ACTIVE LEARNING WITH VISUALIZATION

by

Lulu Huang

Submitted in partial fulfillment of the requirements
for the degree of Master of Computer Science

at

Dalhousie University
Halifax, Nova Scotia
December 2016

To Dalhousie,

a great place,

to my heavenly father,

for His blessing that continues to flow into my life

to my beloved families and husband,

who always encourage me and bring out the best in me,

to my fellowship brothers and sisters,

a group of lovely and sweet spiritual friends

Table of Contents

List of Tables

List of Figures

Abstract

Labeled datasets are limited even though data nowadays are produced with an incredible speed. This affects automatic machine learning methods, especially supervised learning, which requires labels to generate valuable information. However, active learning algorithms help to achieve good analytic models with the least labeling efforts, by querying data points to an oracle, which can provide labels. However, users have no control on the instances to be labeled, and for text data, the annotation interface is usually document only. Both of them decrease user experience, and thus, affect the performance of a learning model. Visualization techniques, particularly interactive ones, help to solve this problem. In this thesis, we study the role of visualization in active learning for text classification with an interactive labelling interface. We compare the results of three experiments to show that visualization accelerates high-performance machine learning model constructions with an active learning algorithm.

List of Abbreviations and Symbols

BOW	Bag of words
BOA	Bag of aspects
QBC	Query by Committee
SVM	Support Vector Machine
T-SNE	t-distributed Stochastic Neighbor Embedding
$\mathcal{L}$	group of labeled data samples
$\mathcal{U}$	pool of unlabeled data samples
$\mathcal{M}$	selected machine learning method

Acknowledgements

First, I want to express my sincerest gratitude to my supervisor, Dr.Stan Matwin, of the Faculty of Computer Science at Dalhousie University. I appreciate his immense knowledge in both academic areas and daily life. He has been supportive since the days I began working as an undergraduate. His continuous patience, humility, and enthusiasm in research, demonstrates what a brilliant and hard-working scientist can accomplish.I could not have imagined having a better advisor and mentor for my study in Dalhousie.

I would also like to thank Dr. Evangelos Milios and great people from his lab. Suggestions from them are valuable for my study.

I would also like to acknowledge Dr. Derek Reilly as the reader of this thesis, and I am gratefully indebted to him for his very valuable comments on this thesis.

My sincere thanks also goes Dr. Rosane Minghim from Universidade de Sao Paulo for her patience and support in overcoming numerous obstacles I have been facing through my research.

There is no way to express how much it meant to have been a member of the Institute for Big Data Analytics. These brilliant friends and colleagues have inspired me over the years.

Chapter 1

Introduction

This chapter explores the motivation of the thesis work, explains the contributions of our study, and goes through the organization of this thesis.

1.1 Motivation and Contribution

"Big data" notion has been mentioned more and more frequently in people's life nowadays, and an increasing number of innovative applications based on this have emerged. As studied in [Hilbert and López, 2011], human history started to enter the "digital age" from 2002, when analog storage and digital storage shared the same amount of global information. Until 2007, 94% of all technological memory was in digital format. Meanwhile, data increments at an incredible speed. As presented in [Turner et al., 2014], if digital data have their own universe, this digital universe grows at an exponential speed, and by 2020 the data people generate annually will reach 44 zettabytes, as many as the star numbers in our physical universe. Figure 1.1 describes the prediction made in [Turner et al., 2014]. The data are obtained from sources such as embedded sensors of software, RFID tags, trackers of moving equipments, and social media applications. Valuable information can be found from the data, but to make the best use of these digitalized data, we need supports from computers. Machine learning in computer science is a suitable approach for this. It helps to make sense of the raw data tsunami for human beings by extracting valuable information using special algorithms. Predictive model and pattern recognition are two popular analyses carried out for huge datasets, and can inspire good business decisions or improvements. This category of machine learning is defined as supervised learning [Russell et al., 2003]. In supervised learning, one necessary element is the labels of obtained data instances. However, in real world, attaining the label datasets is very expensive in terms of time and labor. This is one of the reasons why efficiency of data utilization is not as desired. [Turner et al., 2014] even claims that only 22%

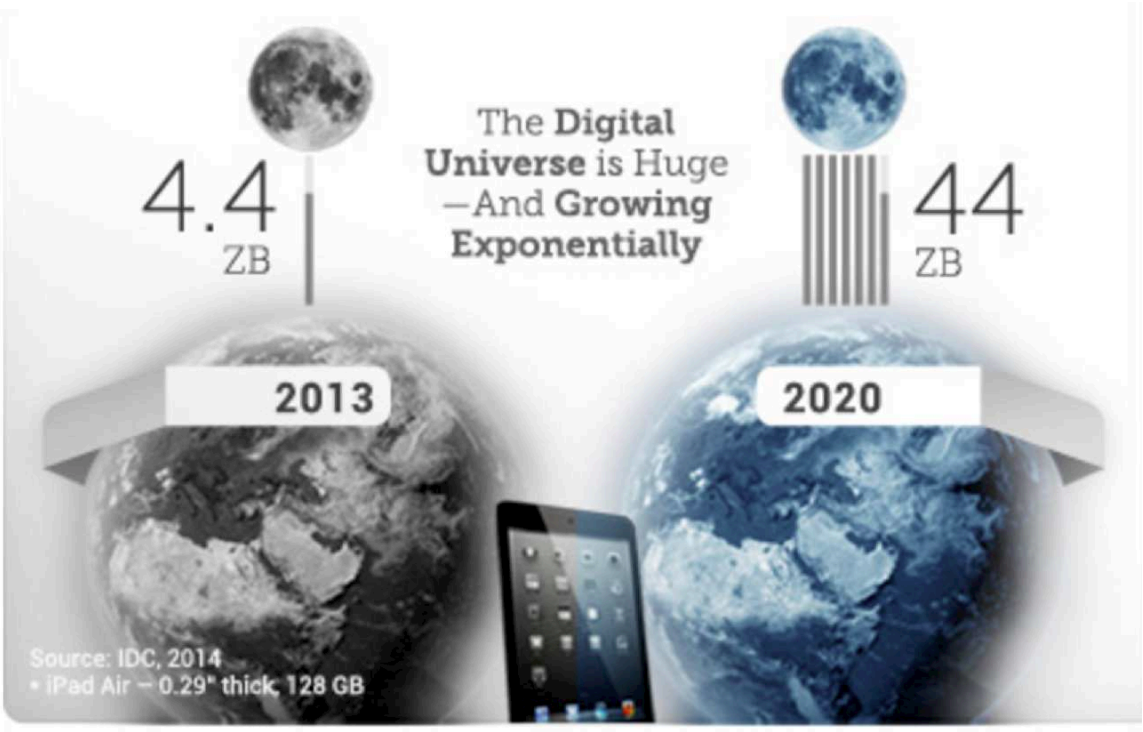


Figure 1.1: Data increasing prediction from IDC [Turner et al., 2014]. The digital universe will grow exponentially from 4.4ZB in 2013 to 44ZB in 2020

of the information out of all digital data, became the candidate for analysis in 2013, but the harsh reality also shows that the actual analyzed data are no more than 5%, and by 2020, useful data will increase to 37% mainly with the contribution from data in embedded systems.

To resolve this problem, active learning has been proposed. As discussed, labels of datasets are not always fully available or cheap to collect, and there are often limited resources for purchasing the labels. So when a small subset of data has labels, we want to make use of these existing labels as much as possible. An active learning algorithm is a tool for evaluating the informativeness of unlabeled data instances based on a small labeled data group [Settles, 2009]. Compared to most supervised learning algorithms, active learning trains a learning model in a less passive manner. Instead of relying on all labeled data instances for model training, as most supervised learning algorithms, active learning takes a few labeled data instances, trains a classifier, iterates the unlabeled data pool, and requests a label of one instance from the unlabeled data pool. Ideally, active learning can achieve a machine learning model with

considerable performance and much lesser training data amount than that required by usual machine learning methods.

An essential step of active learning is annotation. If the annotator of an active learning system is a human being, it is particularly valuable to obtain a user-friendly interface, especially for text data. This demands assistance from visualization techniques. After all, "a picture is worth a thousand words", and comfortable visual interfaces encourage high-quality labels from human annotators. Visualization is another rising technique that supports effective use of the data flood in this digital age. It can convert complex raw data into intuitive facts. For instance, people can easily recognize the distribution of a data table in a scatter plot, while they may need some time to figure out by just reading the numbers in the table. The better communication provided by visualization technique is a relief for human annotators. Today, there are still people who spend hours and days on heavy-duty labeling work, and high demands of annotations remain in the fields of science to daily businesses. The data can be virus streams, feedback from commercial applications, or genome sequences in health studies. Their labels are all meaningful and precious in corresponding fields. The Human and Vertebrate Analysis and Annotation (HAVANA) team [Lagarde et al., 2016], which wants to contribute golden standard labels for gene sequences, is an example.

Though active learning, as well as visualizations, has been proposed for a long while, annotation tools for text data are still limited, and the role of visualization in active learning for text data has not been studied deeply. We thus develop our system to first learn the role of visualization in active learning for text classification problems, and second, to provide an annotation interface for text data. The problem associated with active learning is binary class text classification. To evaluate our interface, we set up three experiments and asked six colleagues to perform all these. Experiment 1 provides visual representation for data, while experiment 3 has only text. Experiment 2 has the same visualization setting as in experiment 1, but with random selection as in the active learning query strategy, where both experiments 1 and 3 have typical active learning query strategies. Accuracy of a classifier is the performance indicator for evaluation purpose. We find that in all experiments, accuracies increase via

human annotators' inputs. Meanwhile, accuracy variances from each user in experiments 1 and 2, where document visual representation is available, are higher than the one from experiment 3, where document visual representation is unavailable. In addition, accuracy variances from each user in experiment 1 are greater than those in experiment 2. This proves that visualization benefits the text classifier's construction with active learning, since greater variances for an incrementing number set stand for more significant improvements in a learning model.

Moreover, for visualization of text documents, we find that the bag-of-words (BOW) model is not always illustrative. Take the force-layout diagram as an example. Each document is represented by a point in the visualization, and the distance between two points is determined by the similarity level between them. Here comes the issue. If we have three documents, as follows:

$d1$	I like this book
$d2$	I don't like this book
$d3$	a waste of money

In the interface, we like to see documents with different sentiments lying away from each other. In terms of these three documents here, the ideal result is that $d2$ and $d3$ are close to each other, and $d1$ is located away from the cluster comprising $d2$ and $d3$. However, if documents are represented by vectors that contain frequency of terms in these three sentences. $d2$ is closer to $d1$, other than $d3$, because $d1$ and $d2$ share the terms "I," "like," "this," "book," while $d3$ does not have any of the terms in $d2$. Considering blue as positive and red as negative, the bag-of-words model will generate a force-layout diagram with mixed red and blue points, which confuses users, and is undesired. We utilized the BOW model, and the bag-of-aspects (BOA) model to investigate if the latter model can produce better segregation for training text data. Concepts and entities embedded in the documents are the aspect notion we mean here. To estimate effectiveness of the BOW model, we align visualizations based on the BOW and BOA models in each available diagram panel. Our test data demonstrates that though these two model-based visualizations fail to keep different class instances apart in a global scope, the BOA-based model can capture local scenarios that BOW-based visualization neglects, while the BOW-based model can show more relationships

to other documents than the BOA model, due to the larger feature numbers in the model.

1.2 Overview

We organize the rest of the thesis in the following order. In chapter 2, background information is introduced. The theoretical foundation of our work includes a brief machine learning description; an active learning algorithm overview, with its scenarios and query strategies; and the visualization techniques. Related work has also been mentioned in this chapter. In chapter 3, we propose our work model by presenting the approach, applied algorithms, evaluation method, and interface design. Chapter 4 puts the theory and proposed work into practice. Dataset is first introduced, followed by technical details before the experiment. Three experiments and user study results are then explained, which form the basis of the later discussion section. Finally, chapter 5 concludes our work and discusses future work.

Chapter 2

Background

2.1 Machine Learning

Machine Learning has come to pervade every aspect of life due to the increasing flood of data. It intends to learn computational models that best fit the existing data, and keep improving their performances with experiments [Mitchell, 1997]. These models can then make high-quality predictions and decisions, or provide solutions to a range of other learning problems. Common machine learning algorithms can be categorized into three groups [Russell et al., 2003, Section VI]. The first group is supervised learning, where datasets contain the category information that observations belong to. The task of supervised learning is usually to identify the class of new unlabeled data instances. Decision tree learning is one of the representative supervised learning algorithms. The predictive tree model built from training dataset map unseen data to their corresponding category. In contrast, in the second group, unsupervised learning, there is no prior knowledge about the categories of data observations for learners. It attempts to conclude hidden patterns within the dataset. For example, clustering algorithms segregate objects into different groups, so that objects are similar within the group, and different between the groups. The last group of machine learning is Reinforcement Learning. Learners of this groups "learn behaviors through trial-and-error interactions with an environment" [Kaelbling et al., 1996]

The significance of Machine Learning is proved by its number of current applications. It is present in almost every domain, from science to business. Bio-informatics for example, which has been a major in many universities, assists to interpret and clarify biology data. Human beings benefit from it by understanding unknown patterns, which could be related to diseases that are currently difficult to handle. Recommendation system is a case close to daily life. If a user purchases items on-line, related products will be displayed. For instance, a universal adapter plug could be shown if a user buys a suitcase from Amazon. Companies earn profit from this since it stimulates

the growth of their trades. Text and speech recognition is another practical utilization of machine learning. Siri in Apple products is a well-known application that provides voice assistance for any operations with natural language processing interfaces. In the field of finance, predictive tools are applied to forecast trends of stocks. Thus, machine learning has come into every aspect of life. It helps to understand big datasets, and dive into poorly understood domains and dynamic situations.

