Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

b-it Bonn-Aachen International Center for Information Technology

R&D Project

Evaluation of Active Learning for Short Answer Grading

Mohandass Muthuraja

Jeeveswaran Kishaan

Submitted to Hochschule Bonn-Rhein-Sieg,
Department of Computer Science
in partial fullfilment of the requirements for the degree
of Master of Science in Autonomous Systems

Supervised by

Prof. Dr. Paul G. Plöger

M. Sc. Deebul Nair

January 2019

We, the undersigned below, declare that this work has not previously been submitted to this or any other university and that it is, unless otherwise stated, entirely our own work.

Date	Jeeveswaran Kishaan
Date	Mohandass Muthuraja

Abstract

Active learning is a machine learning paradigm which has seen useful applications in achieving good performance after learning from less amount of data. As supervised learning needs large amount of labeled data and short answer grading is not a "learn once and apply forever" task, it would be a logical step to experiment active learning on this task. This project concentrates on evaluating the performance of different active learning strategies on the task of short answer grading on three different datasets. In addition, useful features to be extracted from students' answers and different machine learning models are studied. Performance of the best setting have been experimented and analyzed based on the metrics such as accuracy, f1-score, and run time. Furthermore, the efficiency of active learning with variable batch size and seeding have been studied. A web-based GUI is designed and implemented to incorporate a short answer grading system using the best active learning setting.

Based on the experiments conducted in this research, it is affirmed that active learning reaches the performance of supervised learning with less amount of graded answers for training. Selection of answers for grading while training the model based on active learning query strategy performed better than randomly sampled answers in all the datasets. Margin based uncertainty sampling was found to be efficient when compared to other query strategies in most of the experiments. State-of-the-art features from Sultan et al.[73], achieved better performance than bag of words and tf-idf features despite taking a long time to extract the features. The effectiveness of machine learning models was seen to be depended on the dataset used. Querying the graded one by one and equal seeding worked well rather than batch querying.

Acknowledgements

This project would not have been complete without the help and constructive inputs from our supervisors. We want to thank Prof. Dr. Paul G. Plöger for providing us with an opportunity to work on this project and guiding us. We wish to extend our gratitude to M. Sc. Deebul Nair for his invaluable suggestions and comments during the weekly meetings throughout this project.

We thank Rubanraj Ravichandran for assisting us during the development of the web-based graphical user interface(GUI) for our grading system. The support provided by Ramesh Kumar by providing us with the implementations from his research and development project was beneficial and saved our time for which we are thankful. We would also like to thank Senthilkumar Kathiresan and Naresh Kumar Gurulingan for helping us with setting up the college cluster and for their valuable comments.

The survey done by Dr. Burr Settles on Active Learning, and the active learning framework (modAL) by Dr. Tivadar Danka helped us to incorporate active learning for the task of short answer grading.

Finally, we would like to extend our gratitude to our family and friends whose love and moral support helped us in steering this project at the right pace.

Contents

List of Figures

List of Tables

Contributions

This is done as a joint Research and Development Project by Mohandass Muthuraja and Jeeveswaran Kishaan. Both of us wrote the whole project after having a detailed discussion on the work done together. The person who took the lead in drafting the chapters has been tabulated below.

Chapter No.	**Title**	**Author**
1	Introduction	Jeeveswaran Kishaan
2	Related Work	Mohandass Muthuraja
3	Background	Jeeveswaran Kishaan
4	Methodology	Mohandass Muthuraja
5	Experimental Setup	Jeeveswaran Kishaan, Mohandass Muthuraja
6	Results and Evaluation	Jeeveswaran Kishaan, Mohandass Muthuraja
7	AI Assisted Grading System	Mohandass Muthuraja
8	Conclusions	Jeeveswaran Kishaan, Mohandass Muthuraja

1

Introduction

Assessing the knowledge of students is one of the most important phases of the learning process [57]. Different forms of assessments that exist today include multiple choice questions, fill-in-the-blanks, essay questions, and short answer questions. Prior works have shown that multiple choice questions and fill-in-the-blanks fail to capture the vital aspects of the acquired knowledge such as reasoning and self-explanation [75]. In contrast, questions which require the students to construct responses in natural language have been found to be more effective in assessing their grasp on the subject matter [65]. Essay questions and short answer questions belong to this category. This work is more concerned about short answer questions where students construct answers in natural language. According to Burrows et al. [27], short answer questions are characterized by the following aspects:

- "the question must require a response that recalls external knowledge instead of requiring the answer to be recognized from within the question
- the question must require a response given in natural language
- the answer length should be roughly between one phrase and one paragraph
- the assessment of the responses should focus on the content instead of writing style
- the level of openness in open-ended versus close-ended responses should be restricted with an objective question design."

Limited availability of teachers, online learning platforms, and individual or group study sessions done outside classrooms necessitated quick and efficient assessment of free text responses [57]. In addition, almost 30% of the teachers' time is spent on grading the assessments [52]. Computer assisted assessment / automatic grading evolved as a solution to this problem and a lot of research has been done on automating the grading of essay [43] and short answer responses [49, 62, 58]. Assessing the students' responses in time and giving faster feedbacks enables the students to realize the mistakes and learn from them or even appear for re-examination without wasting one semester.

Automatic short answer grading essentially deals with using computational methods to compute the grades for students' answers, thus helping the students in getting their feedbacks as soon as possible by heavily reducing the assessing time. Many automated approaches have been proposed in the past for grading short answer questions. Most of these methods include professors penning down the keywords expected in students' answers for every question or handcrafting the patterns and templates against which students' answers are matched to compute the final score. Though these methods managed to perform with reasonable accuracy, non-generalizability is one of the biggest drawbacks. Features such as keywords and pattern templates are very specific to the questions, and the same features cannot be used for new questions and different domains. Thus, a need for a generalized way of extracting features which would solve the problem of grading the answers in new questions and different domains arose and machine learning was considered as a possible solution. Natural language processing and machine learning techniques are used to extract the features from students' answers which are then used to train a model to compute the grades. Machine learning approaches could save the professors' time and effort in finding the best pattern templates and keywords to look for in students' answers.

Fig 1.1 shows a general workflow of automatic short answer grading as a pipeline. After creating a dataset of all the students' answers and model answers written by the teachers, useful features are extracted from the answers using natural language processing techniques. A model is developed based on these features to calculate the scores and deployed for use. Such a model learns only once from the available set of

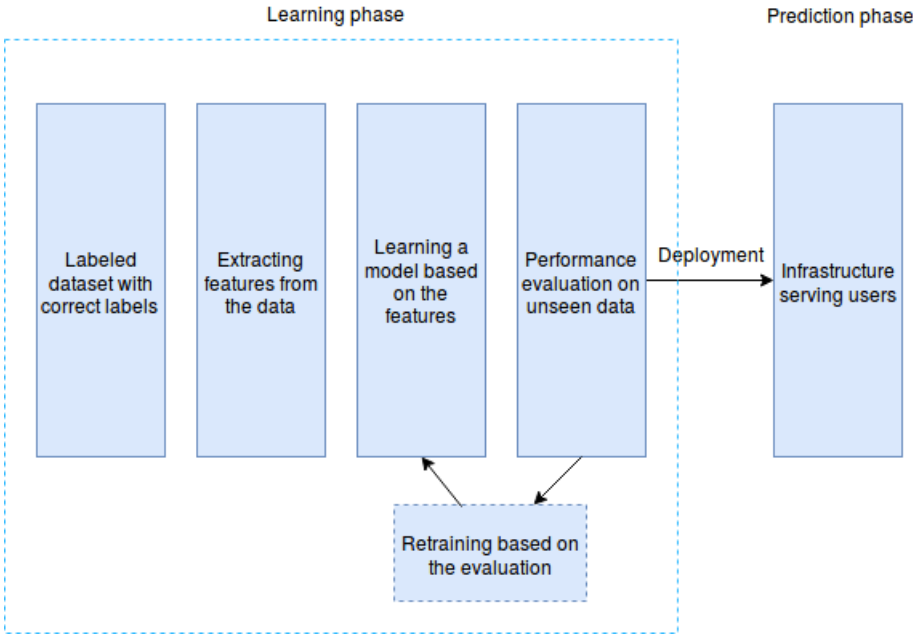


Figure 1.1: Workflow of automatic short answer grading [27].

labeled data and there is no feedback mechanism to fine-tune the parameters based on the wrong predictions after the deployment.

This work proposes an approach where a generic scoring model tries to learn this task of measuring the correctness of each answer continuously with a human in the loop. During the learning stage, the system selects the best samples for the human to grade which would eventually contribute to its knowledge base. Thus, it tries to reduce the human effort of going through all the answers while improving it's understanding of the problem on a cyclical and iterative basis.

Active learning seems to be the best choice for this task as it actively queries the human for grades of the samples it is most uncertain of. Fig 1.2 illustrates a typical workflow of active learning. According to Settles [67], "it is a subfield of machine learning which works under the hypothesis that if the learning algorithm is allowed to choose the data from which it learns it will perform better with less labeled data and training". Such a model queries a user / human expert for the labels of certain data samples in such a way that it can learn to produce the desired outputs with higher accuracy. By actively selecting the data samples to label, it reduces a

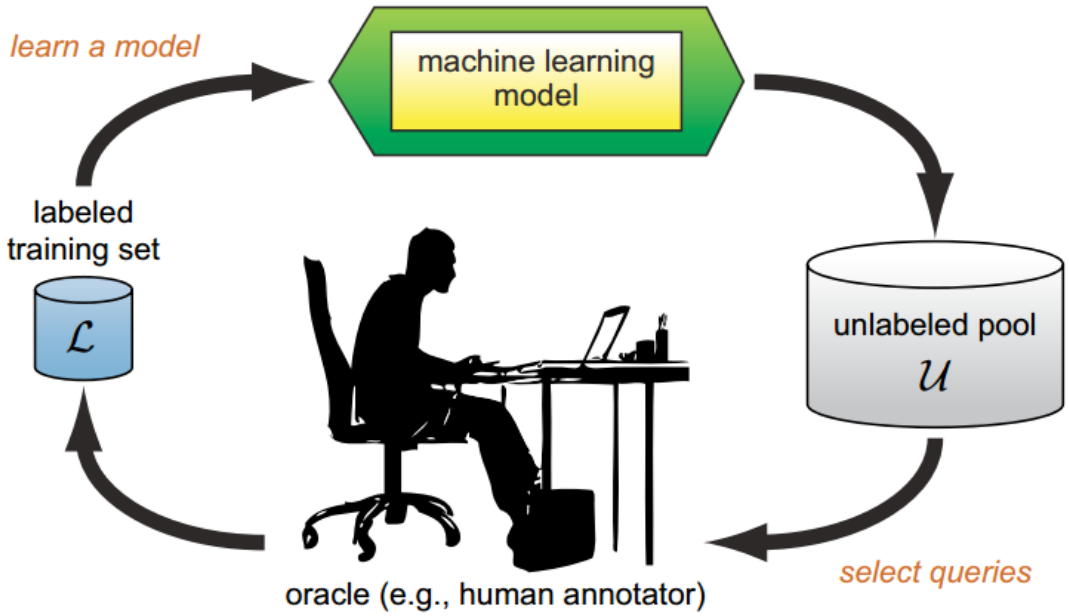


Figure 1.2: Workflow of active learning. Image from [67].

considerable amount of labeled training samples, thus, alleviating the problem of insufficient labeled data which is prevalent in supervised learning approaches where a large number of graded answers are used for learning a model. In addition, it would be a solution to deal with the diversity in answers as there is generally no single best response for an open-ended question.

1.1 Challenges and Difficulties

The conventional methods in automatic short answer grading belong to a learning paradigm called supervised learning where the right grade for every student answer in the training phase is available. Though these approaches were able to produce decent results, they suffer from many shortcomings such as;

- lack of sufficient amount of labeled training data in the domain to learn the models. Reasons include annotation cost, privacy, availability, and the quality of the correct answers.
- the failure to capture the different wordings/phrasing of the students while

trying to answer the short answer questions. It is obvious that anticipating all different ways of answering his questions is practically impossible for the professor.

- inability to capture consistent patterns of misunderstandings among students. Ability to recognize such patterns would enable the automated systems to provide useful feedback to students as to why there was a reduction in marks awarded.

- accounting for small deviations in the answers which might affect the whole meaning of the sentence (for ex. in mathematical terms, though each and every word of the student's answer align with that of the professor, a small negation or inverse operation would change the whole meaning).

- finding a way to understand the underlying concept of various students' answers and bagging the similar ones (or the right and wrong ones separately) is also a very tedious task.

- being a passive learner, these models learn the rules once and apply them on new input answers. Thus, it would be very difficult to achieve robustness when applied to new data over time.

1.2 Problem Statement

It is common in the task of short answer grading that obtaining the correct grade for all different types of student answers is time-consuming and subjective to the grader in nature. In addition, it costs more to have a large amount of dataset (questions, answers, and their grades) to train complex models on them. Thus, a mechanism which can learn on a small amount of graded answers obtained through a query strategy and can produce results on the remaining ungraded answers is more appropriate to this task. In contrast to supervised learning, active learning achieves comparable results with less amount of data. Active learning is an algorithm which chooses a small amount of data which is more efficient and more informative to the learning process. Short answer grading is not a "fit once and apply forever" task as new questions are added every now and then, and the students' answers contain a

vast range of lexical diversity. Active learning can be easily fitted on a new dataset as it includes an adaptive mechanism of training the model which takes into account the new set of data in addition to the one on which it learned earlier. Hence, active learning is chosen to be evaluated for our task of short answer grading as it has achieved comparable results with supervised learning techniques with less number of labeled data for training [76].

This work differs from the approach illustrated in Fig 1.1 by incorporating a feedback mechanism which allows the model to learn continuously based on new labeled input samples. As illustrated in Fig 1.3, we propose a model which implements active learning for the task of short answer grading.

