditional independence between every pair of a feature. *MNB* explicitly models the word counts and thus adjusts the underlying calculations to deal with it.

- **Support Vector Machine**: *SVMs* is a type of supervised learning algorithm that constructs a hyperplane or set of hyperplanes in a high- or infinite-dimensional space i.e a good separation is achieved by the hyperplane that has the largest distance to the nearest training-data point of any class since in general the larger the margin, the lower the generalization error of the classifier

- **Convolutional Neural Networks**: *CNNs*, belonging to a family of neural network originally applied to tasks in computer vision, utilize layers with convolving filters that are applied to local features. They have been successfully applied to NLP tasks such as semantic parsing (Yih et al., 2014), search query retrieval (Shen et al., 2014), sentence modeling (Kalchbrenner et al., 2014), and other traditional NLP tasks (Collobert et al., 2011).

3.4 Evaluation and visualization

Evaluation and validation are critical to performance of any ML system to be used in production. Our platform provides an easy to understand interface for quick visualisation of performance of the querying strategy and type of classifier adopted on an independent validation set provided by the project manager.

Most AL querying strategies are evaluated on how quickly they can improve the model. This is often measured with a plot of the some metric, such as accuracy and/or F1 score as in our case, with respect to number of instances sampled. To further illustrate this point, *Sentinel* also displays a graph which specifically explains the distribution of each category of the final training set.

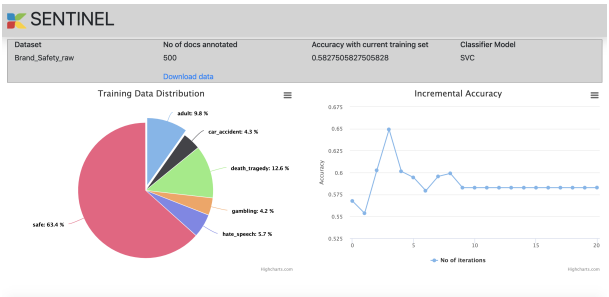

Figure 9: Sentinel results UI

Also, since some users would be only interested in using Sentinel as a data annotation platform, we provide a handy option to download the annotated data(CSV format) to let the user fulfill their requirement. They may then use the annotated corpus as a training set within Sentinel again or as a fixed test/validation set.

4 Experiments

In this section, we describe our results performing different experiments on benchmark datasets as well as for the brand safety task. We specifically evaluate the performance by measuring the precision, recall, f1 score.

4.1 Brand Safety

In this section, we discuss how *Sentinel* helped our team to improve our brand safety offering. Brand safety is one of the most heated topics in online advertising and is of utmost importance at *Knorex*. Real-time bidding opens up a huge volume of inventory for us to display our ads. However, programmatic bidding gives rise to risks of showing our ads to unsafe pages (e.g. adult, violence, hate speech) or pages that will harm the image of the advertiser. Many advertisers inquire about the brand safety measures we take and having a robust brand safety solution is sometimes a necessary condition for clients to sign a deal with us. Brand safety is a challenging task (kind of like anti-virus) because it involves multiple dimensions such as text, malicious links and malware, images, videos, etc. We need to be able to detect and block all of them to ensure the pages where we serve our ads are safe.

4.1.1 Data

The dataset for this task is sampled from actual real time bidding(RTB) requests. The corpus is obtained from extracting webpages, thus obtaining documents for 6 categories namely: *adult*, *car accident*, *death tragedy*, *gambling*, *hate speech*, *safe*. We use only the content of the webpage ignoring the url and title. We compare our results in different settings and provide an analysis below. The test set for all experiments is highly imbalanced with 869 total documents.

4.1.2 Analysis

We frame brand safety as a multi-class text classification problem in which each document is to be classified into one of the n categories where n=6. We perform our experiments in different settings and provide the analysis below. We use the same documents for both *FST random* and *FST ALF* but with different labels. For the final method, *AL Sentinel*, we choose a different training setting. We describe these method in the section below. Unless explicitly mentioned, we use *uncertainty sampling* as the default query strategy and *SVM* as the default classifier/learner and use *TFIDF* features as input to the classifier for all the experiments.

- **Full supervised training with keyword filtering(FST KF)**: For this method, we train the classifier in a fully supervised way on a corpus of random 20k pseudo-labelled documents using keyword matching.