2.2 Active Learning

Active learning is a branch of machine learning that attempts to reach a better performance of learning algorithms with less annotation costs [Settles, 2009]. It assumes that learning algorithms can choose objects they are going to learn, so that similar performances or accuracies can be achieved with less training or labeled data instances. Intuitively, if there are more representative data objects in the training dataset, less learning effort is required.

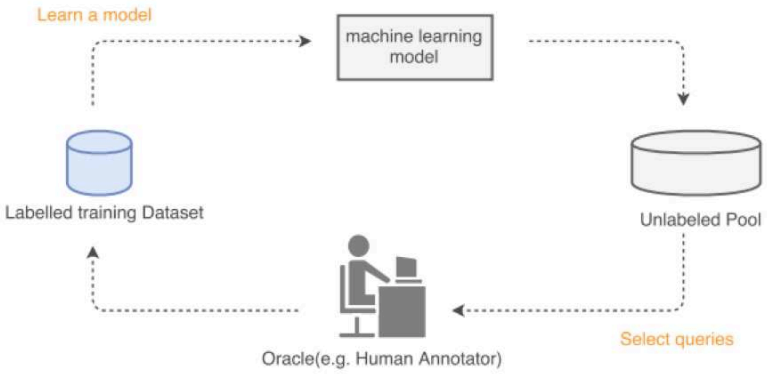


Figure 2.1: The pool-based active learning cycle, where a learning model $\mathcal{M}$ is initially built on the original small set of labeled training dataset $\mathcal{L}$, and then runs active learning strategy in the large unlabeled data pool $\mathcal{U}$ for the human annotator to label. The new labeled instance updates $\mathcal{L}$, following the learning model $\mathcal{M}$, which reruns active learning for a new queried instance.

For example, if a movie recommendation system knows the favorite and least liked movies of a user, it can make better suggestions for the user. Active learning completes the task by querying unlabeled data records for an oracle to label. An oracle can be a human annotator or a powerful machine. Figure 2.1 is an example of the active learning procedure. The machine learning model is constructed from a small set of

labeled training set $\mathcal{L}$, when an active learner selects the most informative data items from unlabeled dataset $\mathcal{U}$ for a human annotator to label. These labeled instances directly join the labeled training set $\mathcal{L}$. After that, the machine learning model is updated with the new labeled training set. This circle continues until some criteria are met. Figure 2.2 is a graph from [Settles, 2009, Chapter 1.1], which visually illustrates the benefits of active learning.

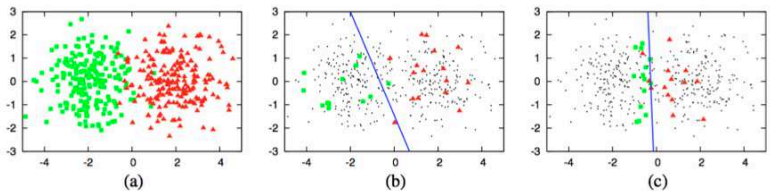


Figure 2.2: Active learning performance example from [Settles, 2009, Chapter 1.1] (a)displays 400 points evenly from two Gaussians; (b) a logistic regression model, indicated by the blue line, constructed on 30 instances from the whole area, achieves 70% accuracy; (c) a regression model, indicated by the blue line, trained on 30 actively queried instances, achieved 90% accuracy.

The dataset of Figure 2.2 is "a toy dataset generated from two Gaussians centered at (-2,0) and (2,0) with standard deviation = 1, each representing a different class distribution [Settles, 2009, Chapter 1.1]. Figure 2.2a presents 400 instances from the sample in two-dimensional space, where 200 of them are class green and 200 class are red; Figure 2.2b displays the blue linear regression boundary built from 30 randomly selected samples, which are evenly distributed in the whole area. This regression model reaches an accuracy of 70%. Meanwhile, Figure 2.2c obtains a linear regression from 30 instances from the active learner, and the accuracy rate is 90%. From Figures 2.2b and 2.2c, the quality of training samples is crucial for a good learning model. Instead of instances across the dataset, trickier instances like the ones around the class boundary are chosen for model construction with active learning. This proves that supervised learning models can gain a better performance with the same amount of training data via active learning algorithms.

2.2.1 Scenarios

There are different scenarios where active learning is demanded. Pool-based active learning in Figure 2.1 is one of the main scenarios, and the other two scenarios in the literature are membership query synthesis, and stream-based selective sampling [Settles, 2009, Chapter 2]. Figure 2.3 shows the differences among these three cases. From top to bottom flows, they are, respectively, membership query, synthesis, stream-based selective sampling, and pool-based sampling. All of them need to start from an input data resource, and query samples for an oracle to label, which is the end part of Figure 2.3. Membership query in the first row selects any unlabeled instances from the input space, including ones the learner generates anew. It does not bias towards querying instances from potential distributions of the dataset, in which way the solution is great for automated pattern discovery, but inadequate for large-scaled data learning.

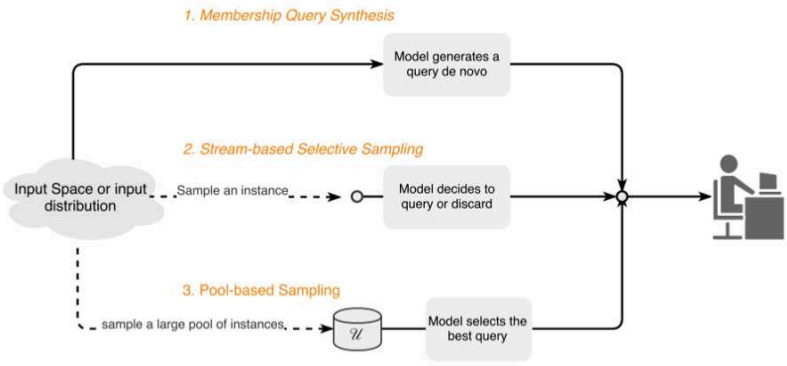


Figure 2.3: Three active learning scenarios – schematic illustration

Stream-based selective sampling, as the second scenario, considers the data input to arrive one at a time. A real-life example is Twitter data. Tweets are retrieved when they occur, so intuitively, they appear one at a time. The active learner in this scenario plays the role of a judge who determines if a data instance is going to be sent to oracles or removed according to the measurements mentioned in the next section. These measurements can be the ones indicating how informative a data sample is, or if an uncategorized dataset is in an explicit region of uncertainty [Cohn et al., 1994] for a machine learning model. Moreover, stream-based selective sampling presumes that it is cheap to get unlabeled data streams, by which real data distribution can be

learned.

Pool-based sampling, in the bottom of Figure 2.3, is the third main active learning scenario. As shown in the figure, compared to the previous two scenarios, pool-based sampling owns a unique unlabeled data pool $\mathcal{U}$ in the flow, which contains all unlabeled data and a small group of labeled data. This brings out the most important hypothesis of pool-based sampling that data collection is performed in the beginning, which contains a small set of labeled data $\mathcal{L}$ and a large group of unlabeled data $\mathcal{U}$, so no unobserved data samples will appear in the learning procedure, just like the example in Figure 2.2. **Algorithm 1** below is the pseudo-code of a generic pool-based sampling. The active learner in this scenario draws a sample $\boldsymbol{d}$ from the unlabeled

Algorithm 1: Pool-based sampling algorithm

Input : $\mathcal{U}$ is a pool of unlabeled data samples; $\mathcal{L}$ is a small group of labeled data samples; $\mathcal{M}$ is a selected machine learning method; $\mathcal{S}$ is an active learning query strategy; $\mathcal{C}$ is a stop criterion

Output: A machine learning model or a labeled dataset

```
while C is not satisfied do
    M = train(L)
    Query the most informative instance d using model M with strategy S
    Obtain label of d from an oracle
    Update L by adding d together with its label, and U by eliminating d
end
```

data pool $\mathcal{U}$, depending on the applied evaluation measurement $\mathcal{S}$. When the label is provided by oracle, the instance will be removed from $\mathcal{U}$ data pool, and added to the labeled data pool $\mathcal{L}$. This process continues until certain stop criteria $\mathcal{C}$ are met. Compared to stream-based selective sampling, pool-based Sampling produces a priority queue for all unlabeled data, and then chooses the most appropriate instance for oracles. Since the input data are available from the beginning, which is more realistic in real-life situations, pool-based Sampling is popular in practical applications. Our work is applying pool-based sampling active learning, as our text datasets are available, and no changes will happen to the dataset.

2.2.2 Query Strategies

No matter which active learning scenario, there is a need for criteria to evaluate how informative an unlabeled data point is. These criteria are specified as query strategies in the literature. Uncertainty sampling, query-by-committee (QBC), and variance reduction [Settles, 2009, Chapter 3] are three strategies utilized in our work. Uncertainty sampling is a well-known strategy that selects instances about which a machine learning model is uncertain, especially a probabilistic model. Intuitively, uncertainty sampling sends instances that lie close to a classification decision boundary. So in a two-class classification problem, points around the boundary that obtain a probability close or equal to 0.5 for each class are the group a machine learning model has difficulties to make a decision, thus are the ones queried. As for multi-class problems, uncertainty sampling gets more general measurements, namely **least confident**, **margin**, and **entropy**.

Least confident measurement is the basic strategy that queries the instances with the least prediction confidence of a learning model. It can be presented by the following equation:

$$x_{target} = \arg\max_{x} \ 1 - P_{\mathcal{M}}(\hat{y}|x)$$

where the queried data instance x_{target} owns an estimated class label $\hat{y} = \arg\max_y \ P_{\mathcal{M}}(y|x)$ under the learning model $\mathcal{M}$ [Settles, 2009]. Margin measurement focuses on the two most possible class labels of a data instance. If instance $\boldsymbol{d}$ holds y_1 and y_2 as the first and second probable class labels,

$$P_{intermediate} = P_{\mathcal{M}}(\hat{y_1}|x) - P_{\mathcal{M}}(\hat{y_2}|x)$$

is about to be calculated. Margin measurement tries to retrieve instances with small $P_{intermediate}$, since a large $P_{intermediate}$ denotes light work for a learning model to decide which class an instance belongs to, whereas a small $P_{intermediate}$ declares a difficult situation for the learner to distinguish. Margin measurement was put forward because the least confident measurement only focuses on the most probable class label, while margin measurement takes the second most possible label into consideration. However, both of them neglect the distribution of other labels, where entropy measurement can take care. Entropy [Shannon, 1948] is a measurement of uncertainty in information theory. If H stands for entropy, it can be computed by the following

formula: $H = -\sum_i p_i \log_2(p_i)$, where p_i is the probability of the current instance being class i. So higher probability of one event p_i leads to lower entropy. In active learning, points with high entropy are the targets, since it claims the ambiguous situation a classifier meets. With labels of these instances, learners can discriminate data instances better.

QBC is another query strategy in active learning that queries instances by a voting system. It keeps a group of committee models $C = \mathcal{M}_1, \mathcal{M}_2, ..., \mathcal{M}_n$ that have right to vote on the label of a data instance. Targets are the ones that maintain the most disagreements among the committee models. Variance reduction, as another methods in our study, tries to query instances that minimizes the output variance after the application of the instance.

2.3 Visualization

Visualization has become a common tool for users to understand huge datasets. It is a way of information exchange using graphical representation [Ward et al., 2010]. Human beings have the ability to process images faster than words, since words understanding is bound with reading speed and image analysis is executed in the human perceptual system [Ward et al., 2010]. Intuitively, people take longer to read a page of explanation about a diagram than studying a diagram. Thus, an effective visualization is a crucial for a learning process.

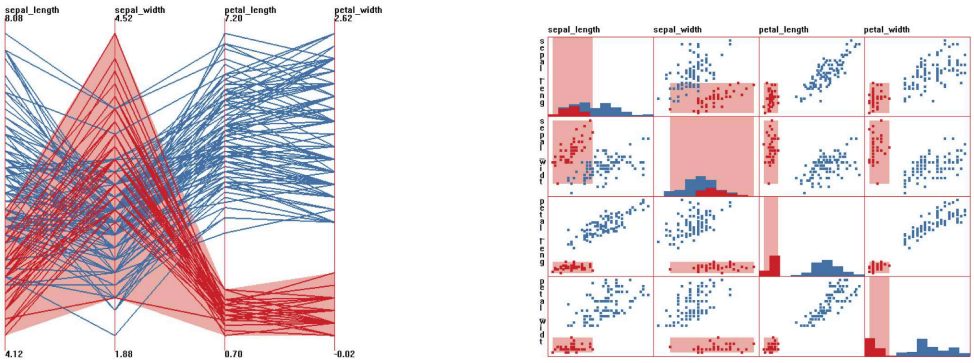


Figure 2.4: Connection operators in a linked brushing example. Selections (red lines) in the left parallel coordinate will update highlighted areas (red boxes) in the right scatter plot matrix.