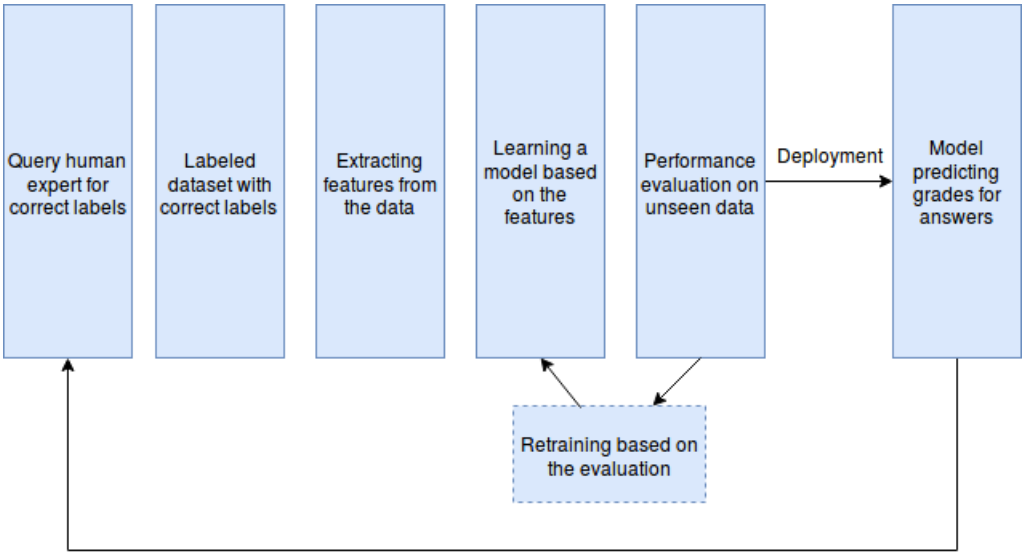


Figure 1.3: Workflow of active learning in automatic short answer grading

In this research project, various active learning strategies are evaluated as a potential solution to overcome the deficits of automated short answer grading. Thus, issues such as high cost of labeled data, understanding common misconceptions among the students, lexical diversity in answers, and inferior performance are addressed in this work. Various natural language processing techniques such as text preprocessing, semantic similarity, and the best features to be extracted from the answers are experimented for the task of short answer grading. Active learning strategies are evaluated based on aspects such as seed selection, instance sampling, and batch size.

The best strategy from each stage is selected for the implementation of the final model. A generic interactive model for assessments in different domains is implemented based on the results of experiments performed using a web-based Graphical User Interface (GUI). All techniques are analyzed on accuracy, reliability, amount of human action(clicks) required, flexibility for different domains, runtime, and user experience.

2

Related Work

2.1 Automated Short Answer Grading (ASAG)

2.1.1 Concept mapping based methods:

Concept mapping based techniques segments student answers into different concepts and the score is given based on the presence of concepts in the reference answer. [27]

- Callear et al.(2001) [28] developed a computer-assisted assessment (CAA) termed as Automated Text Marker (ATM). ATM computes the student grade based on the number of matching concepts between the student and reference answer where the concepts are given certain weight based on their importance.
- Leacock et al. (2003)[49] developed an automated scoring system called C-rater for ETS(Educational Testing Service). C-rater generates model sentences based on the concepts from the student and the reference answer. The model sentences undergo linguistic processing using Natural Language Processing(NLP) tools and then scored based on the availability of the expected concepts.

2.1.2 Information extraction based methods:

Information extraction systems extract a pattern from the student and the reference answers. Patterns from reference answers are kept as a template(parse

trees, regular expression) and are matched with the patterns from student answers. [27].

- Bachman et al. (2002) [25] developed a short answer scoring system called WebLAS. Teachers' answers are converted into significant segments from parsed representation. These converted segments are assigned specific weights based on their importance confirmed by the teachers. Grades are given then based on the regular expression matching.

- Mitchelle et al. (2002) [56] discuss a software called AutoMark. AutoMark assigns a grade based on matching the mark scheme templates obtained from the parse tree representation.

- Oxford UCLES,[62] another information extraction short scoring system that works based on hand-crafted pattern matching. These patterns are provided by the human experts of the particular domain. [62] also compares handcrafted patterns with the patterns extracted through machine learning techniques and claims that the handcrafted patterns work better.

- Jordan and Mitchell (2009) [45] developed a graphical user interface(FreeText Author) for short answer grading. The teacher provides the reference answers and keywords in the GUI. The keywords also mostly associated with their synonyms. Based on these inputs, a template is generated using NLP techniques and are matched with the student answer to compute the grade. Since the templates are obtained using NLP techniques, the teachers are not required to have much knowledge of NLP.

- Raheel Siddiqi (2010)[71] proposes a grading system. This system works by matching the specified structure of the student and the teacher answer. These structures are created based on the content of the answers by "structure editor." These structures are specified using question answer markup language (QAML).

- Meurers et al. (2011) [55] and Hahn et al. (2012) [42] developed a short answer scoring system where the scores are provided based on the matching of an abstract representation of the text obtained by Lexical Resource Semantics

(LRS) method. The abstract representations were converted into graphs before matching.

- Ramachandran et al. (2015) [63] developed a short answer scoring system based on the pattern matching where the patterns were extracted automatically from word-order graphs and lexico-semantic matching methods. Patterns were not only extracted from the teacher's answer but also the best student's response.

2.1.3 Corpus based methods:

Corpus-based are statistical methods where a big corpus (Wikipedia, Google, etc.,) is used to obtain information such degree of similarity, synonyms, etc., It is useful in short answer grading as it is helpful in detecting paraphrasing of reference answers.[27]. These methods have played a vital role in automated essay scoring is now also used in ASAG.

- Nielsen et al., (2009) [60] developed a dependency-based classification component called Intelligent Tutoring System. Here the features used for machine learning were obtained using the lexical corporal statistics.

- Mohler and Mihalcea (2009) [58] computed score based on the similarity between student and reference answer. The main contribution of this work is the comparison between the similarity obtained using eight knowledge and two corpus-based methods. The influence of size and the domain of the corpus is also analyzed.The authors claim that corpus domain plays a vital role than the size of corpus while obtaining the similarity.

- Gomaa and Fahmy (2012) [41] compares string similarity and corpus-based similarity for the task of automatic short answer grading. This work also examines the significance of two big corpora. The authors also suggest that the short answer grading could be interpreted as a similarity task.

2.1.4 Machine learning based methods:

Machine learning based approaches refers to training a model utilizing the features extracted using natural language processing techniques [27].

- Mohler et al. (2011) [57] uses the similarity approach from his previous work [58] along with graph alignment features to train a machine learning model for ASAG. The authors claim that by using both the semantic and graph alignment features helps in grading the student answers more efficiently. Dependency graphs of both student and reference answers help in constructing a structural component for automatic grading.

- Sultan et al. [73] used features such as word alignment, embeddings, question demoting, term weighting and length ratio extracted from the student and the reference responses for training a supervised model. Both regression and classification were performed in two different datasets respectively. The trained supervised model is used to assign grades to the student answers.

- Zesch et al. (2015) [77] used unsupervised machine learning method for grading the student answers. Students answers were clustered using a k-means clustering algorithm, and the human-provided the labels. This clustering model was used to predict the grades.

- Basu et al. (2013) [26] developed an approach termed as "Powergrading." Powergrading groups the student answers into clusters and sub-clusters. This helps the teacher to grade a bunch of answers at a time and also helps in understanding the common mistakes or misconceptions made by the students. Grouping the answers also helps in giving effective feedback for the whole group.

2.2 Active learning in the context of Natural language processing

- Dmitriy et al. (2011) [34] developed an unsupervised language model and incorporated active learning strategies for sampling instead of random sampling to increase the learning rate. It was proved that active learning helps in capturing rare classes and was used in word sense disambiguation.

- Rosa et al. (2012) [37] used active learning for the task of text classification and claims that active learning helps in achieving the same performance compared

to passive learning with much lesser training data. Distance-based sampling and Diversity based sampling was used in this task.

- Andrew et al. (1998) [53] incorporated active learning with Expectation Maximization(EM) to circumvent the use of large dataset. By labeling 50% of the data, the authors achieved the same level of accuracy achieved by training with the complete labeled dataset.

- Simon et al. (2002)[74] discussed standard inductive and transductive settings of active learning. In the task of text classification using support vector machines, it was proved that by using active learning the effort of labeling was drastically reduced in both the standard transductive and inductive settings.

3

Background

3.1 Natural Language Processing

Natural Language Processing evolved as a branch of artificial intelligence by combining fields such as computer science and linguistics. It helps machines to comprehend, manipulate and generate human language. Natural language processing gained interest as it seemed promising in various tasks such as human-machine interaction / communication, machine translation, spam detection, voice controlled devices, summarizing texts, and question answering in search engines [12]. Thanks to the availability of a large number of unstructured data that is created everyday (for eg. social media and medical records), efficient algorithms and powerful computing systems, a lot of advancements have been made in this field in the last few decades.

Though natural language processing is an exciting field which enables everyone to communicate with the computers in their own language (as opposed to the programming language or low level language used by the computer), understanding and making sense of human language has many layers of difficulty. The complex and diverse nature of human language, unique sets of rules and grammars for different languages, new meaning of words, different meanings for the same word, and abbreviations restrict the ability of computers to make sense of natural language. Humans have the capability of understanding, communicating, and construing very elaborate and subtle meanings, but formally defining the rules that govern the language is still a challenge for them [40]. This hinders the capability of taking a

rule-based approach towards natural language understanding by computers as rules differ for every language and word ambiguations is commonplace.

Many Natural Language Processing tools and methods are used to bring the natural language into a format which could be made sense by computers. These techniques enable computational methods to manipulate data and make inferences from them such as how similar two sentences are, whether one sentence could be infered from another sentence, and determining the topic of an article. The most prominent NLP methods and tools are discussed below.

3.1.1 Pre-processing methods in NLP

Pre-processing the text helps in bringing the input data into a form which could be digested by the algorithm. While such methods would help in computational methods, it is important to note that some of the details could be lost from the original input and the key idea is to know the pros and cons of each of these methods and to apply them where necessary.

Tokenization

Tokenization is a process of text segmentation which helps in splitting the text into sentences or sentences into words. Such sentences or words are called as 'tokens'. Common ways of doing this include splitting on whitespaces and splitting on punctuations. However, splitting a sentence based on whitespace or punctuations is not a straightforward task as complex words need careful attention while splitting them; e.g. "Los Angeles", "Mr. Mercedes", "i.e.", "Simon's cat", "on-campus housing" etc. To deal with such cases, rule-based or machine learning approached could be applied [46].

Normalization

This method is utilized to bring all text into the same format so that manipulating them would be less biased and uniform across different forms of sentences and phrases. Simple tasks such as removing punctuations, converting numbers into their word forms, and converting all text to lowercase help in putting all the data on equal

footing. Though these methods sound easy, attention must be given to minute details; e.g. converting "US" to "us" does not give the same meaning. Normalization in NLP is equivalent to normalizing all the image pixels in a computer vision task so that the algorithm doesn't give importance to more colorful or vibrant images. Other normalizing methods that are used in this report includes stemming and lemmatization.

Stemming and Lemmatization

These words convert the words into their base or common form. Stemming does this by "dropping unnecessary characters, suffixes, prefixes, infixes, and circumfixes from a word in order to obtain a word stem" as mentioned in [9]. The results of stemming could be used to find relationships and similarities between large text documents. An example of stemming could be seen as follows.

input = "I started studying yesterday"

output = "I start studi yesterday"

Lemmatization converts the words to the base form after going through the vocabulary and doing a morphological analysis like part-of-speech of every word [46]. Lemmatization can produce better results by utilizing WordNet's lexical database of English. It offers more accuracy at the cost of time as it is an intensive and slower process. Stemming is more useful where simple database queries are carried out and time is an important factor, while lemmatization could be used for more complex tasks such as sentiment analysis.

Stop words removal

Stop words should be filtered out before processing the text further as they occur more frequently in text and contribute little to the overall meaning of it while serving the purpose of connecting different parts of a sentence. Some of the most common stop words in English include "the", "so", "and", "it", and "a" [14]. As can be seen from the following example, stop word removal does not change the overall meaning much.

input = "The player hits the ball over the fence"

output = "player hits ball over fence"

3.1.2 Bag-of-Words

Machine learning algorithms often require structured inputs of fixed lengths. This is one of the major issues in feeding the text into a machine learning model. Text must be converted into numbers or vectors in order to use them in the algorithms. Constructing vectors out of text is called as feature extraction or feature encoding. Bag-of-words in one such simple technique of converting text into vectors of fixed length.

Bag-of-words captures the count of each unique word in a text. It is made up of a collection of unique words and a measure of occurrence of those words in the document. This technique simply bags same words ignoring the order and structure of the words in the text. Bag-of-words works well for topic modeling (determining the topics that occur in a collection of text) and text classification since the intuition behind this is that similar documents mostly share the same words [4].

An example of Bag-of-words model:

Consider each of the following lines as a single text document.

"The boy fell from the tree"

"The boy fell from the tall tree"

"The kid climbed the tree"

"The girl is sick"

The unique words in all of these texts after converting them to lowercase would be;

"the"
"boy"
"fell"
"from"
"tree"

"tall"
"kid"
"climbed"
"girl"
"is"
"sick"

In order to convert the first text into a vector, these unique words are stacked in a row and the number of occurrence of each of these words in the text is represented as a number in the box indexed by the corresponding word. This results in a vector of length 11. Fig 3.1 shows how this is done.

2	1	1	1	1	0	0	0	0	0	0
the	**boy**	**fell**	**from**	**tree**	**tall**	**kid**	**climbed**	**girl**	**is**	**sick**

Figure 3.1: Vector representation of the first sentence.

Similarly, all other texts are converted into a vector of same length as shown in Fig 3.2.

Text	**Vector**
The boy fell from the tree	21111000000
The boy fell from the tall tree	21111100000
The kid climbed the tree	20001011000
The girl is sick	10000000111

Figure 3.2: Vector representation of all the sentences.

It can be seen that the first three texts are closer to each other in the vector representation and the last sentence is significantly different from the first three. Such a representation is very consistent in extracting features from new documents and can scale to larger texts. However, when using a large corpora, the vectors might get sparse due to very few known words in the vocabulary and many new words that get ignored in the vector representation. This ends up in a large sparse matrix which demands more computational power and memory. Techniques such as ignoring frequent words, stemming the words to their root form, correcting the misspelled words and grouping similar words could be used to reduce the size of the sparse matrix [4].

Though the bag-of-words model is very simple to interpret and very flexible, it has few limitations such as;

1. Vocabulary - The words to be included in the vocabulary, which in turn directly affects the vectorization of text and the size of the sparse matrix, should be designed and chosen carefully.

2. Sparsity - The sparse representation of Bag-of-words model contributes to the high space and time complexity. The models are forced to infer knowledge from a very large representation which is quite cumbersome. This should be taken care of while modeling the matrix in such a way that it is as less sparse as possible.