- **Full supervised training with Sentinel and keyword filtering(FST SALF)**: Here, we train the classifier on a corpus of random 20k documents labelled(filtered) using AL. We label these documents using a classifier build using *Sentinel* on a small subset of manually annotated documents.

- **AL Sentinel**: Finally, we train a classifier using our platform Sentinel alone. We initially train on a training set with 30 documents (5 per class). From our unlabelled pool of 10k documents, we iteratively query 500 additional documents in total and manually annotate them during the training process.

4.1.3 Results

Figure 10 shows the performance comparison of the 3 methods in terms of precision, recall, f1-score respectively. We observe that overall scores for many categories improve using our active learning framework.

Methods	Adult	Car accident	Death tragedy	Gambling	Hate speech	Safe	Overall(Macro)
FST KF	0.53/**0.40**/0.46	0.76/**0.84**/0.80	0.51/**0.91**/0.65	0.61/0.54/0.57	0.36/**0.90**/0.52	**0.87**/0.69/0.77	0.61/0.71/0.63
FST SALF	**0.60**/0.39/**0.48**	0.72/0.70/0.80	0.51/0.79/0.63	0.75/**0.65**/**0.68**	0.35/0.79/0.50	0.80/0.73/0.76	0.62/**0.73**/**0.65**
AL sentinel	0.65/0.29/0.40	**0.90**/0.80/**0.85**	**0.67**/0.80/**0.73**	**1.0**/0.29/0.45	**0.76**/0.63/**0.69**	0.80/**0.91**/**0.85**	**0.80**/0.62/0.66

Figure 10: Comparison of Precision/Recall/F1 scores on the RTB dataset

It is clear from observations in second and third row that overall relabelling and filtering using active learning setup helps improve scores for most of the categories. It should also be noted that for the classifier trained using *Sentinel*(third row), increasing the number of instances queried would further improve the current scores but would incur a significant cost in terms of time and effort.

4.2 Benchmark Datasets

The data used for this section comes from 3 different sources: AGNews(A. Gulli, 2005), Reuters R8 and 20Newsgroup(Ken Lang, 1995). For our experiments, we limit the sentence length to 128 tokens. We use the same setup for all baselines and our tri-training with active learning implementation including the models and pre-processing steps as described in following subsections. We briefly describe each dataset below and refer the reader to the source citations for additional details.

- **AGNews)**: We obtained the AG's corpus of news article on the web2. It contains 496,835 categorized news articles from more than 2000 news sources. We choose the 4 largest classes from this corpus to construct our dataset, using only the title and description fields. The number of training samples for each class is 30,000 and testing is 1900.
- **Reuters R8)**: Reuters R8 is a subset of Reuters corpus with 8 classes. There are in total 7674 documents with 5485 train docs and 2189 test docs. The documents in the Reuters-21578 collection appeared on the Reuters newswire in 1987.
- **20Newsgroup**: The 20 Newsgroups data set is a collection of 18846 20,000 newsgroup documents, partitioned (nearly) evenly sorted by date into training(60%) and test(40%) sets is used. The data is organized into 20 different newsgroups, each corresponding to a different topic. The 20 newsgroups collection has become a popular data set for experiments in text applications of machine learning techniques, such as text classification and text clustering

4.2.1 Baselines

We compare our approach, tri-training with active learning with 3 with baselines: entropy sampling, random sampling, EGL querying strategy. To offer fair and comparable comparisons to these baselines, we choose the same learner each for experiment – a CNN with 3 convolutional layers with kernel size of 3,4,5 respectively each with 128 filters and use standard pre-trained word2vec(Mikolov et. al, 2013) embeddings to encode words to 300-dimensional vectors.

- **Random Sampling)**: This strategy is equivalent to standard (or 'passive') learning; here the training data is simply an i.i.d. or more simply a random sample from the unlabelled set U.
- **Entropy Sampling)**: Uncertainty sampling(Lewis and Gale 1994; Tong and Koller 2002; Zhu et al. 2008; Loaiza et al. 2016) is one of the most commonly used querying strategy in which the learner requests labels for instances about which it is least certain. Here, we use entropy (Shannon 2001) as a uncertainty measure.
- **Expected Gradient Length**: Expected gradient length(EGL) strategy aims to select instances expected to result in the greatest change to the current model parameter estimates when their labels are revealed (or provided) (Settles and Craven 2008). The intuition is that one can view the magnitude of the resultant gradient as the value of purchasing a label; if this cost is small, then the label did not provide much new information. For text classification, we use a modification of EGL as described in Active Discriminative Text Representation Learning (Zhang et. al, 2016) where they explicitly select examples that are likely to affect the representation-level parameters (i.e., the word embeddings).