However, learning purposes vary from situations to situations. It can be the trend

of sale numbers, the difference between groups, the most important factor, and so on. Interactive visualization is one solution, for it adjusts the way and the content of one visualization to a user [Ward et al., 2010, Chapter 10]. For example, filtering is an option for users to find a more specific detail via trimming down data amount on the screen according to factors, dimensions, or other constraints. With filters, users are exposed to less data, or less noisy environment, which alleviates the workload. Connecting is another useful interaction technique. With this operation, users get control of links between data features, so the relation between objects can be realized. Figure 2.4 [Ward et al., 2010, Chapter 10.1.6] is an example of applying connecting interactive techniques. This is a linked brushing example, where data selection can be controlled by users. When a user changes the selection in one view, corresponding linked data in another view are highlighted. In Figure 2.4, parallel coordinates, indicated by red lines, are the selected objects. The linked objects in the right scatter plot matrix are highlighted by red boxes. Selections in any view update the highlighted area in the other view. These two linked graphs enable users to find information about changes. Interaction components are commonly included in well-organized visualization systems such as D3 data-driven documents [Bostock, 2012].
Visualization for unstructured data such as texts, documents, images, and videos, is different from structured ones, which is mainly numerical data. Our study focuses on text data, so numbers of text visualization techniques are introduced here. The vector space model [Salton et al., 1975] is fundamental for many text visualizations. A term vector for a text record contains the weight of a word in the record. A basic weight is the term frequency or the number of times a term appears in the record. Term frequency inverse document frequency (Tf-Idf) is another popular weight, which is calculated by

$$tf - idf = tf(t, d) \times idf(t, D) \tag{2.1}$$

where $tf(t, d)$ stands for frequency of a term t in the doc d, and $idf(t, D)$ represents inverse document frequency, defined as $idf(t, D) = \ln \frac{N}{|d \in D : t \in d|}$. $|d \in D : t \in d|$is the number of documents that contains term t, and N is the number of documents. Tf-Idf leverages weights of the number of times a word appears in documents and frequency of a word in the whole corpus [Manning et al., 2008], which helps to achieve a better performance of models. With the vector space model, an unstructured text file can

be converted into structured data format, and subsequently, data cleaning and model constructions are enabled.

2.4 Related Works

Active learning, which helps the text classification process with the SVM module, has been studied in [Tong and Koller, 2001]. They proved that with active learning, the SVM module can reach good performance with less labeled training data instances instead of large random training data instances.

Among the large volume of work for text visualization, such as Jigsaw [Stasko et al., 2008], we mainly focus on introducing text visualization interfaces for active learning tasks. [Settles, 2011] presented an annotation interface DUALIST, shown in Figure 2.5. Instead of traditional document-only interfaces, both factors (informative terms) and document are presented. The left-hand side of the figure contains the data instance, available label options, and skip button, while the right-hand side contains important features of a certain class. Users can add or remove features as per their understanding. The grey button in the top enables users to submit their results. Once all processes are complete, a final page with model accuracy, lists of data instances, and corresponding label are presented to users. A similar graphic user interface is put forward in [Arora et al., 2009]. The documents, discriminant features, and green patterns, are highlighted in Figure 2.6 to support a human annotator's decision.

Figure 2.5: DUALIST [Settles, 2011] user interface. Users label instances with highlighted feature list in the right.

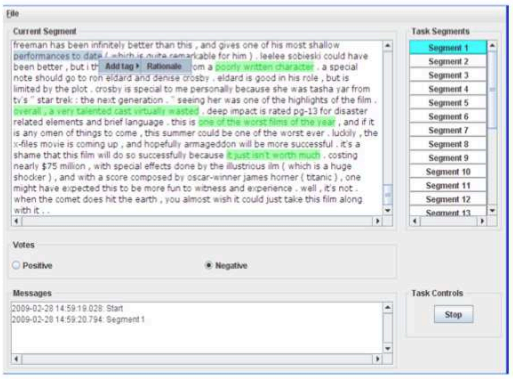


Figure 2.6: GUI in [Arora et al., 2009]. Users label instances with highlighted informative patterns in the document.

Another annotation interface with intuitive clues is shown by [Liao et al., 2015]. Though the data object in the study is video instead of text, the scatter plot of the video data(blue areas in Figure 2.7), has a concept very similar to that of our work. Their work demonstrates that visualization supports and enhances the sample selection of active learning in video annotation.

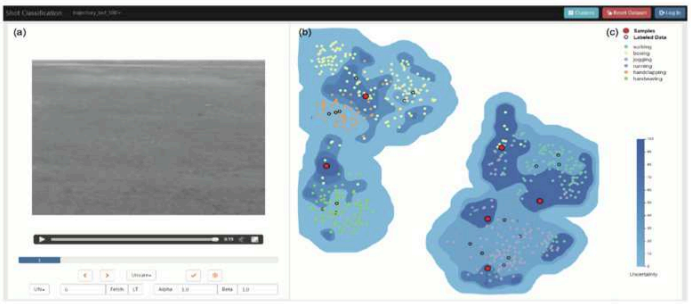


Figure 2.7: Video annotation interface in [Liao et al., 2015]. Blue areas indicate iso-contour-based scatter plots for video data. Each point represents a video. Users can label the video base on the points around.

We are not aware of visualizations of active learning for text classification so far with the clues shown in [Liao et al., 2015]. In this study, we try to integrate active learning for text classification and visualization that contains good indication for annotation via several projections of text data.

Chapter 3

Model

This chapter focuses on the explanation of our active learning work flow, involved machine learning algorithms, and our labeling interface.

3.1 Model Overview

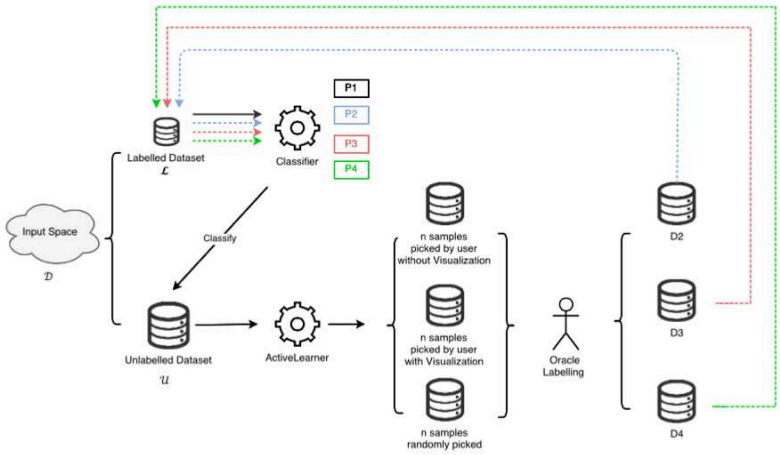


Figure 3.1: Active learning workflow. Starting from the input data pool $\mathcal{D}$, an initial text classifier is trained on labeled dataset $\mathcal{L}$ with performance $P1$. This classifier runs an active learning algorithm on the unlabeled data pool $\mathcal{U}$. A human annotator labels queries from three settings. These three experiments generate different influence to the classifier: $P2, P3$, and $P4$, which are applied to compare the roles of visualization and active learning.

Figure 3.1 exhibits the overall idea of our work. This is a typical pool-based active learning scenario, where input space is available at the beginning and no more new data instances will enter the circle. The input dataset $\mathcal{D}$ consists of a small group of labeled data instances $\mathcal{L}$ for the initial machine learning model construction and a large pool of unlabeled dataset $\mathcal{U}$. As we concentrate on the text classification problem, a classifier is trained with $\mathcal{L}$, the labeled dataset. The performance $P1$ of

this classifier can be calculated subsequently. With this classifier and the unlabeled data pool $\mathcal{U}$, the active leaner can select queries for oracles to label.
To prove our hypothesis, three annotation scenarios are designed, as Figure 3.1:

1. An oracle labels instances suggested by the active learner with only document presentation

2. An oracle labels instances taking the advantage of two-dimensional visual representation of data record

3. An oracle labels unlabeled instances that are randomly selected by the active learner with visual aids

No matter how oracles get the unlabeled instances, new labels are bound to arise. We take $D2$, $D3$, and $D4$ as tags for results of these three experiments respectively. Specifically, $D2$, $D3$, and $D4$ consist of selected queries and corresponding class labels provided by oracles. These new labeled instances are then added to the small labeled data pool $\mathcal{L}$, following updates of both unlabeled data pool $\mathcal{U}$ and the text classifier. Spontaneously, the revised classifier exhibits a new performance. For instance, $P2$ is the performance of the classifier retrained on the updated $\mathcal{L}$ with $D2$. Similarly, $P3$ and $P4$ are the performances of the classifiers with $\mathcal{L}$ updated with $D3$ and $D4$. These performance values are compared to each other to indicate factors that enhance the active learning procedures.
Nevertheless, getting $P2$, $P3$, and $P4$ is not the end point of the work loop. Active learning continues getting queries for oracles, more new labels are produced, and both unlabeled data pool $\mathcal{U}$ and labeled data pool $\mathcal{L}$, as well as the classifier, are kept updated. This goes on until oracle terminates the annotation interface.

3.2 Algorithms and Model Details

This section gives more details of our model, mainly about machine learning algorithms, text representations, and visualization interface.

3.2.1 Algorithms

As illustrated in 3.1, text classification is the machine learning technique involved in the model. Text classification assumes a group of documents with classes or labels. With the help of specific learning algorithms, a classifier or a function is trained to map documents to classes [Manning et al., 2008]. Spam detection and sentiment analysis are two common applications. Classifiers of these two applications are both trained on existing labeled data instances; in other words, it is the supervisor who directs the learning. After that, classifiers make decisions on whether a new arrival email is a spam, or a new comment is positive or negative. Support vector machine (SVM) [Vapnik and Kotz, 1982] and Naive Bayes [Nigam et al., 1998] are two learning algorithms in our work.

SVM as a well-known learning algorithm experiences success in information retrieval problems [Manning et al., 2008]. For a binary class problem, SVM tries to define a decision boundary that best segregates two class objects from each other. These boundaries achieve the maximum margin (blue margin in Figure 3.2) among the training data. Margin in the SVM model stands for distance from the SVM boundary to the closest data instances, which are also called support vectors. In Figure 3.2, the red circles and black triangles on the dashed lines indecate support vectors. With the help of support vectors, there are infinite number of surfaces that can separate the training data, such as a,b, and s in Figure 3.2. However, only s obtains the best margin value. This determines s as the ideal output of the algorithm.

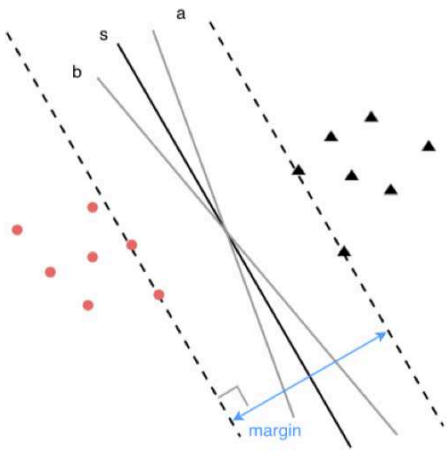


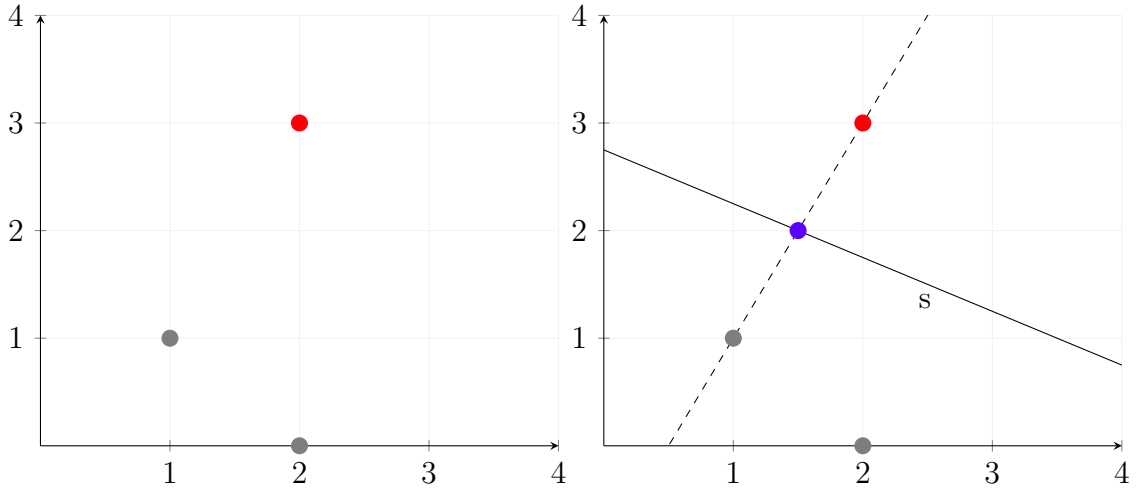
Figure 3.2: SVM on binary class problem. s is the ideal hyperplane of SVM, since it achieves the largest margin among the training data. a and b can separate the training set, but both of them have smaller margin values than the one from s.

If all data points are represented by $\mathcal{X} = \{\boldsymbol{x_1}, ..., \boldsymbol{x_n}\}$ and corresponding label by

$\mathcal{Y} = \{y_1, ..., y_n\}$, where $y_i \in \{0, 1\}$, the target classifier can be written as

$$f(x) = sign(\boldsymbol{w}\boldsymbol{x} + b) \tag{3.1}$$

[Manning et al., 2008]. Points lying on one side of the boundary are classified as class 0, and those lying on the other side as class 1. In a linearly separable case, this boundary is a hyperplane with a normal vector $\boldsymbol{w}$ and an interception b. To achieve the maximum margin value, $\boldsymbol{w}$ should be perpendicular to the hyperplane; otherwise, the hyperplane will be closer to the data point to either class, which leads to a smaller margin value. This $\boldsymbol{w}$ is regarded as the weight vector of the hyperplane [Manning et al., 2008]. In addition, b determines the specific hyperplane among the ones that are perpendicular to the normal vector. Figure 3.3a is an example of



(a) SVM sample training data (b) SVM solution for sample data

Figure 3.3: SVM application with toy data.

applying the SVM classifier with a toy dataset in two dimensions. This dataset consists of three data points, (1,1), (2,0) and (2,3), where (1,1) and (2,0) belong to gray class and (2,3) belongs to red class. This is a linear separable situation, and the solution can be calculated geometrically. As mentioned above, margin is the distance from SVM boundary to the closest data instances, which also means that the margin weight vector is perpendicular to the boundary and parallel to the shortest line connecting data points from two classes. So in the toy example here, the margin weight is parallel to the line passing through (1,1) and (2,3), and the SVM boundary crosses the middle point of (1,1) and (2,3), which is (1.5,2). Thus, we can get the

result line s in Figure 3.3b as: $x + 2y - 5.5 = 0$. To get formula such as formula 3.1, we need more calculations. If (1,1) and (2,0) are class -1 and (2,3) is class 1, given $\boldsymbol{w}$ is (k,2k), we will have two equations:

$$\boldsymbol{w} \cdot (1, 1) + b = -1$$

$$\Rightarrow k + 2k + b = -1$$

and

$$\boldsymbol{w} \cdot (2, 3) + b = 1$$

$$\Rightarrow 2k + 6k + b = 1$$

Accordingly, $k = 0.4$, $\boldsymbol{w} = (0.4, 0.8)$, and $b = -2.2$. The SVM classifier of these three points is $sign(\mathbf{(0.4, 0.8)} \cdot \boldsymbol{x} - 2.2)$, and the margin value is $2/\|\boldsymbol{w}\| = \sqrt{5}$.