3. Meaning - The semantics of the text is ignored in the bag-of-words model as the order of the words are not considered. Thus, the distinct meaning of texts with the same words arranged in different order will not be captured by this model.

3.1.3 Term Frequency - Inverse Document Frequency (tf-idf)

Tf-idf is a statistical tool to determine "how important a word is to a document in a collection or corpus" [21]. Tf-idf is used widely to weight important terms in applications such as text-based recommendation systems, document classification and information retrieval. It captures the most important words in a document by combining two algorithms, namely, term frequency and inverse document frequency.

- Term frequency - This captures the number of occurrence of a word in a document. To consider the length of the document while calculating the term frequency, the frequency of the words is divided by the total number of words in the document. The term frequency of a word t in a document d is defined as mentioned in [20];

$$tf(t,d) = \frac{\text{number of occurences of term in the document}}{\text{total number of all words in the document}} \tag{3.1}$$

- Inverse document frequency - Words like "the" occur a large number of times in most of the documents and these words get more score in the term frequency calculation. However, such words have very less amount of information for most of the tasks. Hence, it is preferable to assign a lower score to these common words which contain no information. Inverse document frequency achieves this by calculating a low score to frequently occurring words and increasing the weights of the words that occur rarely. Inverse document frequency for a word t in a document d is defined as mentioned in [20].

$$idf(t) = \frac{\text{total number of documents in the corpus}}{\text{number of documents with term t}} \tag{3.2}$$

The tf-tdf score of the term t and document d is calculated as;

$$tf - tdf(t, d) = tf(t, d) \times idf(t) \tag{3.3}$$

Words which have a very high term frequency and low document frequency among the collection of documents, thus capturing the important and rare unique words pertinent to the task [21].

3.1.4 Word Embedding

Word embedding is a NLP task which converts text into an array of numbers or real valued vectors. Such a representation of words proved to be useful in many tasks such as semantic similarity measures, text classification, and machine translation. Though one-hot-encoding (where a sentence is represented by a matrix with its dimension equal to the number of words in the sentence and each row made up of 0's with a 1 in the dimension corresponding to the word) was used in the beginning as a word embedding model, it has issues such as being a sparse matrix, redundant memory, and inability to capture the context or the relationship between different words in the same sentence.

More compact and meaningful representations of words evolved later with the use of neural networks and a large corpora to learn the distributions of words on. The most popular algorithms which exist today for efficient word embedding include Google's Word2Vec [22], Facebook's fasttext [7], and Stanford's GloVe [8]. These

algorithms were able to capture both the semantic relationships and context of each word and represent them in a more compact way (dense vectors) [18]. Fig 3.3 below depicts a 2D representation of word embeddings where similar words in context and meaning have been grouped together in the vector space.

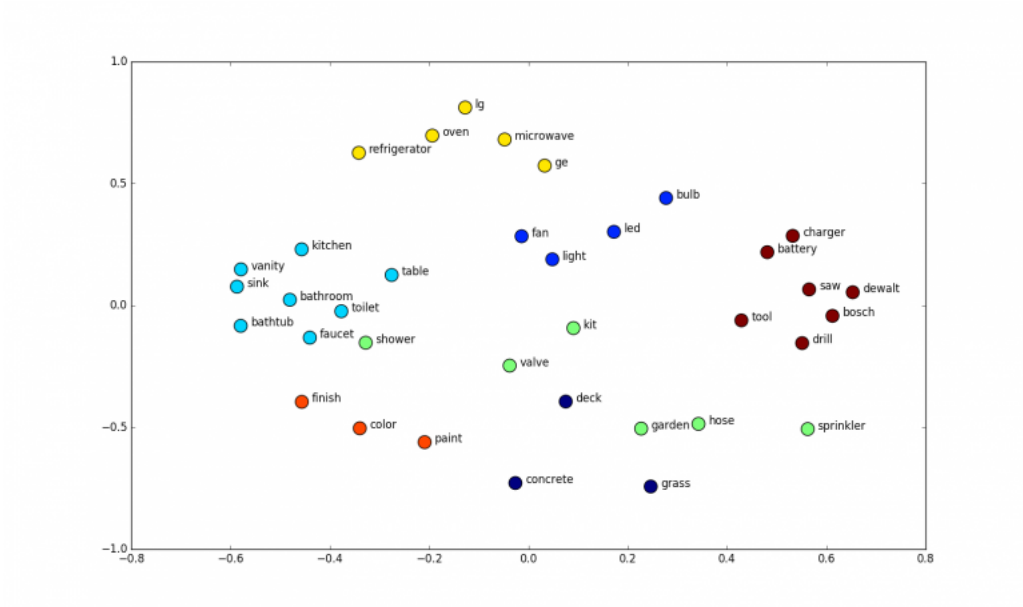


Figure 3.3: Example 2D word embedding space, where similar words are found in similar locations. Image from [1].

3.1.5 Part of Speech(POS) tagging

Same words give different meaning when they occur in different positions of the sentence. The context and the order of words are very important in determining the semantic meaning of a sentence. Bag-of-words and tf-idf do not capture such differences and consider the same words without any regard to their placement. The meaning of the word "address" in "May I have your address?" and in "I had to address to the public" is different. Hence, the position and context of the word in every sentence is imperative to capture the sematics of it.

Part-of-speech of a word is the function of that word in a sentence. Commonly seen parts of speech include nouns, verbs, adverbs and adjectives. In computational

linguistics, such tags are determined based on factors such as its definition, context, its neighboring words and its relationship with them [17].

Tag	Description	Example	Tag	Description	Example
CC	Coordin. Conjunction	*and, but, or*	SYM	Symbol	+,%, &
CD	Cardinal number	*one, two, three*	TO	"to"	*to*
DT	Determiner	*a, the*	UH	Interjection	*ah, oops*
EX	Existential 'there'	*there*	VB	Verb, base form	*eat*
FW	Foreign word	*mea culpa*	VBD	Verb, past tense	*ate*
IN	Preposition/sub-conj	*of, in, by*	VBG	Verb, gerund	*eating*
JJ	Adjective	*yellow*	VBN	Verb, past participle	*eaten*
JJR	Adj., comparative	*bigger*	VBP	Verb, non-3sg pres	*eat*
JJS	Adj., superlative	*wildest*	VBZ	Verb, 3sg pres	*eats*
LS	List item marker	*1, 2, One*	WDT	Wh-determiner	*which, that*
MD	Modal	*can, should*	WP	Wh-pronoun	*what, who*
NN	Noun, sing. or mass	*llama*	WP$	Possessive wh-	*whose*
NNS	Noun, plural	*llamas*	WRB	Wh-adverb	*how, where*
NNP	Proper noun, singular	*IBM*	$	Dollar sign	*$*
NNPS	Proper noun, plural	*Carolinas*	#	Pound sign	#
PDT	Predeterminer	*all, both*	"	Left quote	(' or ")
POS	Possessive ending	*'s*	"	Right quote	(' or ")
PP	Personal pronoun	*I, you, he*	(	Left parenthesis	([, (, {, <)
PP$	Possessive pronoun	*your, one's*	)	Right parenthesis	( ],), }, >)
RB	Adverb	*quickly, never*	,	Comma	,
RBR	Adverb, comparative	*faster*	.	Sentence-final punc	(. ! ?)
RBS	Adverb, superlative	*fastest*	:	Mid-sentence punc	(: ; ... – -)
RP	Particle	*up, off*			

Figure 3.4: Penn Treebank part-of-speech tags. Image from [46].

Determining the part of speech of every word in a sentence is imperative to many applications of NLP. Tasks such as text to speech and word sense disambiguation require POS tagging a pre-requisite to achieve better performance. In order to stress the words in line with their semantic interpretation, text to speech application needs to know the function of each word in the sentence. Similarly, word sense disambiguation makes use of POS tags to find out the sense in which a word, which has multiple meanings, has been used in a sentence. Rule-based tagging, statistical methods which use Hidden Markov Models, and memory based tagging are some examples of computational algorithms which are used for assigning parts-of-speech. Penn Treebank tagset is one of the most popular tagset used in the modern language processing algorithms [46]. Fig 3.4 shows the list of 45 POS tags that are included

in the Penn Treebank tagset.

3.1.6 Named Entity Recognition (NER)

It is often the case that detecting the real world entities in text makes information extraction easier. Named entity recognition is such a technique which goes through the entire text, extracts keywords (nouns) and maps them to pre-defined categories of real world entities such as locations, companies, cities, organization, dates, and individuals. For e.g., an unannotated text "Microsoft was flourishing in business when Barack Obama was the president of the United States." would be annotated by a named entity recognition algorithm as follows;

{"Microsoft": Group, "Barack Obama": Name, "United States": Place}

NER facilitates many real world applications. Classifying and managing a large news database is very time-consuming and error prone when it is done manually. Instead, a NER algorithm can swiftly extract the important entities such as names, locations and organizations mentioned in each article and group them in a hierarchical order using the tags. This ordering could be used for efficient retrieval and categorization of the articles. Optimized search algorithms use NER to concentrate on the keywords for searching rather than the naive method of going through the whole text. Recommendation systems in news and other blog websites implement NER to capture the organizations, places and individuals each reader is interested in and recommend similar news articles or blogs to them. Thus, structuring and categorizing a large unannotated text using NER could be facilitating faster language processing for more complicated inferences [11].

3.1.7 Dependency Parsing

All the techniques which have been described above work in the syntactic level. A lot of information about the context is lost when a task is designed to only take into account the syntactic properties of documents. Such techniques fail in the case when the words have more disambiguation with multiple meanings. Processing those words

in a semantic level would help build a more meaningful and accurate description of a sentence. Dependency parsing helps in sentence level disambiguation by eliciting grammatical relationship between words in a sentence. Based on the verbs, it tries to find out the relationship between the other nouns, adverbs and adjectives in the form of a head-dependent or parent-child format. For e.g., the result of parsing the sentence "Mary laid the mug on the table" using spaCy library's dependency parser is shown in Fig 3.5

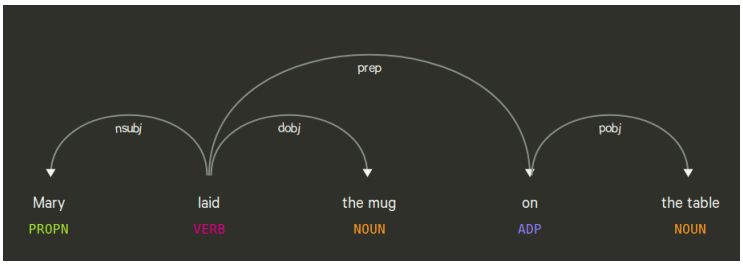


Figure 3.5: Dependency parsing of a sentence using spaCy library [6].

As can be seen from Fig 3.5, it could be easily inferred that "Mary" is the subject (denoted as "nsubj") and "the mug" is the object (denoted as "dobj") of the verb "laid". Parsing in this manner is utilized for question answering systems and in grammar checking tools. nominal subject(nsubj), clausal subject(csubj), direct object(dobj), indirect object(iobj), determiner(det), prepositional modifier(prep), and object of preposition(pobj) are some of the grammatical relations found in dependency parsing [33].

4

Methodology

This chapter and the equations mentioned here were adapted from the survey on Active Learning [67].

4.1 Active Learning

Active learning belongs to a special case of semi-supervised learning algorithm where the learner is allowed to query the user to get the labels for data points which will help the learner to perform better [2][67]. According to [67], when the learner is allowed to choose the data points for learning, the performance of the learner is better with less labeled data.

Supervised learning algorithm performs well when the model is trained with a large number of labeled instances. But the labeled instances are expensive, time-consuming, and difficult to obtain. Active learning helps in overcoming the situations where there is plenty of unlabeled data is available which are difficult to label or the cost of labeling them is high. The main aim of the active learning is to improve the accuracy by labeling less number of instances. The instances that need to be labeled is determined by the active learning query strategies.[67]

Let us consider the Fig 4.1 based on experiments done by [67]. It consists of 400 2-dimensional data points from two Gaussian centered (-2,0) and (2,0) having standard deviation $\sigma = 1$. These data points belong to two classes. (each class has 200 data points).

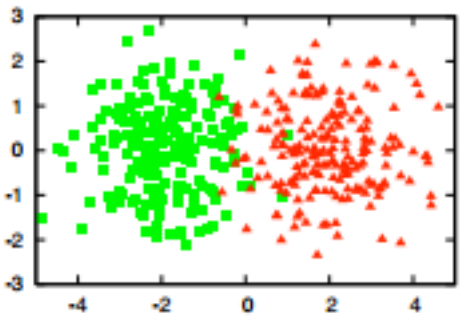


Figure 4.1: Sample data points in a 2D feature space . Image from [67].

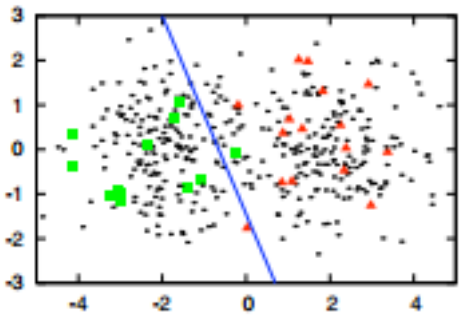


Figure 4.2: Decision boundary obtained by random sampling. Image from [67].

Fig 4.2 shows the decision boundary obtained using a supervised learning approach after selecting 30 data points randomly for labeling. We can infer from the line that the linear decision boundary fitted by a logistic regression model is sub-optimal. This model achieves 70 % accuracy on the remaining unlabeled points because of the poor selection of data points for labeling.

Fig 4.3 shows the decision boundary obtained using active learning approach. The active learning approach selects 30 points close to the decision boundary for labeling. By using the label for these data points, the classifier achieves 90 % accuracy which is significantly better than the approach by random sampling of data points. [67]

4.2 Active Learning Scenarios

There are three main problem scenario that can occur when the learner queries the user. They are

- membership query synthesis

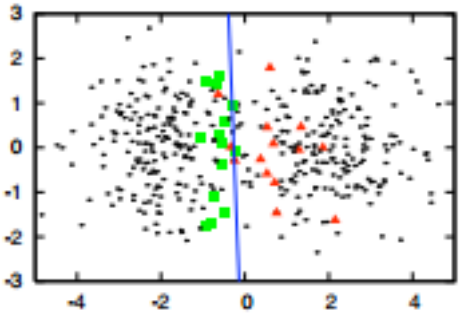


Figure 4.3: Decision boundary obtained by sampling using active learnig strategies. Image from [67].