4.2.2 Results

In order to show the effectiveness of our implementation, we show the comparison of accuracy and F1 scores for our approach with the baselines below. The plots show the performance in correlation with the number of instances queried at each iteration.

Experimental results demonstrate that the tri-training with AL variant performs predominantly better than other sampling strategies. Even though for all strategies the final scores are almost similar and even saturate at the final stage, the results show a clear advantage for our method as we note a steep increase in performance during beginning of training.

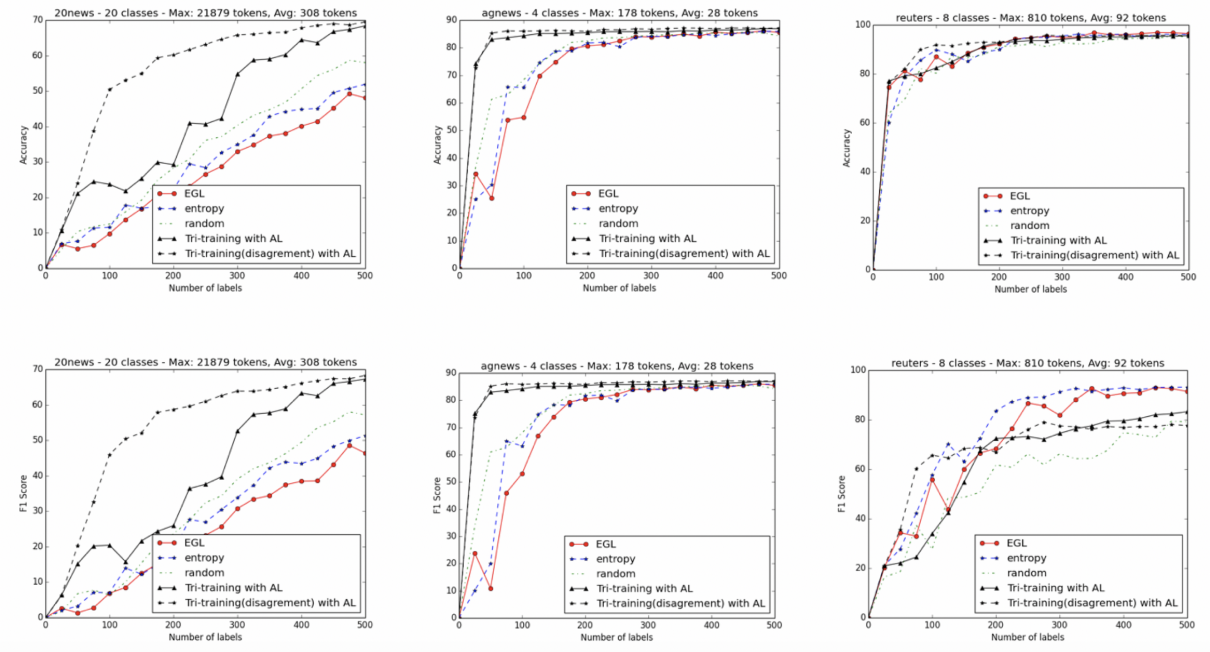

Figure 11: Results on three benchmark datasets. Top row: Accuracy versus number of samples queried. Bottom row: F1 score(macro) vs number of samples queried.

5 Conclusion

We introduce *Sentinel*, an end-to-end web based platform that enables people to build text classifiers using active learning without requiring a large corpus of labelled documents. A key feature of Sentinel is the underlying active learning environment which allows easy and interactive ML model building using a human annotator in the loop.

Future scope of our work includes:

- Extend our work for other NLP tasks
- Include novel querying strategies and other ML algorithms
- A hybrid human-machine annotation platform which uses learning and rule based learning in combination
- Improve the contextual targeting offering at Knorex using Sentinel