The Naive Bayes classifier is another classifier highly popular for classification problems. It is a probabilistic method for deciding the category of a text document, given a list of feature words and corresponding weight information. The algorithm is called "naive" because of its assumption of the Bayes' theorem that all attributes of an instance are independent of each other [McCallum et al., 1998]. The Bayes rule states that the joint probability of events A and B is:

$$\begin{aligned} P(AB) &= P(A|B)P(B) \\ &= P(B|A)P(A) \end{aligned}$$

or the conditional probability $P(A|B)$ (the probability of event A, given event B) is

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{3.2}$$

In Bayesian interpretation [Berger, 2013], $P(A|B)$ in equation3.2 is defined as the posterior probability, the probability of event A given the evidence B; $P(B|A)$ is the likelihood probability, the probability of the evidence B given the parameter A; $P(A)$ is the prior probability, the initial probability of event A; and $P(B)$ is the observation. So the Bayes rule can be converted to:

$$posterior = \frac{likelihood \times prior}{evidence}$$

The Naive Bayes algorithm classifies documents according to the Bayes rule:

$$P(c|B) = \frac{P(B|c)P(c)}{P(B)} \tag{3.3}$$

where c is the class variable; B is a vector consisting of all independent variables, which can also be written as $B = \{b_1, ..., b_n\}$; $P(c)$ is the probability of a document being class c; $P(B)$ is the probability of vector B, which is the observed data; $P(B|c)$ stands for probability of a document with vector B being class c; and $P(c|B)$ represents the probability of being class c given attribute vector B [Manning et al., 2008]. Since $P(B)$ is constant for all documents, the probability equation3.3 can be reformed into

$$P(c|B) \propto P(B|c) * P(c) \tag{3.4}$$

The simple assumption that all attributes are independent of each other, means

$$P(b_j|c, b_1, ..., b_{j-1}, b_{j+1}, ..., b_n) = P(b_j|c)$$

So for all j, the estimated class label c can be calculated by

$$c = \arg\max_c \ P(c) \prod_{j=1}^{n} P(b_j|c) \tag{3.5}$$

Table 3.1 is a small sample data of spam detection. The *class* column indicates if an email is spam or not; *free* and *today* columns show if an email has these two words.

	free	today	class
1	yes	yes	spam
2	no	yes	ham
3	yes	no	ham
4	yes	no	spam

Table 3.1: Naive Bayes application with sample email data

Suppose there is a new coming email containing both "free" and "today." Is this email a spam or ham? According to Naive Bayes, we are trying to find the values of $P(spam|free = yes, today = yes)$ and $P(ham|free = yes, today = yes)$. $P(spam|free = yes, today = yes)$, with the help of Bayes rule, it can be computed via:

$$P(spam|free = yes, today = yes) = \frac{P(free = yes, today = yes|spam)P(spam)}{P(free = yes, today = yes)}$$

To get the value of $P(free = yes, today = yes|spam)$, we can check emails that are labeled as spam, and have both "free" and "today." From Table 3.1, there are

two spams, one of which gets both of them, so the likelihood probability $P(free = yes, today = yes|spam) = 1/2$. Similarly, $P(free = yes, today = yes|ham) = 0$. Prior $P(spam) = 1/2$ since two of the four emails are spam. Though the observation $P(free = yes, today = yes) = 1/4$ is easy to fit, it is not necessary for Naive Bayes, which is proved in equation 3.5. For all variables are available, we are able to determine the new email's label. $P(spam|free = yes, today = yes) = (1/2) * (1/2) = 1/4$ and $P(ham|free = yes, today = yes) = 0 * (1/2) = 0$. The email is certainly a spam since $P(spam|free = yes, today = yes)$ is larger than $P(ham|free = yes, today = yes)$. Naive Bayes is computational fast compared to other complex models, for the strong assumption allows the individual computation for parameters of every attribute [McCallum et al., 1998], which is also illustrated in the example above. This is beneficial to data with a large number of attributes, such as text documents.

To evaluate the performance of algorithms, we apply **accuracy**, the proportion of right predictions among total predictions [Manning et al., 2008]. Confusion matrix, the matrix with the number of correct and incorrect predictions made by the classifier compared to the actual outcomes (target value) in the data, is usually employed to get accuracy value. It consists of $n * n$ values, where n is the number of values of a target class. Table 3.2 is a confusion matrix of a two-class classification problem.

Confusion Matrix		**Actual**	
		positive	negative
Prediction	positive	a	b
	negative	c	d

Table 3.2: Confusion matrix for two-class

In the matrix, rows depict the predicted results from a learning model, while columns display actual distributions. a and d in the table indicate the correct predictions made by the classifier. Specifically, both a positive instances and d positive instances are classified as positive, while b negative instances are classified as positive, which is not expected. $a/(a + b)$ is referred as precision, the proportion of positive instances that are correctly classified, and $c/(a+d)$ is referred as recall, the percentage of actual positive instances that are correctly classified. With the confusion matrix in Table 3.2, $\underline{accuracy = (a + d)/(a + b + c + d)}$. So if we have $a = 60, b = 20, c = 30, d = 50$, the accuracy will be $(60 + 50)/(60 + 20 + 30 + 50) = 0.6875$.

3.2.2 Text Representation

As mentioned in section 2.3, text documents are represented by the vector space model. However, before building a vector space model, the BOW model [Harris, 1954] is applied to extract valuable information, i.e., mainly features from the document. Figure 3.3 presents a transformation process of a toy text dataset. $d1$ stands for "The book arrived fast, thanks!" and $d2$ for "The book arrived in a terrible shape!." $d1$ and $d2$, the group of documents, is usually called *corpus* in the literature [Manning et al., 2008]. The BOW model can then generate a list of words from the corpus: ["the," "book," "arrived," "fast," "in," "a," "terrible," "shape," "thanks"]. With this list, weight information can be computed. If term frequency is the chosen weight, the following two weight lists are the value in the vector space model: [1, 1, 1, 1, 0, 0, 0, 0, 1] and [1, 1, 1, 0, 1, 1, 1, 1, 0]. The vector space model can be illustrated by a matrix containing both features in the bag-of-model and the weight information. This matrix is also named term document matrix (TDM). The matrix on the right end of Figure 3.4 shows the vector space model for the toy dataset.

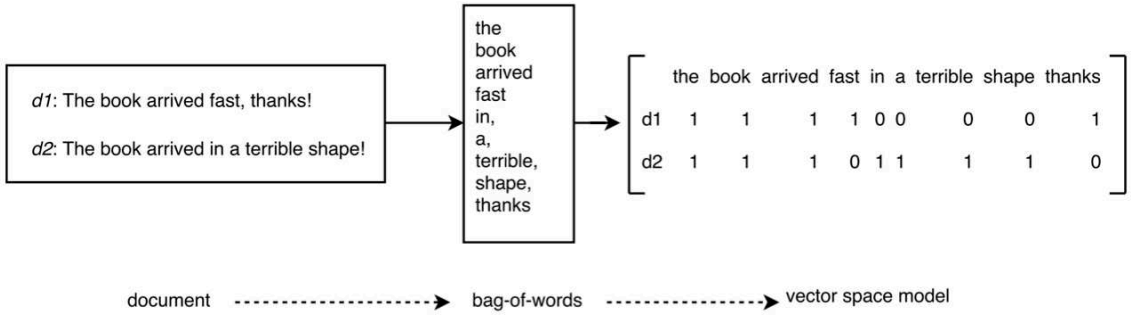


Figure 3.4: Text representation. The transformation from text document of a corpus comprising $d1$ and $d2$ to a vector space model using the BOW model

However, the BOW model only works on single terms from a data corpus, which causes neglect of some important information. For example, bag-of-words result of the corpus in Figure 3.4 will not reveal that "arrived" is always dominated by "the book" or "shape" is in a "terrible" condition. N-gram, a continuous sequence of N characters from a given text [Manning et al., 2008], is a solution for this issue. 2-gram, or bigram of $d1$, is ["the book," "book arrived," "arrived fast," "fast thanks"], which captures the stated missing information. In practice, the vector space model is not limited to single-term features; in contrast, it can own features consisting of any

N-gram. In addition, vector space models are usually refined before machine learning. One modification is that terms that appear too frequently and are meaningless, the so called stopwords, are removed from the matrix. “a,” “the,” “who,” and “at” are some typical examples. Another operation is applying a stemming algorithm [Lovins, 1968], which aims to convert words to their stem. For example, “get” has simple present

	Simple Present	Past Tense	Past Participle	Present Participle
get	gets	got	gotten	getting

Table 3.3: Different tenses of verb “get”

tense “gets,” past tense “got,” past participle tense “gotten,” and present participle tense “getting,” as listed in Table 3.3. If a vector space model has all tenses of a verb, it leads to duplicate situations. Stemming enables less features in the vector space model without any loss of contents.

In our work, we also investigate “aspects” in data instances. We consider aspects as informative features of an object within a document. For example, if a document is“The city has tremendous sightseeing sites, but narrow roads,” the author of this sentence conveys his or her opinion about the “city” via two aspects: “tremendous sightseeing sites” and “narrow roads.” These detailed aspects assist readers to own a thorough understanding. Besides, compared to a list of unigram, bigram or even high grams, aspects of a document make the learning process more precise and less chaotic, just like the examples “tremendous sightseeing sites” and “narrow road” are the most important points. Meanwhile, bag of aspects or concepts is also proved to achieve good performance in text classification problems [Sahlgren and Cöster, 2004]. Intuitively, the vector space model of aspects from text document-only reserves features meet certain criteria. These features can be chosen via statistic measurements such as term frequency and information gain, as well as machine learning techniques like clustering. In our work, we extract noun phrases in a document as our aspects with the help of Part-of-Speech tagger [Marcus et al., 1993], which annotates words’ roles in a speech. For the example sentence above, “city” is assigned with “NN,” which stands for a noun term, either singular or mass; “tremendous” is tagged with “JJ,” which represents adjectives. Similarly, the rest is about to be categorized as verbs, determiners, coordinating conjunction, and other tags introduced in [Marcus et al., 1993]. The

vector space model of aspects is mainly used for the visualization interface presented below.

3.2.3 Visualization Interface Details

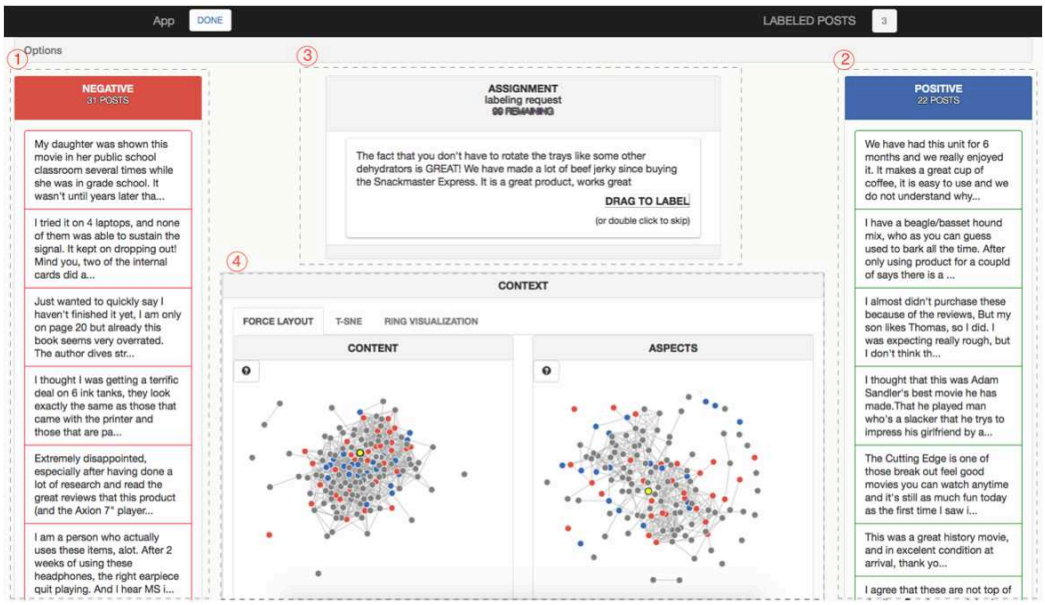
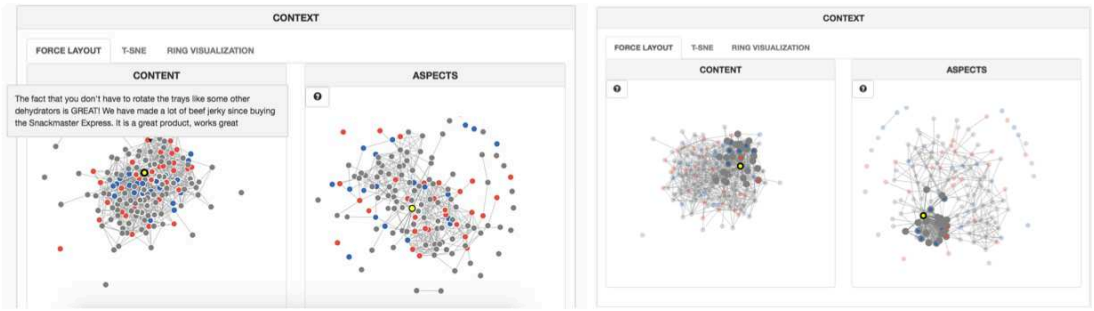


Figure 3.5: Active learning annotation interface. Sections 1 and 2 contain labeled documents from corresponding class; Section 3 is the actively selected query for a human annotator to label; Section 4 is the text visualization areas, where documents are represented by points, colored ones are labeled instances, gray ones are unlabeled, and the yellow point is the actively selected document. Force-layout, t-distributed stochastic neighbor embedding (T-SNE), and chord diagrams are three available visualizations; navigation bar enables users to change active learning strategies and classification models. Right top corner displays the label amount already provided by a user. The "Done" button leads a user to the result page.