- stream-based selective sampling
- pool-based sampling

4.2.1 Membership query synthesis

In membership query synthesis, the learner is allowed to choose any unlabeled data points which can include the a new query that the learner generates from the underlying distribution instead of the existing data points.

Figure 4.4: Membership query synthesis. Image from [3].

Fig 4.4 shows the workflow of membership query synthesis.The shortcoming of this setting is that the query generated by the learner can sometimes be unrecognized for the human to label. When applying this setting to the task of handwritten classification, the learner generates hybrid characters that has no natural semantic meaning [67][3].

4.2.2 Stream-based selective sampling

In this setting, the instances are drawn from the data source in a sequence and the learner decide whether to query the particular instance or not.

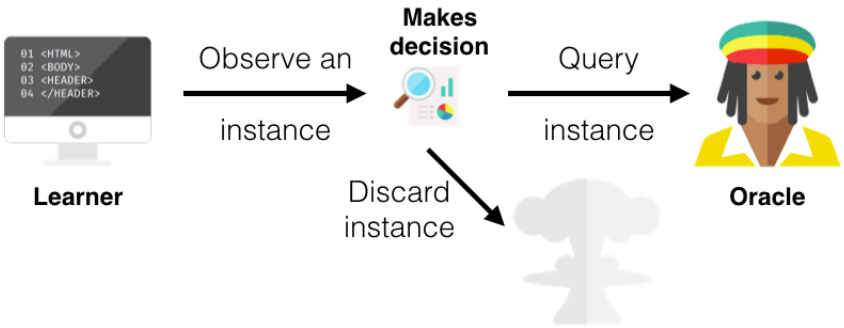


Figure 4.5: Stream-based selective sampling. Image from [3].

Fig 4.5 shows the workflow of membership query synthesis.The decision on querying the instance is decided by various query strategies that will be discussed in the upcoming section. [67][3]

4.2.3 Pool-Based Sampling

The pool-based labeling setting consists of two pools of data which include a large set of unlabeled data and a small set of labeled ones. The instances are queried based on the usefulness to evaluate the other instances in the unlabeled pool.

Figure 4.6: Pool-based selective sampling. Image from [3].

Fig 4.6 shows the workflow of membership query synthesis. This is different from the previous one where the data is queried from the stream of data coming whereas here the data is selected from the entire data pool based on its usefulness. [67][3]

4.3 Query strategy frameworks

Passive learning approach has a large amount of labeled data which are sampled from an underlying distribution is used to train a model. Then the trained model is used for prediction. But in active learning, the learner query the most informative instance or the best instance which is the main difference between active and passive learning. There are various strategies to select the most informative query which are discussed below. [67]

4.3.1 Uncertainty Sampling

Uncertainty sampling is a straightforward yet powerful active learning query strategy. In uncertainty sampling, the active learner query the data for which the prediction probability is very uncertain to classify into a particular class. The three types of uncertainty sampling are discussed below.

Least confident uncertainty sampling

Least confident uncertainty sampling queries the instance for which the learner is not confident about its prediction. The instance that has to be queried using the least confident uncertainty sampling is found using the following equation given by [50].

$$x^*_{LC} = \underset{x}{\operatorname{argmax}}\, 1 - P_\theta(\hat{y}|x) \tag{4.1}$$

Here, $P_\theta(\hat{y}|x)$ is the prediction probability of an instance belonging to a most likely class. This helps in finding the least confident instance for which the prediction probability for a particular class is high.

For example, table 4.1 shows the prediction probability of the three instances from which one needs to be queried to the user for its label.From the table 4.1, the uncertainties of the three data points are 0.2, 0.5 and 0.6 respectively. It is clear

Instances	Class A	Class B	Class C
I	0.1	0.8	0.2
II	0.35	0.15	0.50
III	0.3	0.3	0.4

Table 4.1: Prediction probabilty of three instances with respect to three classes

that the third instance has more uncertainty than the other two. Hence the third instance will be queried to the user by the active learner. [67] [10]

Margin uncertainty sampling

Margin uncertainty sampling chooses instance considering the prediction probability of both the first and second likely class for which the instance belongs. This is in contrast to the least confident uncertainty sampling where it considers only the class for which the instance has maximum prediction probability. Scheffer et al.[66], computes the margin uncertainty of the instances using,

$$x_M^* = \underset{x}{\operatorname{argmin}}\, P_\theta(\hat{y_1}|x) - P_\theta(\hat{y_2}|x) \tag{4.2}$$

where, $P_\theta(\hat{y_1}|x)$ and $P_\theta(\hat{y_2}|x)$ were the prediction probability of the first and second likely class of a particular instance respectively. The instance for which the margin uncertainty is minimum is queried.

For example, let Table 4.1 shows the prediction probability of the three instances from which one need to be queried to the user for its label using margin uncertainty sampling. Margin uncertainty of first, second and third instance computed using the above equation are 0.6,0.15 and 0.1. Clearly, the third instance has less margin uncertainty which will be queried next.[67] [10]

Entropy uncertainty sampling

Entropy uncertainty sampling helps in overcoming the deficit of margin uncertainty sampling in cases where the number of classes is more. This is because margin uncertainty sampling considers only the first and the second most probable class.Entropy can be found using the below equation given by [70].

$$x_H^* = \underset{x}{\text{argmax}} - \sum_i P_\theta(y_i|x)\ log(\ P_\theta(y_i|x)) \tag{4.3}$$

where $P_\theta(y_i|x)$ is prediction probability of instance x belonging to i^{th} class. The instance for which the class probabilities closely represent a uniform distribution has more chance of having a higher entropy and being queried by the learner. [67]

4.3.2 Query-By-Committee

Query-based committee algorithm has a committee which has a bunch of models. These models are trained on the available labeled data. Every model has its hypothesis. The central idea here is to select the instance in which there is more disagreement on the predictions among the models in the committee. This is done to bind the version space(all available set of hypotheses that are consistent with the labeled data). Query-based committee algorithm achieves this by selecting the instances in the disputed region. [67]

The disagreement among the models in the committee is measured mainly using the following three methods [67].

Vote Entropy

Vote entropy was proposed by [31]. The vote entropy of each instance is found by using the below formula:

$$x_{VE}^* = \underset{x}{\text{argmin}} \sum_i \frac{V(y_i)}{C} log \frac{V(y_i)}{C} \tag{4.4}$$

in which 'i' refers to classes, '$V(y_i)$' refers to the count of votes that particular class achieves, and 'C' refers to the number of models in the committee. The instance having the highest vote entropy value is queried [31] [67]. This can be explained with an example based on [10].Assume that there are three models, three classes(1,2 and 3) and five instances. Let the following table 4.2 show the predicted label by each model.

Now the task is to select the instance with high vote entropy. To find the vote entropy, probability distributions of each class have to be calculated first. For the

Instances	Model 1	Model 2	Model 3
I	1	1	1
II	2	2	1
III	3	1	1
IV	1	2	3
V	2	2	1

Table 4.2: Predicted labels by the three models.

first instance, all the models predict 1. Hence the distribution for Class 1 is 1 and 0 for other classes. In the second instance, class 2 is selected twice. Therefore the distribution for Class 2 is 0.6667 (2/3) and for Class 1 is 0.3333(1/3). Similarly, the probability distribution for every class in other instances is found. The table 4.3 shows the probability distribution of each class.

Instances	Class 1	Class 2	Class 3
I	1	0	0
II	0.3333	0.6667	0
III	0.6667	0	0.3333
IV	0.3333	0.3333	0.3333
V	0.3333	0.6667	0

Table 4.3: Probability distribution on each class.

Then the entropy for each instances is found and are displayed in table 4.4.

Instances	Entropy
I	0
II	0.6365
III	0.6365
IV	1.0986
V	0.6365

Table 4.4: Entropy value for each instances.

From the table 4.4, it is very clear that the fourth instance has more vote entropy value. Hence the fourth instance will be queried. [67]

Consensus entropy

Consensus entropy differs from the previous method by calculating the entropy values based on the consensus probabilities. The consensus entropy is calculated based on [67] as,

$$P_c(y_i|x) = \frac{1}{C}\sum_{c=1}^{C} P_{\theta(c)}(y_i|x) \tag{4.5}$$

where C refers to the set of models in the committee, x is the instance and y_i is the predicted class. The above consensus probability is found by computing the mean of the class probabilities of each model in every instance.This can be explained with an example which is based on [10]. Assume that there are four models, three classes(1,2 and 3) and five instances. The table 4.5 shows the class probabilities by every model in first instance. Table 4.6 shows the calculated consensus probability.

Model	**Class 1**	**Class 2**	**Class 3**
Model 1	0.6	0.2	0.2
Model 2	0.5	0.3	0.2
Model 3	0.55	0.35	0.1
Model 4	0.1	0.5	0.4

Table 4.5: Class probabilities by every model in first instance.

Then the entropy value for instance will be 1.06. Similarly the table 4.7 shows the

Class 1	**Class 2**	**Class 3**
0.44	0.34	0.22

Table 4.6: Consensus probability for each class of the first instance.

class probabilities by every model in second instance. Table 4.8 shows the calculated

Model	**Class 1**	**Class 2**	**Class 3**
Model 1	0.3	0.2	0.5
Model 2	0.3	0.5	0.2
Model 3	0.35	0.15	0.5
Model 4	0.2	0.3	0.5

Table 4.7: Class probabilities by every model in second instance.

Class 1	Class 2	Class 3
0.29	0.29	0.42

Table 4.8: Consensus probability for each class of the second instance.

consensus probability. Then the entropy value for instance will be 1.08. Here the second instance is selected since it has more consensus value. [67]

Max disagreement

Max disagreement method was proposed by [54] based on KL divergence. KL divergence is used to measure the difference between the probability distributions [47]. The below equation for each instance helps in determining the KL divergence of each model to the consensus probability distribution according to [47].

$$x^*_{KL} = \underset{x}{\operatorname{argmax}} \frac{1}{C} \sum_{c=1}^{C} D(P_{\theta^{(c)}} | P_C), \tag{4.6}$$

where

$$D(P_{\theta^{(c)}} | P_C) = \sum_i P_{\theta^{(c)}}(\frac{y_i}{x}) log \frac{P_{\theta^{(c)}}(\frac{y_i}{x})}{P_C(\frac{y_i}{x})} \tag{4.7}$$

Here θ^C denotes the parameters learned by models in the committee and $P_c(y_i|x) = \frac{1}{C}\sum_{c=1}^{C} P_{\theta^{(c)}}(y_i|x)$ denotes the consensus probability for y_i to be the right class. The instance having the high KL divergence value is queried. [67]

4.3.3 Expected Model Change

Expected model change query strategy uses an approach called as expected gradient length (EGL) which was proposed by [69].This approach can be used wherever the model is trained based on gradient descent. This type of query strategy chooses the instance which after labeling through querying can increase the gradient length. Gradient length refers to the vector that is used to update the parameters. The expected length is calculated using the following equation based upon the models current belief according to [67].

$$x^*_{EGL} = \underset{x}{\operatorname{argmax}} \sum_i P_\theta(\frac{y_i}{x}) ||\nabla l_\theta(L \cup < x, y_i >)|| \tag{4.8}$$

where $\nabla l_\theta(L \cup < x, y_i >)$ refers to the gradient vector that been found by including the queried instance 'x' with the class 'y_i' to the trained labeled data 'L'. [67]

4.3.4 Expected Error Reduction

Expected Error Reduction query strategy in contrast to the previous query strategy(Expected model change query strategy) query the instance for which after labeling reduces the generalization error. In this approach, the forthcoming error of the model is estimated based on the current belief and the instance which reduces the error more than others is queried. The expected error change for all class possibility is estimated based on the model learned so far, since the true class for the queried label is unavailable during the estimation of the future error. The expected 0/1 loss is found using the below formula according to [67].

$$x^*_{0/1} = \underset{x}{\operatorname{argmin}} \sum_i P_\theta(\frac{y_i}{x}) (\sum_{u=1}^{U} 1 - P_{\theta^{+<x,y_i>}}(\hat{y}|x^{(u)})) \tag{4.9}$$

where U refers to the set of instances on the unseen pool and $P_{\theta^{+<x,y_i>}}$ refers to the estimated probability of the instance based on the model trained after adding $< x, y_i >$ in the trained labeled dataset. The instance which minimizes the expected 0/1 loss is queried next. Instead of the expected 0/1 loss, the expected log loss can also be found and can be used in the selection of instance that can be queried. Likewise this method is not only constrained to minimization of the loss but also can be extended to other evaluation metrics such as maximizing precision, recall, f1 score etc., The computation cost for the expected error reduction is very high because it retrains the model for all the unseen instances to find the instance to be queried. Also it retrains the model after the query is labeled. O(LUG) is the time complexity where L is the number of trained labeled dataset available so far, U is the number of unseen data in the pool, and G is the number of computation required to find the gradients. [67]

4.3.5 Variance Reduction

Variance reduction helps in overcoming the shortcoming of the expected error reduction query strategy where the computation cost is very high. This is achieved by decomposing the expected error based on the approach given by [39] as

$$E_T[(\hat{y} - y)^2|x] = E[(y - E[y|x])^2] + (E_L[\hat{y}] - E[y|x])^2 + E_L[(\hat{y} - E_L[\hat{y}])^2] \quad (4.10)$$

where E_L refers to the expectation on the already available labeled set, E refers to the expectation on the probability of the true label given an instance and E_T combine both the previous expectation, $E[(y - E[y|x])^2]$is the variance which does not depend in the training data, $(E_L[\hat{y}] - E[y|x])^2$ is the bias and does not affect the generalization error. Hence only $E_L[(\hat{y} - E_L[\hat{y}])^2]$ can affect the generalisation error of the model as this is the variance of the model. Here the models variance refers to the models mean squared loss.The variance reduction query strategy can be implemented by finding the value of x^*_{VR} for all the unseen instances using the below formula and the instance which reduces the variance is queried to the user.

$$x^*_{VR} = \underset{x}{\operatorname{argmin}} < \widetilde{\sigma}^2_y >^{+x} \quad (4.11)$$

where $< \widetilde{\sigma}^2_y >^{+x}$ is the estimated mean output variance. The instance which reduces this variance also reduces the generalization error. [67] [29] [30]

4.3.6 Density-Weighted Methods

Density weighted method while querying not only considers the data in the uncertain region but also the input spaces dense region. This is done to avoid querying the instances that are outliers. Uncertainty based sampling and Query by committee methods suffers from this problem. Sometimes they query the outliers. This can be illustrated from the figure 4.7 shown below.