In active learning, oracle's labelling is a crucial step. Published annotation interfaces of active learning for text documents usually contain only text, as discussed in chapter 2.4. We propose our own interactive visualization interface with both text documents and document projections. Figure 3.5 exhibits our annotation interface. As shown in the image, there are four main visualization sections. Panels 1 and 2 are containers for objects of two classes. In our case, panel 1 is for negative data instances, and panel 2 is for positive data instances. Panel 3 is the area presenting the document that needs to be labeled. These three panels primarily provide users text only information, while panel 4 contains document projection as a visual aid for users. There

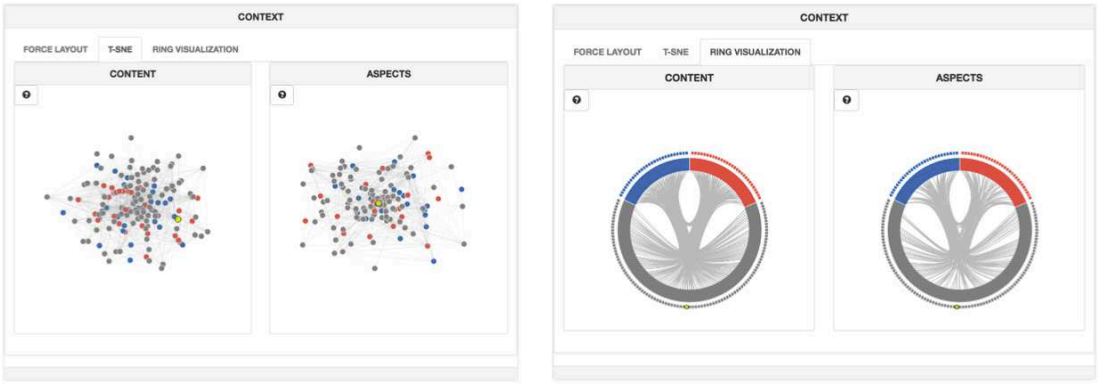
are three main projection methods in panel 4: force-layout [Bostock et al., 2011], T-SNE (Figure 3.7a) [Maaten and Hinton, 2008], and chord diagram (Figure 3.7b) [Bostock et al., 2011]. All of them have a BOW and a BOA model perspective. The text in panel 3 is highlighted in these projections, which visually is a yellow point with black stroke. This makes the target clear among the group of data that are shown in the project. The initial projection after loading the page is force-layout. Figure 3.6 demonstrates the interactions in both force-layout and T-SNE projections.



(a) Mouse over effect in force-layout (b) Selection operation in force-layout

Figure 3.6: Interactions of force layout diagram in annotation interface

When users move their mouse to the highlighted point, as shown in Figure 3.6a, corresponding text will appear on the screen. Besides, if users single-click the point, as shown in Figure 3.6b paths to similar points or documents will show up. Double-click in any other spot helps to clean the paths if users want to go back to the overview. Users can move to T-SNE and the chord diagram panel if they think the force-layout is not helpful. Figure 3.7 presents a typical image a users is about to get when they enter these two panels. T-SNE is a technique to present data with high dimensions in two-dimensional space. The method achieves significantly better performance than other visualization techniques in the field [Maaten and Hinton, 2008]. Text data usually owns a large number of features, so for visually friendly projections, we use T-SNE in our interface. Moreover, the chord diagram is involved in the annotation system as well. Though it is designed for showing dependencies between clusters [Bostock et al., 2011], it is still applicable for explaining the relationships between different documents. To make chord diagram fully functional, users can single-click the target text representation point. Figure 3.8 reveals the effect after clicking operation. Curves to other data points will be marked out if the document has any relationships



(a) T-SNE panel in the interface (b) Chord diagram panel in the interface

Figure 3.7: T-SNE and chord diagram panels in active learning annotation interface

to other documents. Similarly, corresponding text information will appear if users hover the target point in the interface. In all these three projection panels, users are allowed to double-click any grey points to change the labelling object, through which they can be more involved in the system.

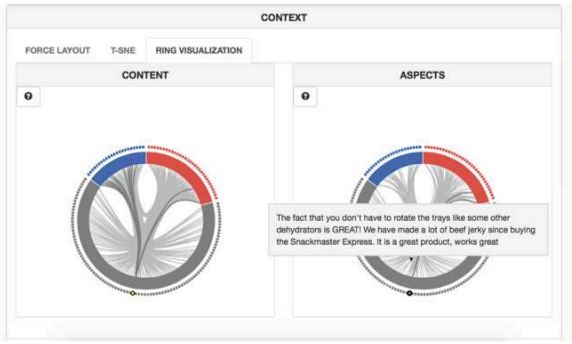


Figure 3.8: Interactions in chord diagram of annotation interface

In addition to all panels mentioned above, there is a hidden panel, called option panel. It unfolds when users click the "Options" section in the interface. Figure 3.9 is what users are exposed to after choosing "Options." This division basically enables users to select methodologies they want for their experiences. In the Query Strategies section, there are uncertainty sampling, QBC, variance reduction, and random sampling choices, which are all introduced in chapter 2. As for the machine learning models, SVM and Naive Bayes explained in the start of this chapter are included. Users can simply click the radio button before the methods to get desired algorithms to function.

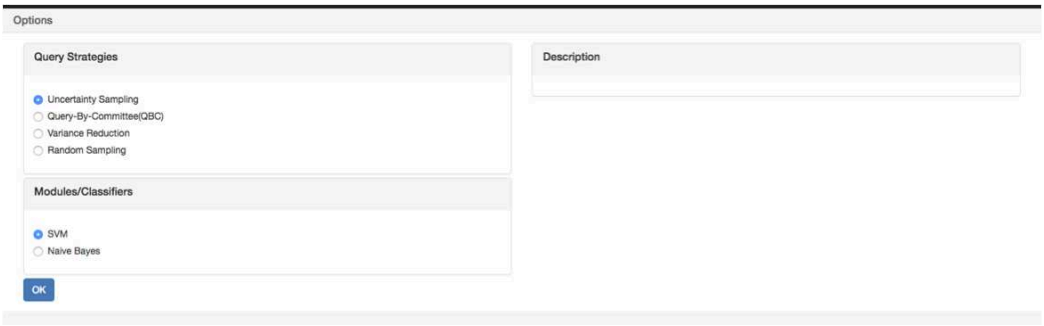


Figure 3.9: Option panel in navigation bar of the annotation interface.

To annotate data instances, users can drag the text in the middle to the class it belongs to, and wait for the next text document. The number in the top right corner indicates the number of instances the oracle has labeled. Whenever users want to terminate the labelling task, they can finish by clicking the "Done" button in the top left corner. This button navigates to the result page. Figure 3.10 is an sample result page. It is a page with a bar chart of accuracy changes during input from users.

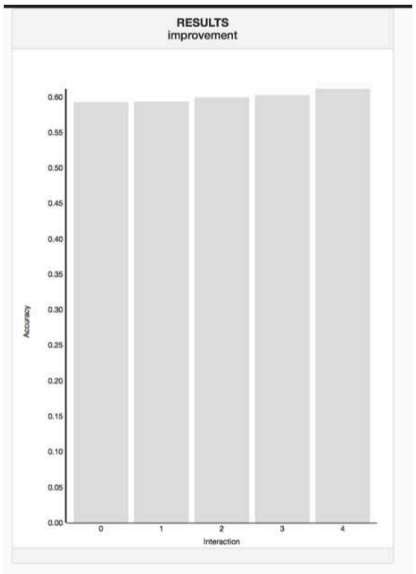


Figure 3.10: Result page of the annotation system. A bar chart with value of accuracy through annotations made by a human annotator. This is a bar chart for accuracy change of five annotations.

Chapter 4

Experiments, Results, and Analysis

This chapter first describes the standard dataset used for our active learning system, then introduces three experiment settings for participants together with some implementation details, and finally discusses the result of each experiment.

4.1 Dataset

Our test dataset is an Amazon Review set shared by [Blitzer et al., 2007]. This multi-domain sentiment dataset consists of product reviews from Amazon.com, which are mainly in four domains: Book, DVD, Electronic, and Kitchen. Each category has its own directory, and reviews are organized into three files: *negative.review*, *positive.review*, and *unlabeled.review*. *negative.review* and *positive.review* are utilized in our work, since we need ground truth to evaluate the performance of a learning model.

tag	value
unique_id	0316059625:we_love_this_book!:m._little
asin	316059625
product_name	The Peace Book:Books: Tod Parr
product_type	books
helpful	9 of 9
rating	5
title	We Love This Book!
date	24-Mar-06
reviewer	Mr.Little
reviewer_location	Brunswick, MD
reviewer_text	Pretty much all the Todd Parr books we've rad are great but the "Peace Book" and the "Family Book" are by far our favorites,. They are very postivie and carry a theme of tolerance. We'd highly recommend this book!

Figure 4.1: An Amazon Review Example

So the ultimate dataset we get is 8000 reviews from the initial downloaded file, and each type has 2000 reviews, where 1000 are positive and 1000 are negative. The review files *negative.review* and *positive.review* initially arrives with a pseudo XML scheme; the details are listed in XML tags. Generally, each review has ***unique_id, asin, product_name, product_type, helpful, rating, title, date, reviewer, reviewer_location, review_text*** attributes. Figure 4.1 shows a positive book review from the dataset. For our experiments, only two attributes are selected, as highlighted by the red box in Figure 4.1: ***asin*** and ***review_text***. As declared by the owner of the dataset that the unique_id attribute is

not unique all the time, we edit the ***asin*** attribute by adding numbers from 1 to 8000 in a string format with a length of 4. The particular number string added to asin depends on the order in which the review is read by the compiler. This modified asin attribute is then able to point to a unique review_text. As for the label value, we simply assign reviews from *negative.review* as -1 and the ones from *positive.review* as 1.

4.2 Experiments and Results

The first step for our experiments is to clean the extracted review data from Section 4.1. Cleaning pipeline is listed in Figure 4.2. Python programming language [Van Rossum et al., 2007] is the main tool to complete the data pre-processing task. As the texts are originally in XML format, the first step of data pre-processing is to escape HTML characters such as "&" and "<." With regular expressions, "&" is translated to "&" and "<" to "<." Then, with the appropriate encoding setting, usually "UTF-8," HTML-character-free reviews are tokenized into tokens. Tokenization is a process for compilers to convert input characters into a meaningful sequence of tokens [Aho et al., 1986], and the tool that accomplishes tokenization is called a tokenizer. A tokenizer follows certain patterns to recognize expressive tokens from input characters, where a token is "a pair consisting of a token name and an optional attribute value" [Aho et al., 1986]. The token name can be a keyword, an operator, an identifier, or any other semantic unit. For example, if an input string is "Active learning is helpful," the output of a word tokenizer will be "Active," "learning," "is," "helpful," "!." Once the tokenization is done, which means each review is divided into sections with semantical value, stopwords

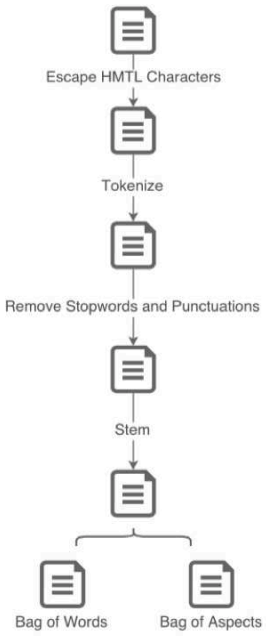


Figure 4.2: Text pre-processing Pipeline. Detailed steps for text data cleaning

and punctuations are about to be removed. As mentioned in the previous chapter, stopwords are usually very common in a document, but can not signify any valuable information; the removal operation creates a less noisy corpus for data analysis. After that, a stemmer will identify the stems of each string. After the stemming, we will build two vector space models: one based on BOW and the other on BOA. This finishes our preparation step of the experiment setup.

The next step for the experiment is to construct the front-end interface and back-end machine learning algorithms. The front-end web interface mentioned in Figure 3.5 is built with HTML, CSS, and JavaScript, where d3.js [Bostock, 2012] is the main JavaScript package utilized for document projections. As for the back-end machine learning algorithms, they are implemented in Python programming language. Since our work is pool-based active learning, it is essential to have an initial labeled data set $\mathcal{L}$. For training purpose, we split the 8000 reviews into two groups: 33% set as the test data and the rest as training data. As a start, the program randomly selects 50 data records to create a classifier. Besides, SVM and uncertainty measurement are set as the default classifier and active learning query strategy, respectively. However, users can always switch the classifier to Naive Bayes, and query strategy to QBC, variance reduction, or random sampling. To manage the data communication between front-end interface and back-end python program, Common Gateway Interface (CGI) [Graham, 1995] is put into use. Figure 4.3 explains how the annotation system runs.

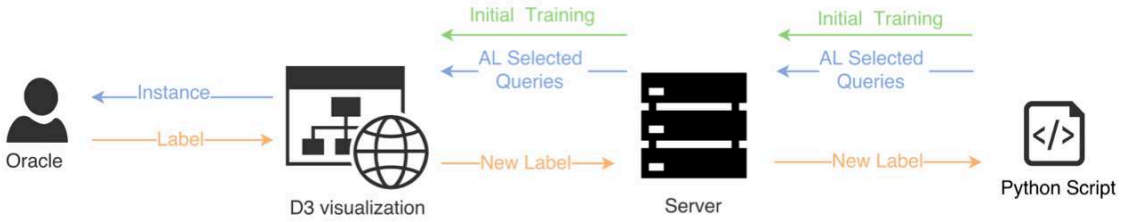


Figure 4.3: System working architecture, which describes system's data communication between front-end and back-end.

There are basically two working scenarios. The first one is when a user initially enters or reloads the web interface. This triggers the script to get 50 randomly labeled instances from the corpus, and trains a classification model, following the active learning query making. The chosen query, 50 labeled data instances, another

100 random unlabeled instances, and their corresponding distance matrices are packed to send to the front end for visualization purpose. A distance matrix retains distances between documents. Table 4.1 is an example distance matrix, where $d1, d2, ...dn$ are documents of a corpus. From the table, we can obtain that documents have no instances to themselves, and distance from d_i to d_j is always the same as that from d_j to d_i. With regard to the value in distance matrix, the standard cosine similarity

	d_1	d_2	...	d_n
d_1	0	0.2	...	0.8
d_2	0.2	0	...	0.3
...	...	...	...	...
d_n	0.8	0.3	...	0

Table 4.1: Distance matrix example for a corpus consists of n documents

is adopted. Cosine similarity is a typical measurement for similarity between two documents d_1 and d_2 [Manning et al., 2008]. If $\vec{d_1}$ and $\vec{d_2}$ are vector representations of d_1 and d_2, the cosine similarity can be written in

$$cos\_sim(\vec{d_1}, \vec{d_2}) = \frac{\vec{d_1} \cdot \vec{d_2}}{\|\vec{d_1}\|\|\vec{d_2}\|} \tag{4.1}$$

The numerator is the inner product result of two vectors, which can be calculated with $\sum_{j=1}^{N} x_j y_j$, where x_j and y_j are feature values of documents d_1 and d_2, and N is the number of features of these two documents. The denominator is the multiplication result of the documents' Euclidean length. For example, if d_1 and d_2 have attributes "free," "call," and "cost," and corresponding values are listed in Table 4.2, their cosine similarity will be (5*0 + 4*1 + 3*0) / ($\sqrt{5^2 + 4^2 + 3^2}$ $\sqrt{0^2 + 1^2 + 0^2}$) = $2\sqrt{2}/5$ ≈ 0.57. With the sent distance matrices, we can draw visualizations in the interface.

	d1	d2
free	5	0
call	4	1
cost	3	0

Table 4.2: Document distance calculation example with sample email data. For documents $d1$ and $d2$, distance is computed with corresponding value of three features "free," "call," and "cost."

Moreover, we define at most 200 points in each panel. One important reason is, as

shown in Figure 3.5, that the projections attain a limited space; meanwhile, it is unrealistic to show all document points in a single panel. However, 200 points avoids visual clutter, and are efficient to reflect valuable information. Therefore, the initial load of the interface notifies the server, and then the server sends signals to activate back-end Python scripts to generate a data group for original model training, get an active query, and calculate distance matrices of BOW model and BOA model. Server-side working flow of the first scenario is colored in green in Figure 4.3. After the Python back-end calculation, the front end will display the text contents and project distance matrices into corresponding force-layout panel, T-SNE panel, and chord diagram panel. The second scenario occurs when oracles provide labels for active queries. Once a user starts annotation by dragging an active query to the class it belongs to, the label message is delivered to the server, and then the server will be alerted to wake back-end Python scripts up for classifier retraining, for labeled and unlabeled data pool updates, and for a new active query. This working pipeline is colored in orange in Figure 4.3. To respond to the request, Python scripts will perform the calculations in the first setting again, and reply with the new active sample, and updated distance matrices. When the front-end interface receives the latest information, it makes a refreshment to itself since the new sample changes the distribution of the whole data group, which leads to different distance matrices. This reply pipeline is the blue working line in Figure 4.3.

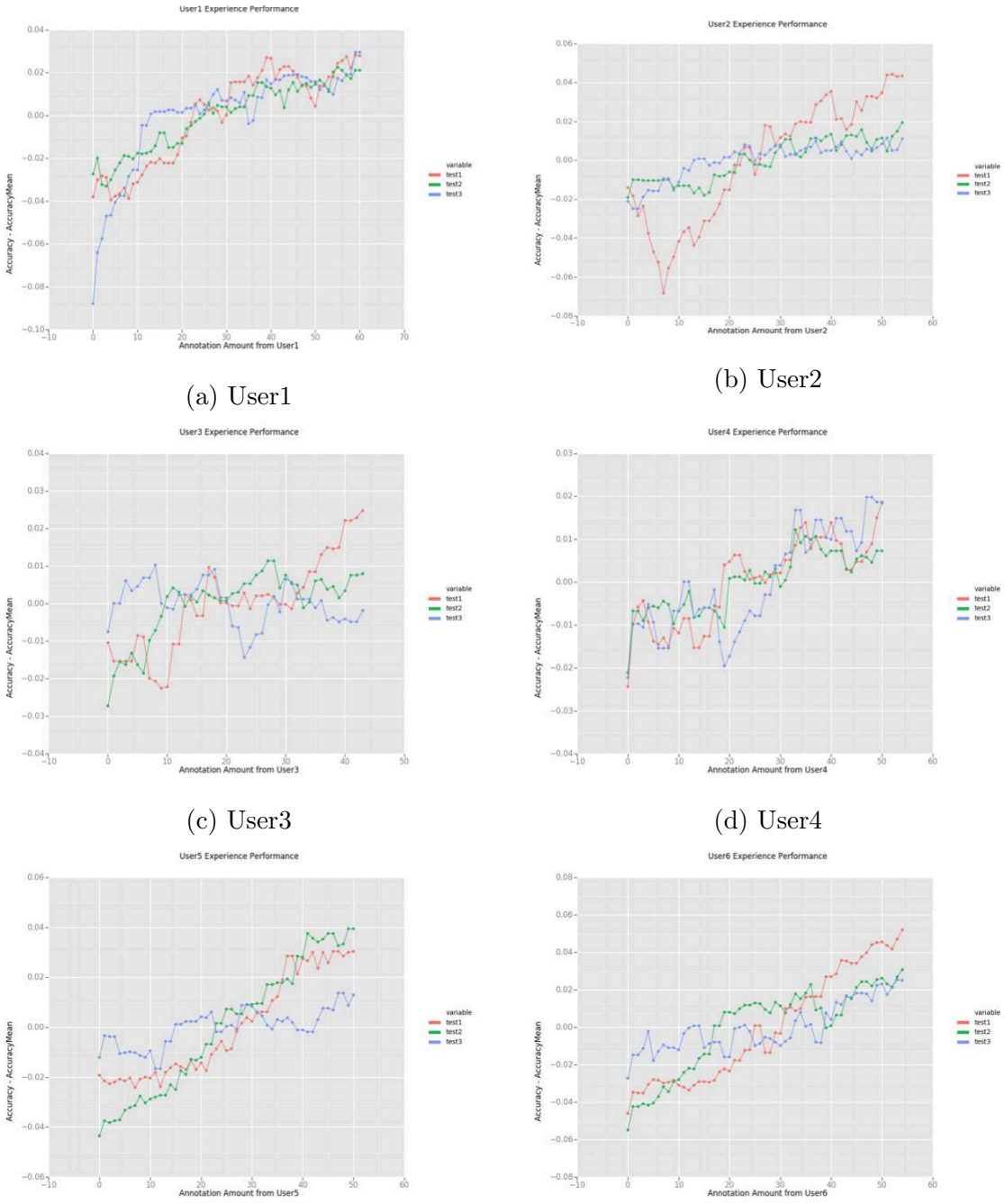
With cleaned dataset and working system, experiments will be carried out to prove our hypothesis that visualization can accelerate good classifier construction. As introduced in chapter3.1, there are three annotation scenarios.

1. Log into the main interface, and then label 20–50 instances.

2. Log into the main interface, in the Option panel, choose “Random Sampling” as the query strategy, and then label 20–50 instances.

3. Log into the interface that has no projections, and then label 20–50 instances

Six colleagues from the Big Data Analytics Institute participated in the user study. They were instructed with this user study purpose, interface functions, three experiment settings, and what they should do and submit. For experiments 1 and



(a) User1

(b) User2

(c) User3

(d) User4

(e) User5

(f) User6

Figure 4.4: Accuracy change over the experiment of six users.

2, participants were highly recommended to take advantage of the visual aid provided in the interface. For all these three experiments, users were suggested to stick to one classifier, which is beneficial for comparisons. Figure 4.4 above exhibits the accuracy changes during annotations. X-axes in the sub-figures indicates the number of annotations made by users, and Y-axes indicates the value of $Accuracy - Accuracy\_Mean$. Y-axis is set to this value due to the different starting model accuracy. With $Accuracy - Accuracy\_Mean$ values, model performance differences are easier to find out visually. Red lines in the figure are the accuracy change in experiment 1, where documentation projections are available to users; green lines are the accuracy change in experiment 2, where random sampling is taken as the query strategy; blue lines, the last color category, denote accuracy change in experiment 3, the test without visualization. From Figures 4.4b, 4.4c, and 4.4f, which are produced by User2, User3, and User6, respectively, we can find, blue lines have milder fluctuations than green and red lines, while red lines usually obtain the most volatility. This firstly verifies the value of inputs from human annotators, since their efforts improve the performance of the selected classifier. This also proves that through visualization, users can make better decisions, and no matter which active learning query strategy is enrolled, visualization boosts the performance of the classifier. Moreover, comparing red lines to green lines, variation in red lines is usually more considerable, which confirms that the active learning query strategy is more efficient than random selection.

	test1	test2	test3
user1	0.00049	0.00025	0.00056
user2	<u>0.00099</u>	0.00011	0.00008
user3	<u>0.00019</u>	0.00017	0.00003
user4	0.00010	0.00005	0.00014
user5	0.00040	<u>0.00069</u>	0.00005
user6	<u>0.00086</u>	0.00053	0.00016

Table 4.3: Accuracy variance values of each user in the three experiments

Table 4.3 contains the variance in accuracy changes of the three experiments from six users. Likewise, for user2, user3, user5, and user6, test3's variances are the smallest among the three numbers, which is consistent with fluctuation situations in Figure 4.4, since large variance indicates large difference among numbers of one group. Data

in column test1 and test2 also confirms the above statement that active learning query strategies win more improvements for classifiers.

Figure 4.5 is the clustered columns chart of numbers in Table 4.3. This graph better illustrates the values in the table. Though highest value in different groups vary a lot, patterns in these groups are similar to each other.

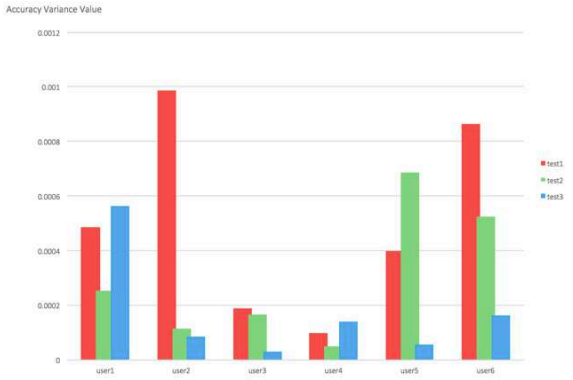


Figure 4.5: Accuracy variance bar chart of each user in the three experiments. Each bar chart group is from one user. Y-axis is the accuracy variance value.

However, data from user1 and user4 are a bit unlike statistics from other participants. User1 claimed that at the beginning of the last experiment, queries were easier to make judgement since interesting topics came into sight, which drew more attention though there was no visual aid during the experiment. Besides, in user1's third experiment, the initial model had a low performance. So with high-quality labels, the classifier can develop faster and becomes more fair. Nevertheless, from Figure 4.4a, it is not difficult to figure out that after around 12 annotations, the blue line tends to grow slower, and the performance improvement is not as significant as the red and green lines. Accordingly, this still supports our assumption that visualizations benefit classifiers that are built with active learning. In terms of user4, a different judgement method from that in the case of other participants was taken during the annotation procedure. Though more long reviews appeared in the annotation group, User4 tried not to spend time on visualizations, but to label fast by scanning the whole paragraph via searching some sentiment indicators such as "good," "terrible," "boring," and "waste" since the target class value of the dataset is "positive" or "negative." So this case is special, and is not appropriate for the evaluation of visualizations. However, the data are still valuable, because they at least, to some extent, confirms the feasibility of the BOW model due to performance improvements of all classifiers from user4's experiments.