In the figure 4.7, the triangle and square represent the data for which the classes are known already. The circle represents the data in the dense pool. Now the uncertainty based sampling will query the data point A since it lies close to the

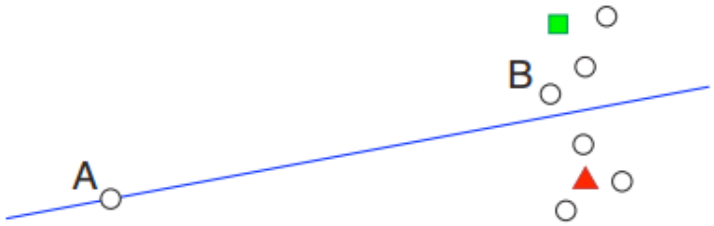


Figure 4.7: Outlier point A will get queried instead of point B since it lies very close to the uncertain region. Image from [67]

margin. However, point B is the most useful query in this situation. This problem can be avoided if the instances are queried in the dense region.

Instance are queried based on the below formula given by [68],

$$x^*_{ID} = \underset{x}{\operatorname{argmax}}\, \phi_A(x) * (\frac{1}{U}\sum_{u=1}^{U} sim(x, x^{(u)}))^{\beta} \tag{4.12}$$

where $\phi_A(x)$ refers to the informativeness based on any other query strategy and $(\frac{1}{U}\sum_{u=1}^{U} sim(x, x^{(u)}))^{\beta}$ refers to the weight of the informativeness. Sim here refers to the similarity measures like cosine distance, Euclidean distance or KL divergence.The computation cost increases quadratically with the number of unseen instances. This can be optimized by having a look-up table or precomputed data. [68] [67]

5

Experimental Setup

5.1 Libraries and Toolkits

In order to do seamless experiments, a lot of libraries and toolkits came in handy while parsing, preprocessing the text, and extracting the features. These libraries include functions which would do the most repetitive tasks in NLP that would require significant time to code from scratch. The open source libraries which were used in our experiments include Numpy, Pandas, NLTK, Spacy, and scikit-learn. These are discussed below.

5.1.1 Numpy

Numpy is a handy library in Python which is used to work with multidimensional arrays and matrices along with some high-level mathematical functions to operate on these arrays. In addition, it has functions with linear algebra, Fourier transform, and random number capabilities [15].

5.1.2 Pandas

Pandas provides high-end, easy-to-use data structure and data analysis tools written in Python programming language [16]. It was helpful in filtering out the required fields from the dataset and visualize the results of preprocessing steps in a structured manner.

5.1.3 NLTK - Natural Language Toolkit

Natural Language Toolkit is a platform for working on human language using Python. It is open-source and one of the most versatile libraries available today. It has a collection of huge corpora and a large set of lexical database. In addition, it has functionalities for various NLP tasks including classification, tokenization, stemming, tagging, parsing, and semantic reasoning [13].

5.1.4 spaCy

spaCy is a very famous open-source libary which is extensively used in various applications of Natural Language Processing written in the programming languages such as Python and Cython. It is more suitable for large-scale information extraction tasks as it is fast and efficient, with seamless interoperation with most of the deep learning libraries and frameworks. Non-destructive tokenization, Named entity recognition, support for 31+ languages, 13 statistical models for 8 languages, Pre-trained word vectors, and Part-of-speech tagging are some of its features. Opposed to NLTK, which is widely used for teaching and research, spaCy is focused more on providing software for production usage [19].

5.1.5 ModAL

ModAL [32] is a Python3 based framework for active learning which acts as a wrapper over scikit-learn library. It allows easy incorporating of active learning query strategies such as uncertainty based ones and committee based sampling with other functionalities to change the batch size. In addition, it allows to write custom query strategies which was used to perform random sampling in the experiments in this project.

5.1.6 scikit-learn

scikit-learn is widely used for data mining and data analysis tasks as it is very efficient and simple to use. It is built over Scipy, Numpy and matplotlib. It

is an open-source library written in Python programming language. The most prominent algorithms built in this library include classification, regression, clustering, dimensionality reduction, model selection, and preprocessing. We used scikit-learn to employ different machine learning algorithms which would suit best for our task such as logistic regression, naive Bayes classification, and random forest [61].

5.2 Datasets

5.2.1 Mohler'11 Dataset

This dataset was created at the University of North Texas by Rada Mihalcea and Michael Mohler. It consists of the answers for assignments and exams written by undergraduate students of the Computer Science course. The answers were for 10 assignments consisting of 4 to 7 questions each and 2 exams consisting of 10 questions each. The assignments were graded in the range of 0 to 5 whereas the exams were graded from 0 to 10. The exam grades were normalized to 0 to 5 scale for the purpose of using autograding algorithms on them. In total, there were 2273 answers after removing the answers of the questions which were not short-answer type questions and were of selection/ordering type. One of such questions which was removed is "Order the following functions by their running time. n^2, $log(log(n))$, $n!$, n^3". Fig 5.1 shows the grade distribution of all the answers. From this figure, we can clearly determine that the grade distribution is skewed towards the higher grades [57].

The grades for each answer were given by two graders, and the average of them was provided too. Thus, there were three grades for each and every answer. The average score was used as the ground truth for the experiments conducted with this dataset. The analysis on the grades given by the two annotators revealed that there is a significant percent of disagreement between them. This is illustrated in Fig 5.2.

The Fig 5.2 shows the percentage of answers with respect to the difference in the grades given by each annotator. For e.g., the percentage of answers which were graded by the two annotators with a difference of 3 is 5.5%. Only 57.7% of the answers were given the same grades [57].

This dataset is chosen for the experiments in this project as it is relatively a big dataset in the ASAG research community. This dataset is often used to benchmark

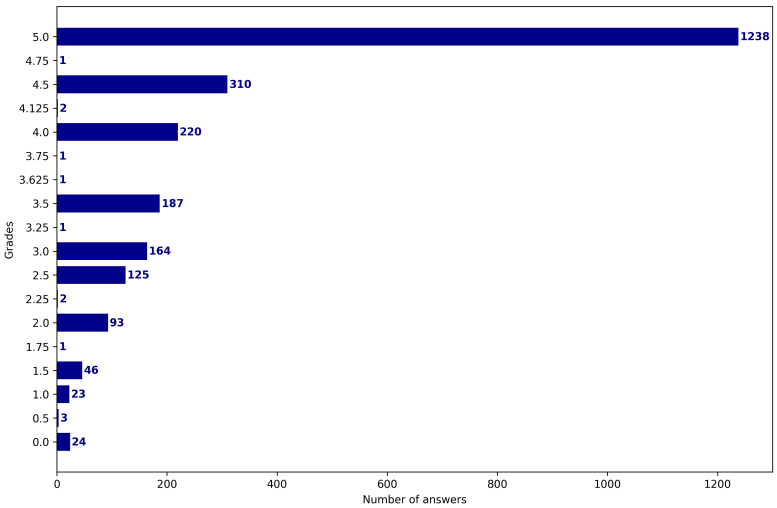


Figure 5.1: Grade distribution of Mohler'11 dataset

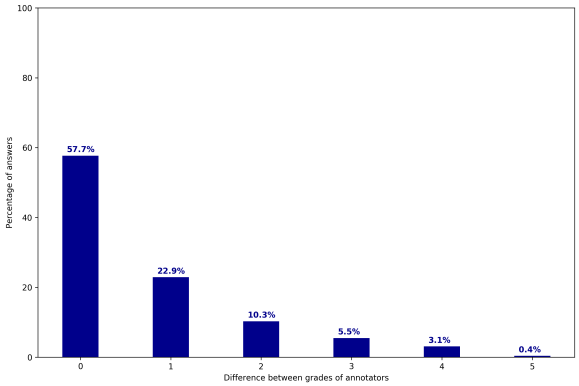


Figure 5.2: Inter-annotator grade analysis [57]

many short answer grading systems such as [73] [57] [63].

5.2.2 Neural Network dataset

This is an in-house dataset collected by the Professor and the Teaching Assistant of the Neural Network course offered at Hochschule Bonn-Rhein-Sieg University of Applied Sciences, Sankt Augustin. It consists of 663 answers for 17 questions written by 39 students for the final exam in Jupyter notebooks. Out of these, the answers written in the German language was removed before being used in our experiments as the algorithms were based on English corpus. This resulted in a dataset of 646

answers written by 38 students. Every answer was graded on a scale of 0 to 2. The grade distribution of this dataset is shown in Fig 5.3. The grade distribution in this dataset shows a skew towards the correct answers, but not as much as in the Mohler'11 dataset.

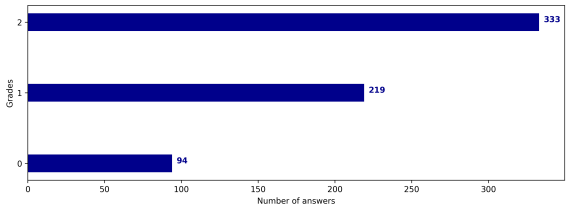


Figure 5.3: Grade distribution of the Neural Network dataset

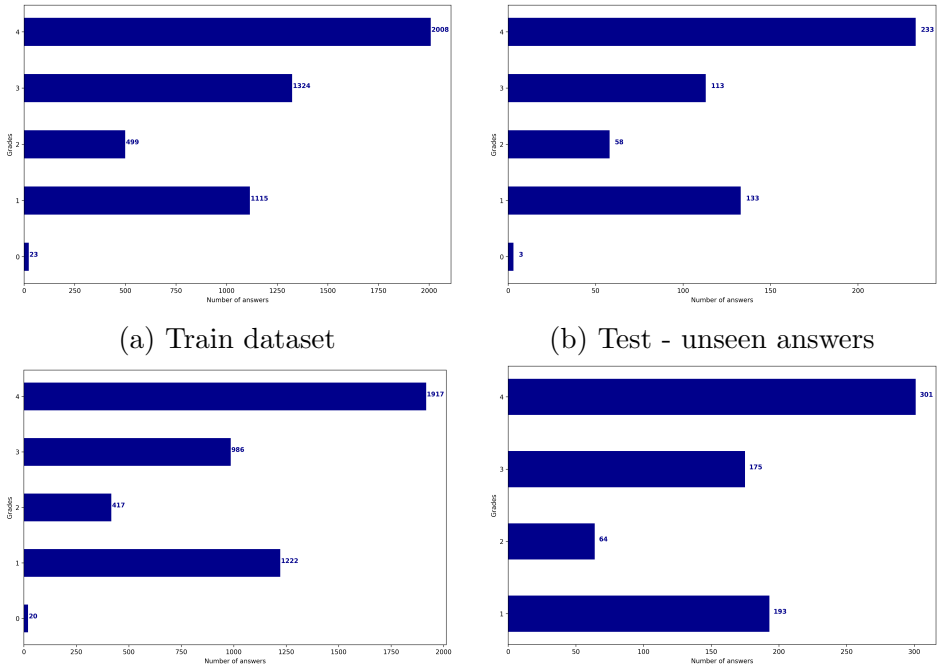
The overall objective of this research is to utilize the autograding models for in-house assignments and examinations. Hence, the answers of students for the Neural Networks course were included as a dataset for this task.

5.2.3 SemEval-2013 Task 7

The SemEval-2013 Task 7 consists of two datasets, namely, Beetle dataset and Science Entailment Corpus (SciEntsBank) dataset. The latter one is used in the experiments of this project. SciEntsBank dataset is a collection of student answers to Science assessment questions compiled by Nielson et al. 2008 [59]. This consists of 10804 question answer pairs which are divided into four sections. The first section consists of 4969 answers which are meant to be used as training set. The second section consists of 540 answers which are unseen when compared to the training set. The third section of answers consists of 4562 answers which does not have any common questions with the training set. The fourth section consists of answers which belongs to a totally different domain than the ones in the other three sections. In contrast to the datasets mentioned before, this dataset has five classes, namely, correct, partially correct incomplete, contradictory, irrelevant, and unknown domain [35].

In this project, we mapped these classes into numeric grades as;

{correct:0, partially correct incomplete:1, contradictory:2, irrelevant:3, and unknown domain:4}



(a) Train dataset

(b) Test - unseen answers

(c) Test - unseen domains

(d) Test - unseen questions

Figure 5.4: Grade distribution of SciEntsBank dataset [35].

The class distribution of these four subsections of this dataset are shown in Fig 5.4.

One of the unique features of this dataset is that it belongs to a different domain than the first two dataset. In addition, the fact that the test dataset has been explicitly divided into unseen answers, unseen questions and unseen domains which facilitates evaluating the generalization ability of the trained model. Recent researches done in ASAG have used this dataset for their evaluations [51] [48].

5.3 Feature Extraction

Feature extraction deals with getting the relavant features from the raw dataset. The first task involves bringing all the raw data into a common format. In this research work, all the raw data were converted into a comma separated values (csv) file format so that it could be fed into a pandas dataframe. Five main features were extracted from the dataset consisting of questions, student answers, reference answers and their grades in order to feed them into our machine learning models. These

features were stored as columns in the dataframe which facilitates using different collection of them while training a machine learning model. Having the data in a pandas dataframe enables faster processing of the data and efficient extraction of features from them when compared to explicit looping through each and every data sample. In addition, the machine learning library used in this project supports objects of type dataframe. Cleaning each of the dataset and the process of converting it into a pandas dataframe format is discussed below.

Mohler'11 Dataset - This dataset was downloaded from http://web.eecs.umich.edu/~mihalcea/downloads/ShortAnswerGrading_v2.0.zip, which was available in a text file (.txt) format. The student answers for each question, reference answers and questions were in separate files. A script was written to collect the questions, their corresponding student and reference answers. Later they were converted into a csv file which is stored as a pandas dataframe. Each row of the csv file consists of the question id, question, student's answer, reference answer and the grade. The first 5 lines of this dataframe is shown in Fig 5.5.

	question_id	question	ref_answer	student_answer	grade
0	1.1	What is the role of a prototype program in pro...	To simulate the behaviour of portions of the d...	High risk problems are address in the prototyp...	3.5
1	1.1	What is the role of a prototype program in pro...	To simulate the behaviour of portions of the d...	To simulate portions of the desired final prod...	5.0
2	1.1	What is the role of a prototype program in pro...	To simulate the behaviour of portions of the d...	A prototype program simulates the behaviors of...	4.0
3	1.1	What is the role of a prototype program in pro...	To simulate the behaviour of portions of the d...	Defined in the Specification phase a prototype...	5.0
4	1.1	What is the role of a prototype program in pro...	To simulate the behaviour of portions of the d...	It is used to let the users have a first idea ...	3.0

Figure 5.5: Dataframe of Mohler'11 dataset.