Aggregating the whole performance data, we get Figure 4.6, where all data are plotted

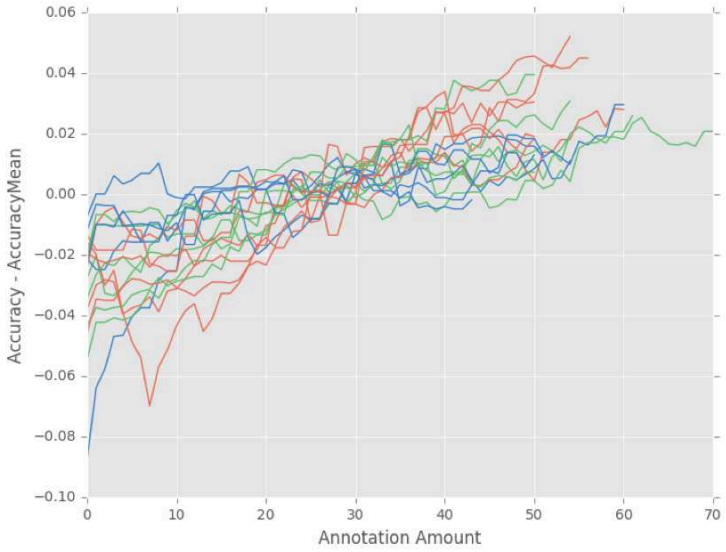


Figure 4.6: Performance from all users. Accuracy of classifiers from all the users in the three experiments; X-axis is the annotation amount contributed by users, and Y-axis is the value of $Accuracy - Accuracy\_Mean$; Red lines are accuracy changes of experiment 1, green lines are accuracy changes of experiment 2, and blue lines are accuracy changes of experiment 3.

according to the experiments. Red lines indicate accuracy changes in experiment1 from all participants, green lines indicate performance changes in experiment2, and blue lines represent dynamic accuracies during experiment3. The values in the graph are accuracy minus mean value of all accuracies in one experiment. This makes all lines comparable in the same coordinate. Again, three group lines in figure 4.6 cater to the conclusions drawn before. Despite the blue line that passes through the point around (0, -0.08), coming from test 3 of user 1, a situation which is explained above, fluctuation level of the red group beats those of the other two groups. The green group is in the middle, while the blue group owns the mildest fluctuations.

4.3 Discussions

In addition to the conclusions summarized from all performance data, we also harvest other valuable information from our user studies. As commented by all participants, the first experiment is the most difficult test, the second experiment provides obvious sentiment reviews, and the third experiment, though offers tricky reviews as the first

test, participants in this experiment get more familiar with the system and all review contexts. The second experiment is relatively easy for participants, because of the random sampling query strategy applied in the Python script. Random sampling does not pay attention to the influence of the new label; instead, it just grabs instances that are still unlabeled. This also explains why the growth of green lines in Figure 4.4 is gentler than that of red lines. The selected samples may only from one class, either the positive or the negative, and may also are ones lying far away from decision boundary, which have no significant contribution to improve the classifier. As to the first comment that test 1 asks for more efforts, there are a couple of reasons. On the one hand, the initial classifier is trained on 50 data samples. It is not possible to retrieve great performance by applying this model to over-2500-record test data. The model could have strong bias on either class. Consequently, the visualizations in force layout, T-SNE, and chord diagram lead users to a hard situation. Figure 4.7 is a screen shot of the case where visualization fails. In the chord diagram, the highlighted curves indicate the close relationship between the active query and positive samples, while the content, "I was disappointed in this video" reveals user's discontent with the product. This is the second query in the experiment, which can be found at the top right corner of the screen. The classifier now is probably more against the negative class. This inaccurate visualization, as a result, pushes participants not to think as highly of the role of the visual aid tool as they expected.However, participants also stated that visualizations turned better through the whole experiment, as evidenced in Figure 4.8. This is the 12th active query in the experiment, shown in Figure 4.8. The re-

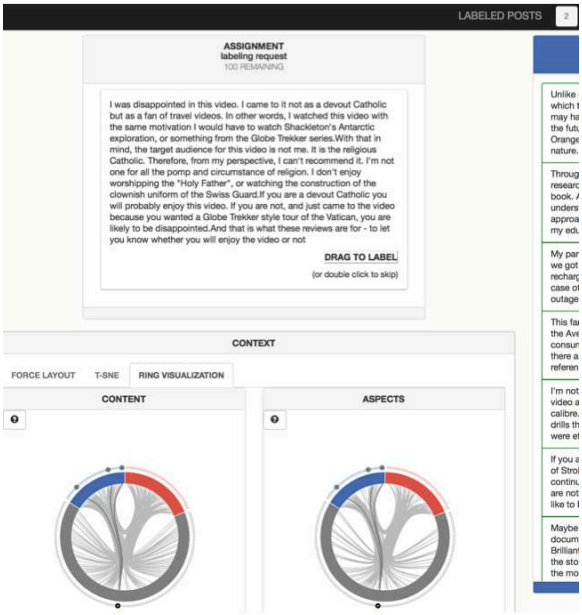
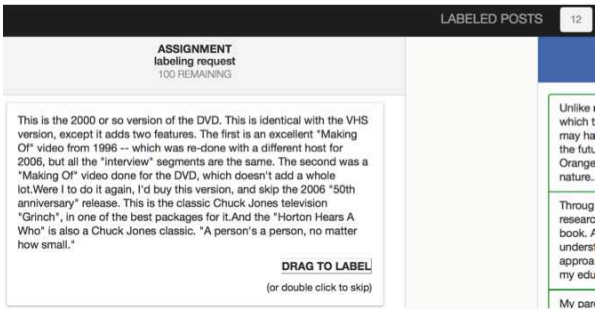
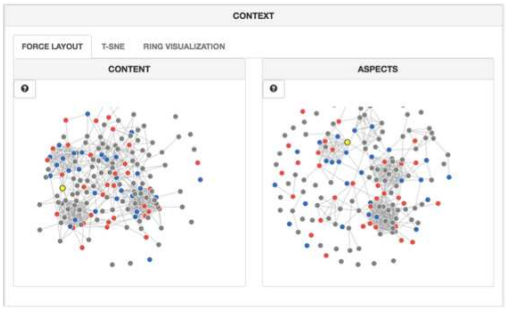


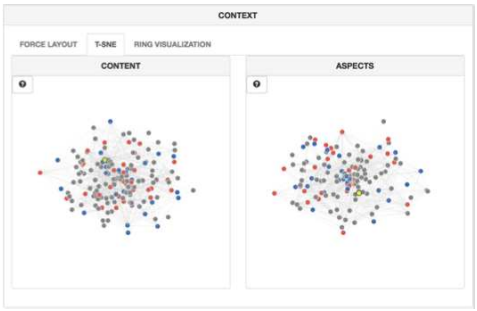
Figure 4.7: The case where visualization fails to provide accurate suggestions



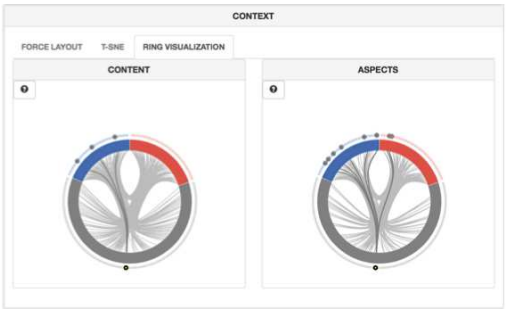
(a) Review Content



(b) Force Layout



(c) T-SNE plot



(d) Chord Diagram

Figure 4.8: Sample review with better visualization. Screen shot of a queried text in (a) and corresponding force-layout diagrams in (b), T-SNE in (c) and chord diagram in (d). The visualization becomes suggestive when the machine learning model becomes mature

view describes the user's high evaluation of the DVD, in comparison to the other two DVDs. Figure 4.8d is quite straightforward by the number of links to positive class samples. Figure 4.8b's aspect-based force-layout plot suggests positive label with surrounding blue points. Its BOW force-layout plot is not very informative due to too many grey points mapped around. Figure 4.8c's BOW force-layout plot, similarly, advocates positive label by blue points laying back. Its aspect-based force-layout plot gathers unknown points around, so it will be ignored by users easily. Hence, to some degree, the efficiency change of visualization is also the performance improvement of a classifier.

On the other hand, for the comment that experiment 1 requires hard work, the active learning algorithm tends to choose difficult instances that is hard for itself. Like the review in Figure 4.8a, a lot of reviewers compare the product with others they know, especially books and DVDs. Figure 4.8a is a user-friendly review, for many long reviews only focus on chapters from other books, or actors from other DVDs. This causes more workload from participants. Meanwhile, sarcasm is also present in the dataset. One example phrase our participant met is: ***if you are going to watch these two incredible movies in one day, you will definitely die***. Sarcasm is challenging for classifiers to detect or for the visualization interface to interpret. To realize the sentiment of these reviews is time-consuming for our participants. Another group of reviews contain both complains and compliments. Take the following review from our participant for example: ***The larger of the bowls was broken when I received it, so it was not packed very well. I had to use the smaller bowl and loved it.*** When the participant raised this issue, other participants expressed diverse opinions. This group data furthermore affect input from users.

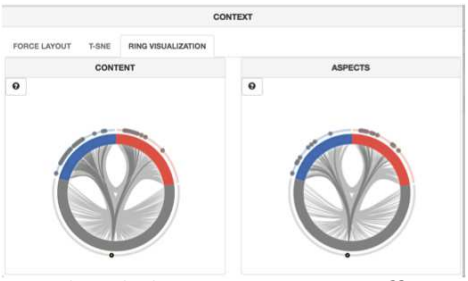
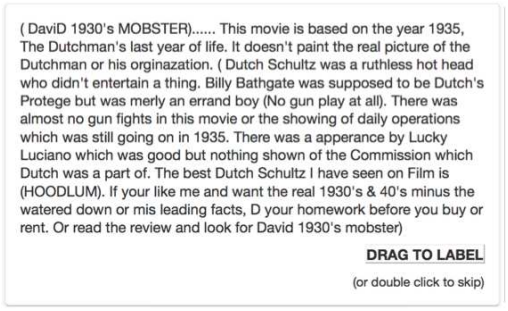
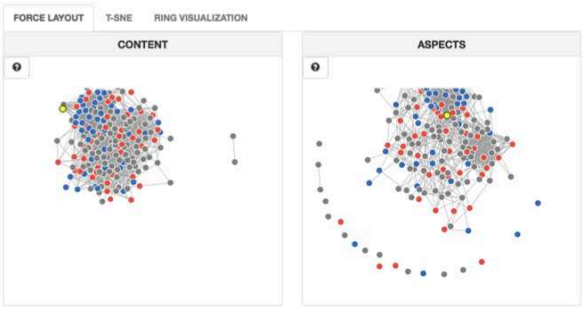


Figure 4.9: The case where chord diagram is not sufficient for decision making due to the similar amount of highlighted points from each class.

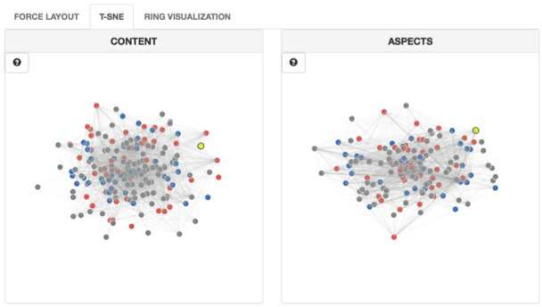
Moreover, all participants agree that the chord diagram is the most helpful visualization among the three projections, since the highlighted linking curves offer informative clues. However, chord diagram is not always useful. Figure 4.9 is a situation users will run into. It is not difficult to find that the numbers of marked points from the blue and red groups are comparable. Under this circumstance, chord diagram is not reliable as before, and users tend to shift back to force-layout and T-SNE panel. The example in Figure 4.8 mentioned above illustrates how users can determine the label by surrounding points. However, users stay more on the chord diagram panel than others during all three experiments.



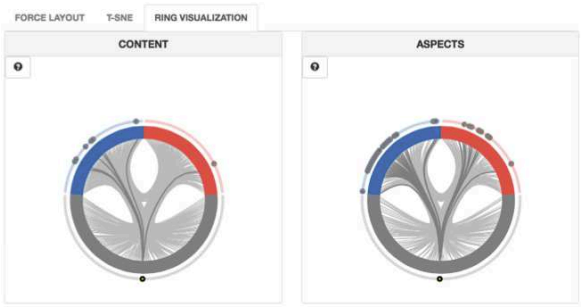
(a) Review Content



(b) Force Layout



(c) T-SNE Plot



(d) Chord Diagram

Figure 4.10: Common sample review. Screen shot of a queried text in (a) and corresponding force-layout diagrams in (b), T-SNE in (c) and chord diagram in (d). This is a case a user is more frequently exposed to.

Besides, we find that BOA and BOW models work in different local scenarios though neither can segregates two classes perfectly in the global scope. Aspect-based chord diagram in Figure 4.8d owns two additional link curves to negative samples. This happens again in Figure 4.10d, a more common case appearing during annotation. Though in many long reviews, BOW-based chord diagrams capture more connections

to other files, BOA model detects more relationships with other files when the BOW model meets with issues, like the ones in Figure 4.8d and 4.10d. BOA model projection provides valuable suggestion for the review in Figure 4.10. The text shown in Figure 4.10a, is a negative review, but refers to another movie the writer prefers. So both negative and positive features show up in the document. From the BOA-based force-layout plot in Figure 4.10b, our target point is encircled by red points, while in the BOW force-layout, it lays away from other documents. However, in the T-SNE plot, as shown in Figure 4.10c, both of them are closer to red class, which supports the right label. Though users are prone to chord diagram, cooperating with all plots is more beneficial to the decision making.