The features were extracted from this dataframe.

Neural Network dataset - The student answers from the final semester exam of Neural Networks course in Hochschule Bonn-Rhein-Sieg were collected by the Professor and the teaching assistant of the course. Each student wrote his/her answers in a Jupyter Notebook file as shown in Fig 5.6.

The grades were entered in a separate spreadsheet file. The reference answers were constructed from the keywords provided by the Professor for the manual grading purpose. A script was written to parse these notebook files, spreadsheet of keyword answers, and the spreadsheet of grades into a csv file format similar to the Mohler'11

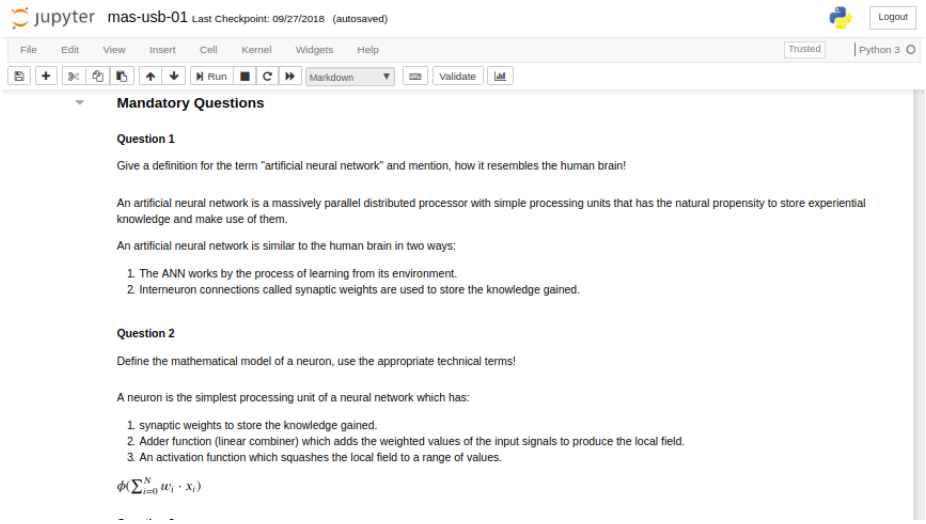


Figure 5.6: Example of a student submission in jupyter notebook.

dataset. The first 5 lines of the dataframe constructed from this csv file is shown in Fig 5.7. The features were extracted from this dataframe.

	question	student_answer	grades_round
0	Give a definition for the term "artificial ne...	An artificial neural network is a massively pa...	2
1	Give a definition for the term "artificial ne...	Artificial neural network consists of: . Large...	2
2	Give a definition for the term "artificial ne...	An artificial neural network is a massive dist...	1
3	Give a definition for the term "artificial ne...	An ANN is a layered graphical model containing...	2
4	Give a definition for the term "artificial ne...	Artificial Neural Networks are large parallel ...	2

Figure 5.7: Dataframe of Neural Network dataset.

SemEval-2013 Task 7 - This dataset was from a textual entailment challenge which has questions and student answers in different domains. These files were found in a eXtensible markup language (xml) file format. Each file contains a questions, its reference answer, the students' answers and their corresponding grades as shown in Fig 5.8.

These xml files were converted into a csv file with the help of a script file. The first 5 lines of the dataframe constructed from this csv file similar to the Mohler'11 dataset as shown in Fig 5.9. The features were extracted from this dataframe.

```
- <question id="EM_45b" module="EM">
  - <questionText>
      You used several methods to separate and identify the substances in mock rocks. How did you separate the salt from the water?
    </questionText>
  - <referenceAnswers>
      <referenceAnswer id="EM_45b-a1">The water was evaporated, leaving the salt.</referenceAnswer>
    </referenceAnswers>
  - <studentAnswers>
      <studentAnswer id="EM.45b.110.1" accuracy="irrelevant">By letting it sit in a dish for a day.</studentAnswer>
    - <studentAnswer id="EM.45b.113.1" accuracy="correct">
        Let the water evaporate and the salt is left behind.
      </studentAnswer>
      <studentAnswer id="EM.45b.114.1" accuracy="correct">The water evaporated and left salt crystals.</studentAnswer>
    - <studentAnswer id="EM.45b.261.1" accuracy="irrelevant">
        I saw a pinkish grayish color that was blocking the water.
      </studentAnswer>
    - <studentAnswer id="EM.45b.383.1" accuracy="irrelevant">
        You have to slowly tip the vial for only the water to go.
      </studentAnswer>
```

Figure 5.8: SemEval-2013 Task 7 dataset in xml format.

	ques_id	question	reference_answer	student_answer	grades_round
0	EM_45b	You used several methods to separate and ident...	The water was evaporated, leaving the salt.	By letting it sit in a dish for a day.	1
1	EM_45b	You used several methods to separate and ident...	The water was evaporated, leaving the salt.	Let the water evaporate and the salt is left b...	4
2	EM_45b	You used several methods to separate and ident...	The water was evaporated, leaving the salt.	The water evaporated and left salt crystals.	4
3	EM_45b	You used several methods to separate and ident...	The water was evaporated, leaving the salt.	I saw a pinkish grayish color that was blockin...	1
4	EM_45b	You used several methods to separate and ident...	The water was evaporated, leaving the salt.	You have to slowly tip the vial for only the w...	1

Figure 5.9: Dataframe of SemEval-2013 Task 7 dataset.

5.3.1 Bag-of-Words(BOW)

The following preprocessing tasks were done before converting all the student answers into bag-of-words.

- Converting into lowercase
- Removing the punctuations
- Stop words removal
- Lemmatization

After performing all these tasks, the answers were brought into a common format. The bag-of-words representation of the answers were obtained using sklearn library's CountVectorizer. This technique returns a sparse matrix of count numbers after considering all the answers and building a vocabulary of tokens in them. This process is done efficiently when feeding the answers in a pandas Series format.

For e.g., tokens extracted from the first sentences of three answers from the Neural Network dataset 'artificial neural network massively parallel distributed processor', 'artificial neural network largely parallel distributed processor', and 'artificial neural network consists neurons' would be;

['artificial', 'consists', 'distributed', 'largely', 'massively', 'network', 'neural', 'neurons', 'parallel', 'processor']

The bag-of-words matrix constructed from these answers based on the tokens shown above would be as follows;

$$\begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \end{bmatrix}$$

5.3.2 Term Frequency-Inverse Document Frequency (tf-idf)

Similar to the bag-of-words model, tf-idf constructs sparse matrix for student answers based on the word frequencies of the unique tokens. Tf-idf takes the most important words into account and weights unimportant words less. Hence, the presence of keywords and subject specific terms have more chance of being explicitely represented in the tf-idf matrix. The preprocessing steps before converting the answers into tf-idf format is similar to that of bag-of-words. Here the sklearn library's TfidfVectorizer tool is used for the construction of the matrix. The tf-idf matrix for the same three answers of the Neural network dataset is shown below.

$$\begin{bmatrix} 0.30371 & 0. & 0.39108 & 0 & 0.51423 & 0.30371 & 0.30371 & 0 & 0.39108 & 0.39108 \\ 0.30371 & 0. & 0.39108 & 0.51423 & 0. & 0.30371 & 0.30371 & 0. & 0.39108 & 0.39108 \\ 0.33838 & 0.57292 & 0. & 0. & 0. & 0.33838 & 0.33838 & 0.57292 & 0. & 0. \end{bmatrix}$$

5.3.3 Features from Sultan et al., 2016 [73]

The state-of-the-art supervised learning model for short answer grading system used five different features for learning and those features are experimented in this work too. The features are mainly based on text similarity between the students' answers and the corresponding reference answers. These features in the supervised

learning model were used for a regression task whereas the features are used in an active learning setup for a classification task in this project. The five main features are as follows, and these will be explained in detail below.

- Length Ratio
- Word Alignment
- Semantic Vector Similarity
- Word Alignment (with question demoting)
- Semanctic Vector Similarity (with question demoting)

Question demoting is the task of removing words that appear in the question from the reference answer and the student answer. This is done in order to reduce the possibility of earning more grades just by repeating the words in the question [73]. Fig 5.10 illustrates how the student answers and reference answers are converted into a feature array of five columns which is then fed into a machine learning model. Every feature column is normalized using scikit-learn's MinMaxScaler before using them in the model to avoid bias towards a particular feature.

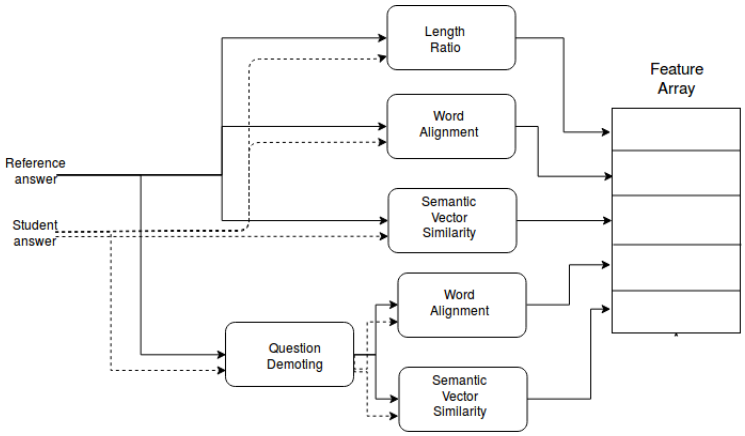


Figure 5.10: Block diagram of construction of feature array. Image adapted from [23] [73].

Length Ratio

This feature tries to combat the chances of providing high grades for students answers which are intentionally written in a long format without proper content in it. The feature value was calculated by dividing the student answer's length by the reference answer's length. For e.g., a student's answer from the Mohler'11 dataset [57], the corresponding reference answer, and the length ratio is as follows;

Reference answer: 'To simulate the behaviour of portions of the desired software product.'
Student answer: 'To simulate portions of the desired final product with a quick and easy program that does a small specific job. It is a way to help see what the problem is and how you may solve it in the final project.'
Length Ratio: 0.268 (11/41)

Word Alignment [72]

In order to capture the semantic similarity between the student answer and the reference answer, each pair of words from both the sentences are considered for scoring based on their contextual and lexical similarity the the scores of all those pairs were combined in the form of a weighted sum to calculate a final alignment measure. Paraphrase database [38] was used to measure how a pair of words are similar lexically. Dependency measures are employed to capture the contextual similarity of a pair of words. For e.g., a word in a student's answer and another word in the reference answer is processed for their dependencies in the respective sentences and the calculated score was weighted in the final calculation of the final alignment score.

The original implementation of word alignment by [72] utilized Stanford CoreNLP. A fellow research student in our E-assesment team, Ramesh Kumar implementated the same algorithm using NLTK and achieved similar results [23]. We have used Ramesh's NLTK implementation for getting the word alignment score in our datasets. For e.g., a reference answer, a student's answer in the Semeval dataset [59] and the aligned words are as shown below;

Reference answer: 'The water was evaporated, leaving the salt.'
Student answer: 'Let the water evaporate and the salt is left behind.'
Aligned words: [['evaporated', 'evaporate'], ['water', 'water'], ['salt', 'salt'], ['leaving', 'left']]

Similar alignment scores were calculated for every pair of question demoted student and reference answers and added as a separate feature in the feature array.

Semantic Vector Similarity

Cosine similarity is one of the mathematical techniques to find the similarity between two vectors. Two vectors which point in the same direction and have a small angle between them would have a high cosine similarity than two vectors which point in opposite directions. This measure could be utilized for texts as well. When the cosine similarity of two sentences (which are converted into vectors by some means) is high, it could be inferred that there is a high chance of them having the same meaning [5]. Semantic vector similarity is a score calculated based on the cosine similarity between the vectors of the student and reference answers.

In order to compute the cosine similarity, the student and the reference answers were converted into word vectors using the gensim word2vec model. Each word in a student and reference answers was converted into a 400 dimensional vector and they were summed up for each sentence. The cosine similarity was calculated between these vectors using scipy.spatial library in order to produce a score for the semantic vector similarity feature. Similar kind of vectors were calculated for question demoted versions of student and reference answers the cosine similarity between these two vectors was used as another feature in the feature array.

5.4 Machine Learning Models [64]

The machine learning models used in this project are implemented with the help of the scikit-learn library [61].

5.4.1 Logistic Regression

Logistic regression is widely used for classification problems which is most suitable for classifying linearly separable data. In a simple binary classification task, the

model calculates the weight for each sample such that;

$$P(y = 1|x) = \sum_{i=0}^{m} w_i x_i \tag{5.1}$$

where the w_i is the weight for the i^{th} feature and x is the sample. The weights are learned from the training samples by minimizing the cost function (a measure of how the predictions are off from the actual outcome) using the gradient descent method. Taking $\sum_{i=0}^{m} w_i x_i$ as z, the model calculates the probability that a given sample belongs to a particular class using the sigmoid function(logistic function) as shown below.

$$\phi(z) = \frac{1}{1 + e^{-z}} \tag{5.2}$$

The resulting value is compared against a threshold to decide whether the sample belongs to the positive or negative class. Similarly, the model works in multi-class settings by considering one class as positive and all other classes as negative (one vs. rest strategy).

5.4.2 Naive Bayes Classifier

Naive Bayes classifier calculates the probability of each sample belonging to a particular class by taking into account every feature of the sample independently (the independence assumption of each feature causes the name "naive"). From the training samples, it calculates the probability of each feature given the class ($P(x_i|C_i)$) and the class distribution of all the samples. With this information, it calculates the class for a new sample by using the Bayes' theorem as stated below.

$$P(C_i|x) = \frac{P(x|C_i)P(C_i)}{P(x)} \tag{5.3}$$

For applications where every feature could take multiple values, a multinomial Naive Bayes (generalization of binary Naive Bayes) is used. This model is known for its simple implementation and good performance with less data.

5.4.3 Random Forests

Random forests is a bag of decision trees (ensemble learning of decision trees) which makes a prediction after considering the vote of each decision tree in the ensemble. Each decision tree builds a tree with features as branching factors and splits the samples based on the decisions made by the branching factor. Every branching factor (a feature of the sample) splits the samples according to the value of that particular sample using a threshold (for ex., left if the value is lesser than the threshold or right if it's more than the threshold). The features to be considered at every branching point is decided based on how informative or useful it is to split the samples based on that feature. Information gain is one of the metrics used to find the most informative features. The features and the decision thresholds are learned from the training samples and it is applied to the test samples.

Though random forests are difficult to interpret, they are known for dealing with multiple features which could be correlated, and reduced variance.

5.4.4 Support Vector Machines

Support Vector Machines (SVM) are models which define a decision boundary such that the data are linearly separated. Fig 5.11 shows how an SVM would classify the data with binary classes. The data points which lie along the decision boundary (also known as support vectors) contribute a lot in calculating the parameters of the decision boundary. The model tries to fit the boundary such that the margin between the line and the support vectors are as large as possible so that the model generalizes well (predicts well for unseen data). The model becomes a hyperplane in higher dimensions.

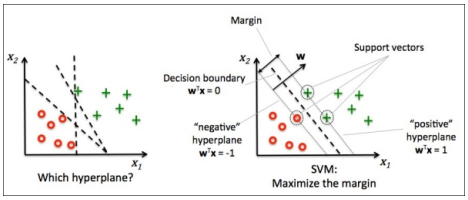


Figure 5.11: SVM - Hyperplane separating the two classes. Image from [64]

In case of nonlinearly separable data, SVM tries to project the data on to a higher dimension and try to fit a hyperplane there such that the data belonging to different classes are separated. Radial Basis Function is one of the kernel methods which is used to project the data onto a higher dimensional feature space.

5.5 Experimental Setup

Experiments were carried out in various settings in order to evaluate the performance of different features, and machine learning models, which could be used within an active learning pipeline. In addition to determining the best features to extract from the answers and the best model to learn the prediction task, the main objective of these experiments is to evaluate the active learning strategies, seed selection mechanism, and batch size during querying for the task of short answer grading.

5.5.1 Experiment Pipeline

Fig 5.12 illustrates the experimental setup and the different aspects being evaluated in this work.

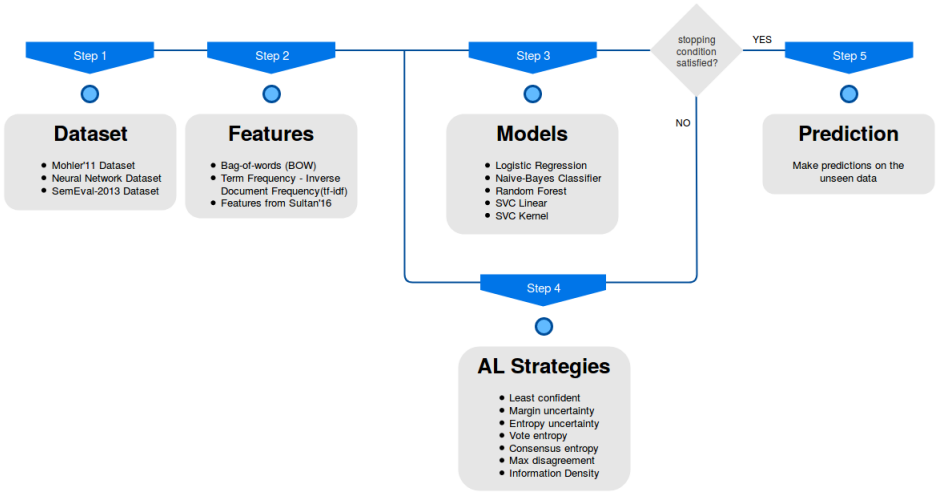
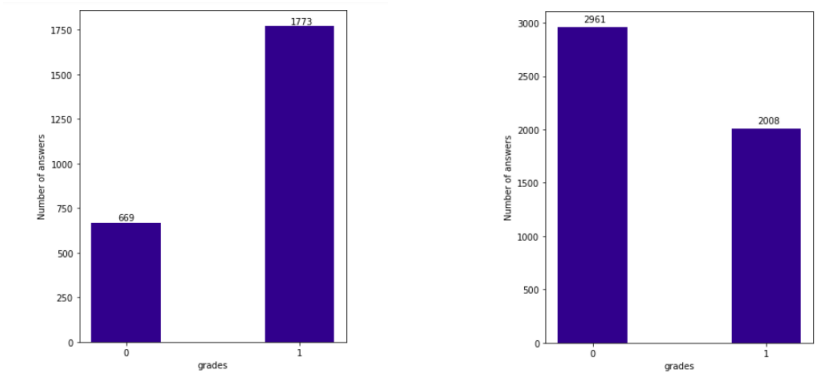


Figure 5.12: Experiment pipeline of this work.

As shown in the figure, every evaluation metric is calculated for a unique setting of features, models, and active learning strategies for every dataset. A large number



(a) Binary class distribution of Mohler dataset.

(b) Binary class distribution of SemEval 2013 dataset.

Figure 5.13: Binary class distributions.

of experiments were carried out where in every experiment, for every dataset, a feature was fixed and the performance was observed for all different models using different active learning strategies. These experiments were done for both binary and multi-class classifications.

5.5.2 Binary Classification

Experiments were done considering the task of grading student answers as a binary problem. Each answer was labeled as a correct answer or an incorrect answer based on the actual grade it was awarded. Table 5.1 shows how this task of converting the grades into a binary representation was accomplished.

Dataset	Class 1 (correct)	Class 0 (incorrect)
Mohler'11 dataset	greater than 3.5	less than or equal to 3.5
SemEval 2013 dataset	greater than 3.0	less than or equal to 3.0

Table 5.1: Coversion of grades into binary labels as correct and incorrect.

Though the grades for both the datasets were on a scale from 0 to 5, the threshold for Mohler'11 dataset was set higher than that of SemEval 2013 dataset as there was a huge inbalance between the high and low grades in it. Increasing the threshold helped in avoiding a large imbalance of the two classes.

Fig 5.13a and Fig 5.13b show the class distributions after the conversion. Binary classification task was not performed on the Neural Network dataset as it had only 3 classes and it was difficult to split them into correct and incorrect answers.

5.5.3 Multi-class Classification

As opposed to Binary classification, the real world application includes awarding grades which will not be just two numbers (0,1). Thus, multi-class classification includes experimenting the settings with multiple grades. The grades were 0,1,2 in case of Neural Network datatset, 0,1,2,3,4 in case of SemEval 2013 dataset and real numbers in the range of 0 to 5 in case of Mohler'11 dataset. We rounded off the actual grades of Mohler'11 dataset into 6 classes (0 to 5 in steps of 1.0) in order to use them in our multi-class classification experimental setting. Fig 5.14 shows the class distribution after this conversion.

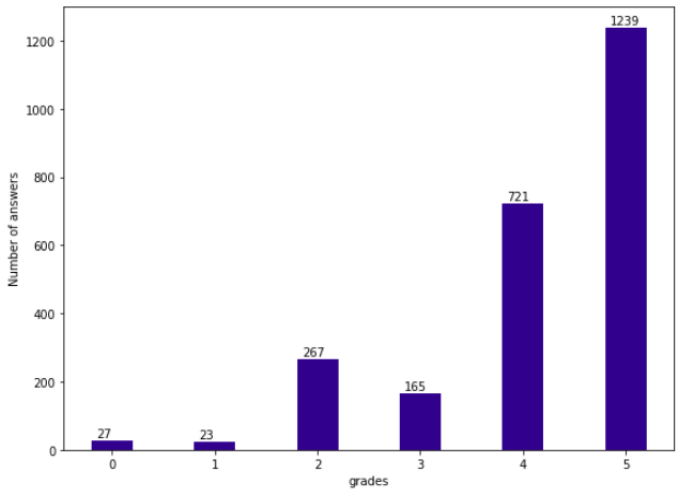


Figure 5.14: Multi-class distribution of Mohler'11 dataset.

6

Results and Evaluation

The results of experiments conducted in this work as described in the chapter 5 have been discussed here. The performance of models and different active learning query strategies have been studied using metrics such as accuracy, F1 score, kappa score and average time taken for each query. In addition, we analyze the performance of the query strategy-model combinations using a bump plot (which shows the ranking of each combination with respect to the accuracy as the number of queries increases). This unique plot enables to track the performance of any combination and helps in determining the consistent strategies and the models associated with them. Bump chart gives a more clear insight when compared to the accuracy plots which are intertwined in most of the cases.

In order to observe the efficiency of active learning, every model-query strategy combination is compared with passive learning where the query instances are sampled randomly. For every dataset, we select the best active learning setting and compare the results with that of supervised learning. Here, the supervised learning models were trained with a dataset (which was obtained by a random split) of size comparable to that of active learning setting. The plot for supervised learning was obtained by averaging the results after 10 iterations.

The graphs depict the performance of different settings with respect to the number of queries made to the human grader in order to get his/her grades. The models were trained from the queried answers and the performances were calculated based on the grades assigned to the remaining answers. For e.g., if 10% of the data were

queried from a human, the remaining 90% of the data was used to calculate the performance. From the model's performances based on the features used, accuracy and F1 scores, the best ones were selected to be viewed as a bump chart.

6.1 Experiment 1: Mohler'11 Dataset using Features from Sultan'16

This experiment was done using Sultan'16 features which were obtained as shown in Fig 5.10. The task of grading was split into binary classification and multi-class classification. The objective of binary classification is to identify correct and incorrect answers whereas multi-class classification's objective is to assign the correct grade for each answer.

6.1.1 Binary Classification

The experimental results of different model-query strategy combinations in a binary classification setting are shown in Fig 6.1 and 6.3a. As the class distribution of this dataset is uneven, accuracy alone would not be enough to judge the performance of the models. Hence, the results were also analyzed using the F1 score which are shown in Fig 6.2 and Fig 6.3b.

Based on these graphs, we observe that all the active learning query strategies outperform the passive learning. We also observe that all the active learning strategies in their respective models performed comparably to each other. Hence, we selected the margin based uncertainty sampling for each model in order to compare them using a bump chart. This chart shows the performance of every model ranked by their accuracies which is shown in Fig 6.4. The bump chart for binary classification shows that all the models compete each other for the best performance and it is impossible to chose a clear consistent winner out of these.

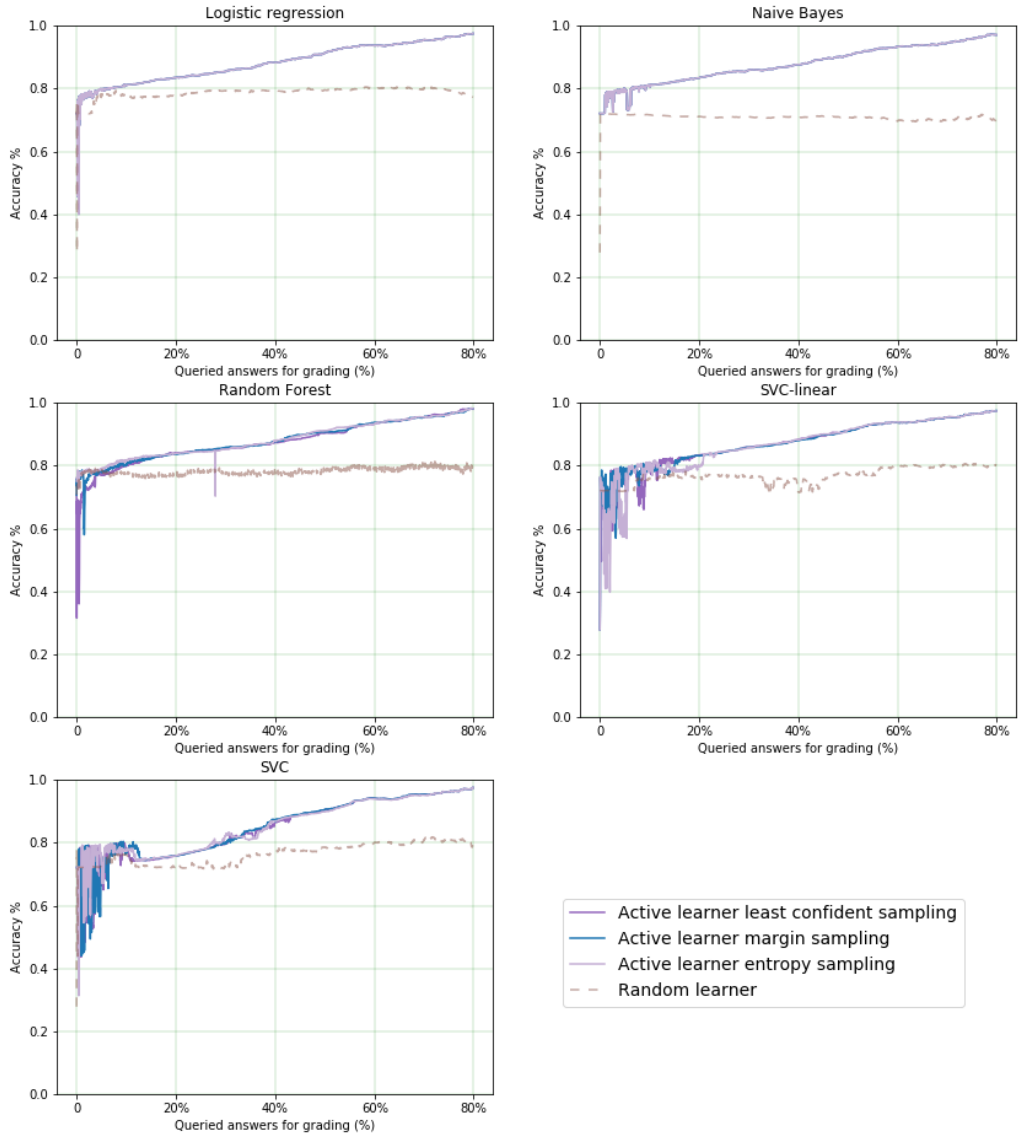


Figure 6.1: Results in terms of accuracy for different models with uncertainty based query strategies.