One more thing we found during the user study is that our Amazon dataset is not totally in English. Spanish exists in the review set. This is a noise for both English annotator and the classifier, because no relationships are found in chord diagram, and it lies away from all other points in both force-layout and T-SNE diagrams. However,

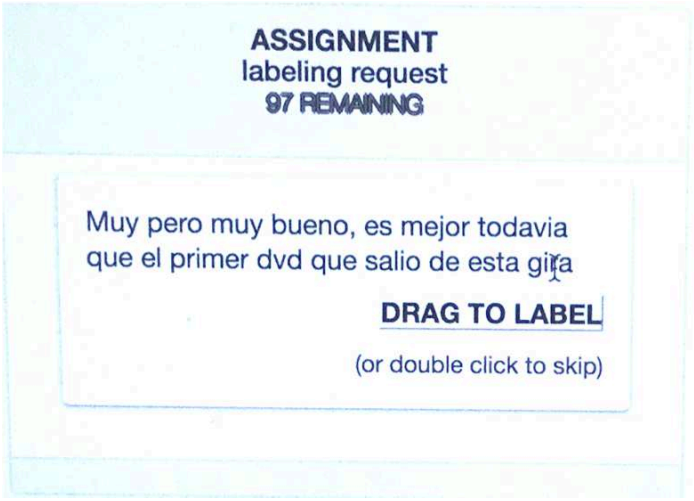


Figure 4.11: Spanish review in corpus met by a user during user study

detecting this reminds us to perform the language check before training the classifier. Figure 4.11 is the screen shot of the user who encountered the special case.

Chapter 5

Lessons Learned and Limitations

Our thesis work is a pilot study on the role of visualization in active learning with text classification problems. We believe that active learning can be further assisted by visualization techniques, especially interactive ones. Specifically, we see a role for the user in making an informed selection of instances from the list provided by the active learning system. Visualization can give the user an additional insight about the potential contribution of an active instance towards a "sharpening" of the inductive hypothesis. But we only had access to very limited resources including the pool of human users therefore, there are limitations. This chapter discusses lessons we learned from the existing work, and improvements in the settings for both visualization and learning algorithm.

5.1 Visualization

In chapter 4, we listed the user study tasks and corresponding results. Six participants were invited to perform the tasks. Though the results for the tasks were consistent, it is valuable to enroll more participants for the user study. Six is not a good number for a statistical significance study, but the results will be more reliable if the number of participants is increased. Moreover, these six participants are from Institute for Big Data Analytics, and are professionals in machine learning. A potential bias might exist among this group, which can be eliminated with more participants. Besides, users were instructed to do the tasks in the same order, which brings the influences of users' learning curves. In Figure 4.4, almost all lines fluctuate widely in the first 20 annotation efforts. This confirms the effects of the users' learning curves. One possible method to address the issue is that users can complete tasks in different orders. In addition, the main interface in Figure 3.5 only provides the force-layout diagram for users. This limits the amount of visual information a user can obtain each time. Aligning all visual aids in the interface can be a solution; this enhances

the brushing and linking effect, so that users can realize the changes and relations between instances during interactions with lesser efforts. Moreover, if the interface is extended for a multi-class classification task, aligning all document projections in the interface instead of the way it is now is more informative.

5.2 Machine Learning

The main task we study in the thesis is a two-class classification problem, which is a limit. It will be valuable if the same framework can be applied to a multi-class problem. The tool will be more robust, and the results will be more reliable if more classes are involved, which naturally triggers improvements in the interface mentioned in section 5.1. Another limitation of our study is the evaluation methods. In Figure 4.6, we presented results with all lines aggregated in the coordinate. Apparently,

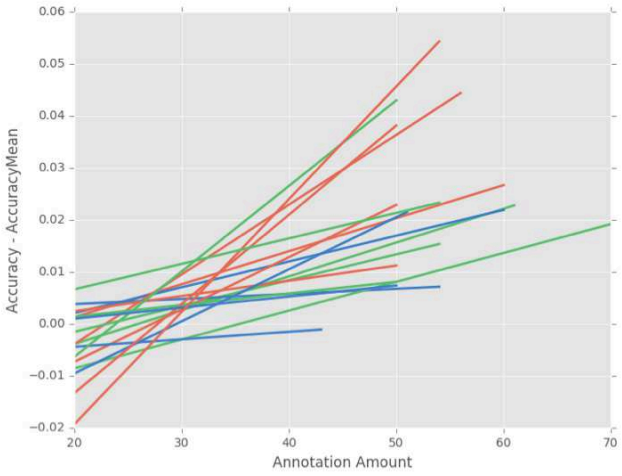


Figure 5.1: Performance from all users after the 20th annotation. Accuracy change of classifiers from all users in the three experiments; X-axis is the annotation effort contributed by users(how many annotations have been labeled at the given point), and Y-axis is the value of $Accuracy - Accuracy\_Mean$; Red lines are accuracy changes of experiment 1, green lines are accuracy changes of experiment 2, and blue lines are accuracy changes of experiment 3.

the slopes of red lines are sharper than those of the rest of the lines. But due to limited experiment data, the tendency is difficult to observe, thus we applied accuracy variance to illustrate the increasing scale of red lines and use it as the evaluation measurement, which is still not strong enough for the proof. One possible solution

is to focus on accuracy changes after the 20th label from users, which is displayed in Figure 5.1. As discussed in the previous section, from Figure 4.4 and 4.6, we can find that oftentimes, accuracy fluctuates widely when a user annotates the first 20 selected instances, which is caused by users' learning curves. Obtaining accuracies from the 20th annotation helps to get rid of influences from users learning curves. Indeed, slopes of red lines in Figure 5.1 are sharper than those of the rest of the lines. A time variable can be another solution for a more persuasive conclusion. For example, instead of annotation efforts from users, we can record the time users take for a certain task, so that the annotation speed of users can be calculated for the evaluation. If the visualizations provided by our system help the process, a user can complete a task faster than the the time required for the task without visual aids. Alternatively, we can also constrain the time that a user can spend on each task. In this case, the annotation amount will be the indicator of the conclusion. If visualization helps active learning, a user can label more instances than in a setting without visualizations.

Chapter 6

Conclusion and Future Work

We live in a data explosion age, where data analysis is predominantly important. In data analysis, supervised learning is the main part, which tries to infer a mapping pattern from labeled data. However, labeled data are very limited no matter how fast data increments these days. In the meantime, obtaining labels is very expensive. This effects the efficiency of data utilization. This motivates our work. We believe that the emerging visualization techniques can provide a hand for the annotation cost reduction.

In this thesis, we have investigated the role of visualization in binary class text classification with active learning problems. To evaluate this, we have proposed our system which involves both machine learning and visualization techniques. SVM and Naive Bayes classifiers are learning models we used, for the good performances they have achieved in the field. Accuracy of one classifier is the chosen measurement that reflects the performance of the classifier in our work. In the settings of active learning, we enroll pool-based active learning because of the static experiment dataset we own. The involved active query strategies are uncertainty measurement, QBC, and variance reduction. Random sampling is also employed for comparisons of experiments. For the visualization side, we propose our annotation interface. In addition to document visualization as in previous work, we add document projects. Documents are symbolized by points, and the relationship between documents is illustrated by distances between points in the plot. Force-layout, T-SNE, and chord diagram are three projections in our interface. In each projection section, we draw two visual diagrams, one based on the BOW model and the other on the BOA model.

Our dataset is multi-domain sentiment data from [Blitzer et al., 2007], which contains product reviews from Amazon.com on four domains: Book, DVD, Electronic, and Kitchen. Essential cleaning processes are performed with the dataset. To evaluate our hypothesis, we invited six colleagues to perform the user study. Results in chapter4

prove that given the same active learning settings, human annotators' inputs with visual aids boost the performance of the selected classifier. In addition, given the same visualization settings, active learning algorithms support improvements of the selected classifier. Moreover, visualization built from BOA is complementary to that on BOW. Though neither of them can achieve perfect global segregation for instances in different classes, BOA works better in local areas.

For the future work, it would be valuable to test our interface with other text datasets. Aspects-based visualization can possibly reach good performance for a non-sentiment dataset. Moreover, so far, our annotation interface has had a slow processing time, since our web interface applies python CGI service. With python CGI, requesting a service with ajax call from front end basically invokes a new process on the server. Starting new process, getting python scripts compiled, and receiving interruptions from users are very time-consuming, while our interface is very interactive. Thus, in the future, switching to another web service will make the system more user friendly.

Bibliography

[Aho et al., 1986] Aho, A. V., Sethi, R., and Ullman, J. D. (1986). *Compilers, Principles, Techniques*. Addison-Wesley.

[Arora et al., 2009] Arora, S., Nyberg, E., and Rosé, C. P. (2009). Estimating annotation cost for active learning in a multi-annotator environment. In *Proceedings of the NAACL HLT 2009 Workshop on Active Learning for Natural Language Processing*, pages 18–26. Association for Computational Linguistics.

[Berger, 2013] Berger, J. O. (2013). *Statistical decision theory and Bayesian analysis*. Springer Science & Business Media.

[Blitzer et al., 2007] Blitzer, J., Dredze, M., Pereira, F., et al. (2007). Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *ACL*, volume 7, pages 440–447.

[Bostock, 2012] Bostock, M. (2012). D3. js. *Data Driven Documents*.

[Bostock et al., 2011] Bostock, M., Ogievetsky, V., and Heer, J. (2011). D^3 data-driven documents. *IEEE transactions on visualization and computer graphics*, 17(12):2301–2309.

[Cohn et al., 1994] Cohn, D., Atlas, L., and Ladner, R. (1994). Improving generalization with active learning. *Machine learning*, 15(2):201–221.

[Graham, 1995] Graham, I. S. (1995). *The HTML sourcebook*. John Wiley & Sons, Inc.

[Harris, 1954] Harris, Z. S. (1954). Distributional structure. *Word*, 10(2-3):146–162.

[Hilbert and López, 2011] Hilbert, M. and López, P. (2011). The worlds technological capacity to store, communicate, and compute information. *science*, 332(6025):60–65.

[Kaelbling et al., 1996] Kaelbling, L. P., Littman, M. L., and Moore, A. W. (1996). Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285.

[Lagarde et al., 2016] Lagarde, J., Uszczynska-Ratajczak, B., Santoyo-Lopez, J., Gonzalez, J. M., Tapanari, E., Mudge, J. M., Steward, C. A., Wilming, L., Tanzer, A., Howald, C., et al. (2016). Extension of human lncrna transcripts by race coupled with long-read high-throughput sequencing (race-seq). *Nature communications*, 7.

[Liao et al., 2015] Liao, H., Chen, L., Song, Y., and Ming, H. (2015). Visualization based active learning for video annotation. *IEEE Transactions on Multimedia*.

[Lovins, 1968] Lovins, J. B. (1968). *Development of a stemming algorithm*. MIT Information Processing Group, Electronic Systems Laboratory Cambridge.

[Maaten and Hinton, 2008] Maaten, L. v. d. and Hinton, G. (2008). Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605.

[Manning et al., 2008] Manning, C. D., Raghavan, P., and Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA.

[Marcus et al., 1993] Marcus, M. P., Marcinkiewicz, M. A., and Santorini, B. (1993). Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330.

[McCallum et al., 1998] McCallum, A., Nigam, K., et al. (1998). A comparison of event models for naive bayes text classification. In *AAAI-98 workshop on learning for text categorization*, volume 752, pages 41–48. Citeseer.

[Mitchell, 1997] Mitchell, T. M. (1997). Machine learning. 1997. *Burr Ridge, IL: McGraw Hill*, 45:37.

[Nigam et al., 1998] Nigam, K., McCallum, A., Thrun, S., Mitchell, T., et al. (1998). Learning to classify text from labeled and unlabeled documents. *AAAI/IAAI*, 792.

[Russell et al., 2003] Russell, S. J., Norvig, P., Canny, J. F., Malik, J. M., and Edwards, D. D. (2003). *Artificial intelligence: a modern approach*, volume 2. Prentice hall Upper Saddle River.

[Sahlgren and Cöster, 2004] Sahlgren, M. and Cöster, R. (2004). Using bag-of-concepts to improve the performance of support vector machines in text categorization. In *Proceedings of the 20th international conference on Computational Linguistics*, page 487. Association for Computational Linguistics.

[Salton et al., 1975] Salton, G., Wong, A., and Yang, C.-S. (1975). A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620.

[Settles, 2009] Settles, B. (2009). Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison.

[Settles, 2011] Settles, B. (2011). Closing the loop: Fast, interactive semi-supervised annotation with queries on features and instances. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1467–1478. Association for Computational Linguistics.

[Shannon, 1948] Shannon, C. E. (1948). A note on the concept of entropy. *Bell System Tech. J*, 27:379–423.

[Stasko et al., 2008] Stasko, J., Görg, C., and Liu, Z. (2008). Jigsaw: supporting investigative analysis through interactive visualization. *Information visualization*, 7(2):118–132.

[Tong and Koller, 2001] Tong, S. and Koller, D. (2001). Support vector machine active learning with applications to text classification. *Journal of machine learning research*, 2(Nov):45–66.

[Turner et al., 2014] Turner, V., Gantz, J. F., Reinsel, D., and Minton, S. (2014). The digital universe of opportunities: rich data and the increasing value of the internet of things. *IDC Analyze the Future*.

[Van Rossum et al., 2007] Van Rossum, G. et al. (2007). Python programming language. In *USENIX Annual Technical Conference*, volume 41.

[Vapnik and Kotz, 1982] Vapnik, V. N. and Kotz, S. (1982). *Estimation of dependences based on empirical data*, volume 40. Springer-Verlag New York.

[Ward et al., 2010] Ward, M. O., Grinstein, G., and Keim, D. (2010). *Interactive data visualization: foundations, techniques, and applications*. CRC Press.