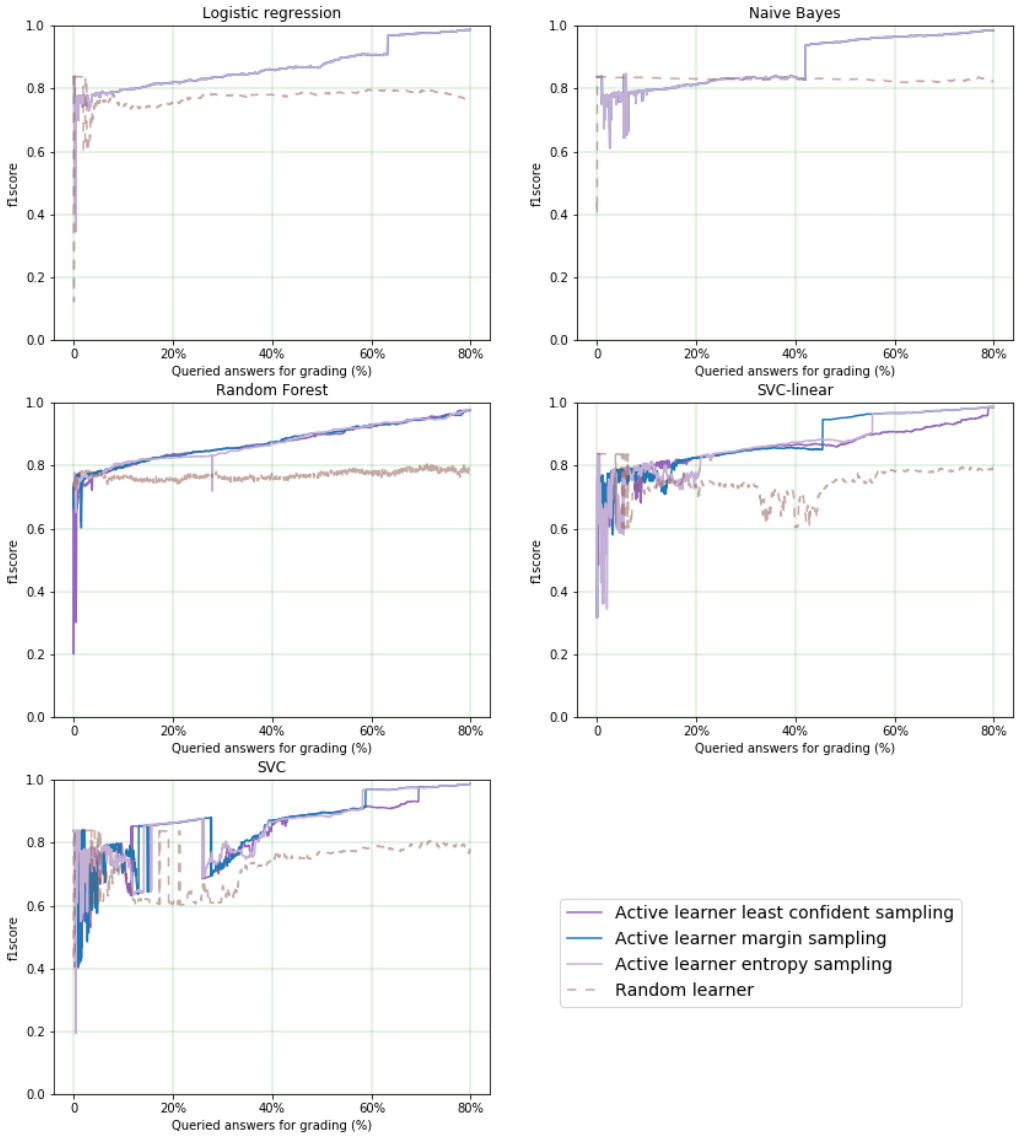
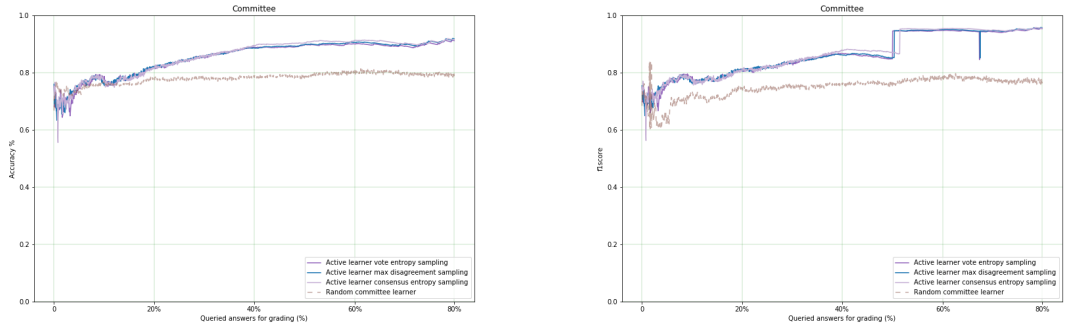


Figure 6.2: Results in terms of F1 score for different models with uncertainty based query strategies.



(a) Results in terms of accuracy for committee based query strategies.

(b) Results in terms of F1 score for committee based query strategies.

Figure 6.3: Committee-based binary classification.

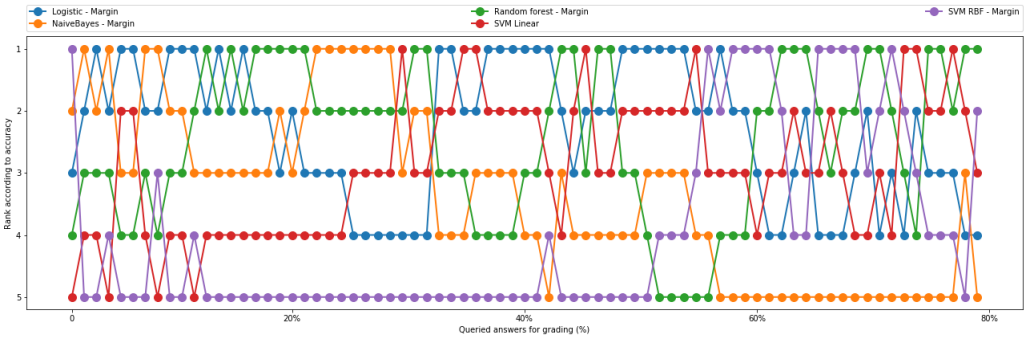


Figure 6.4: Bump chart of model performance in binary classification.

6.1.2 Multi-class Classification

The experimental results of different model-query strategy combinations in a multi-class classification setting are shown in Fig 6.5 and 6.7a. The results were also analyzed using the F1 score which are shown in Fig 6.6 and Fig 6.7b.

We observe that all the active learning query strategies outperform the passive learning (random sampling). Considering the accuracy and the F1 scores, logistic regression with margin uncertainty sampling, naive Bayes with margin uncertainty sampling, random forest with uncertainty sampling, and linear SVM with margin uncertainty sampling were selected for comparison using a bump chart. In case of committee, max disagreement sampling performed better and is added to the bump chart comparison. This chart shows the performance of every model ranked by their accuracies which is shown in Fig 6.8.

The bump chart clearly shows that the random forest classifier model with least confident uncertainty sampling is preferable over logistic regression for this setting. Random forest classifier shows consistent performance over the whole range of queries.

The time comparison of different models is tabulated in Table 6.1. Based on the bump plot and time comparison, it could be inferred that random forest with least confident uncertainty sampling is best for Mohler'11 dataset with Sultan'16 features.

	Time taken per query (secs)					
Models	**least confident uncertainty**	**margin based uncertainty**	**entropy based uncertainty**	**vote entropy**	**max disagreement**	**consensus entropy**
Logistic Regression	0.0145	0.0150	0.0151	0.4220	0.4211	0.4184
Naive Bayes	0.0059	0.0062	0.0064			
Random Forest (100 trees)	0.3216	0.3235	0.3182			
SVC - Linear	0.1341	0.1305	0.1354	-	-	-
SVC - RBF	0.3869	0.3843	0.3879	-	-	-

Table 6.1: Time comparison between different models for multi-class classification.

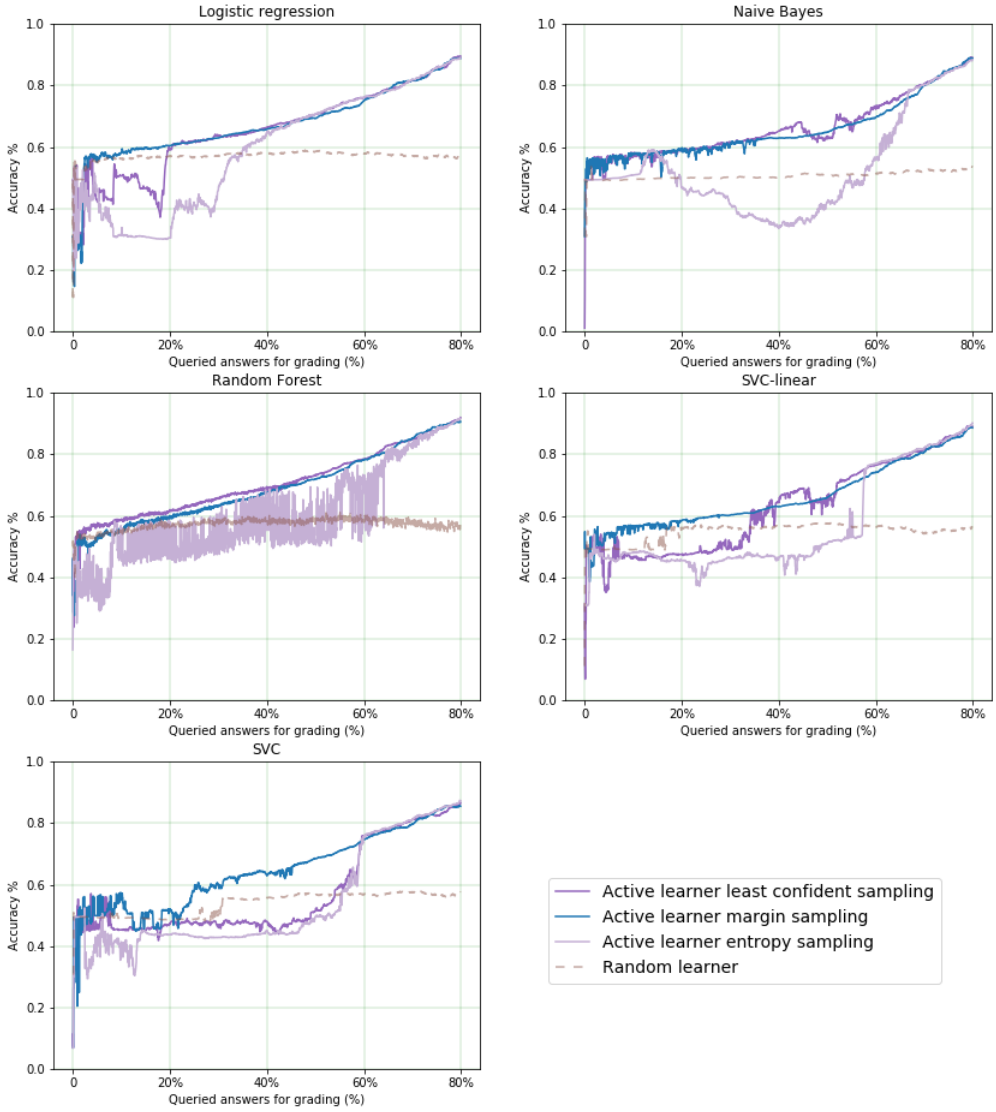


Figure 6.5: Results in terms of accuracy for different models with uncertainty based query strategies.

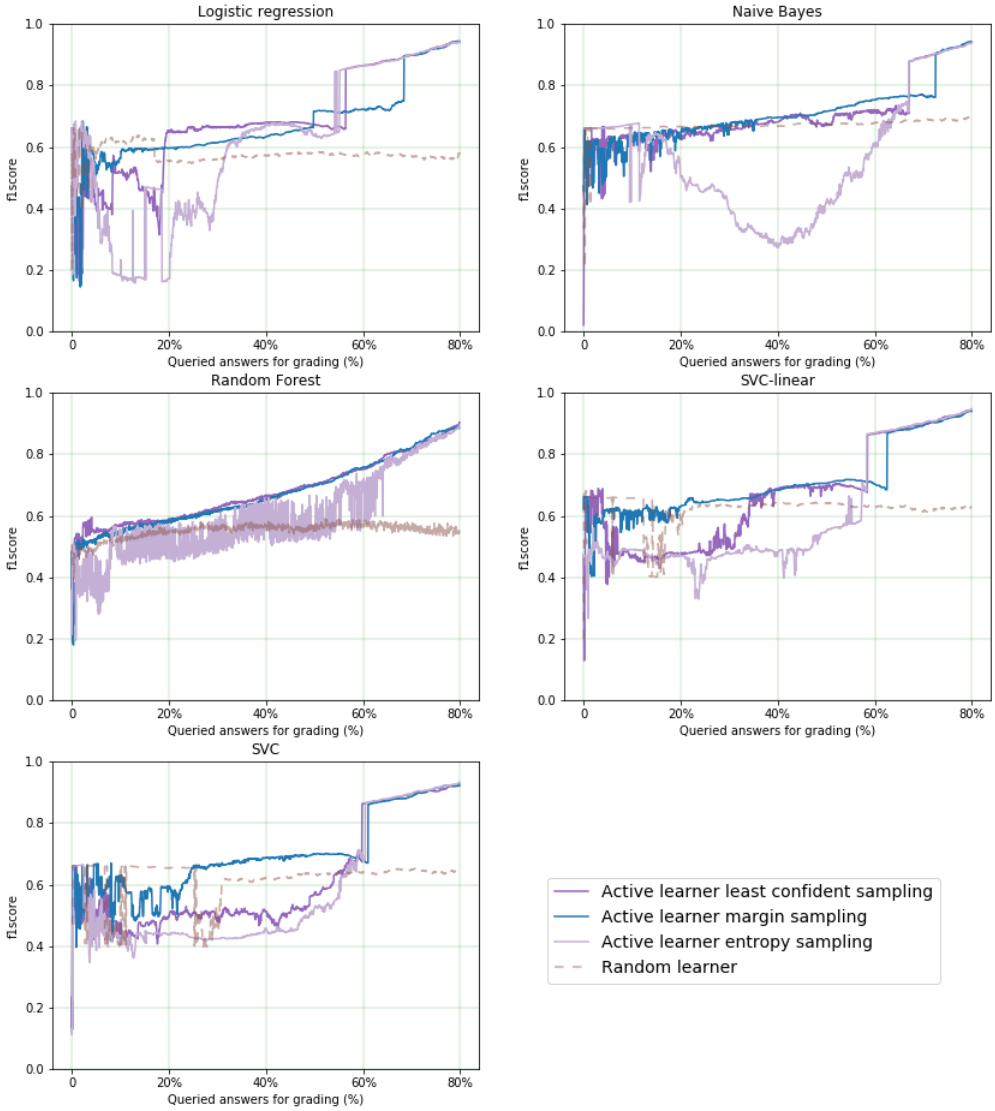
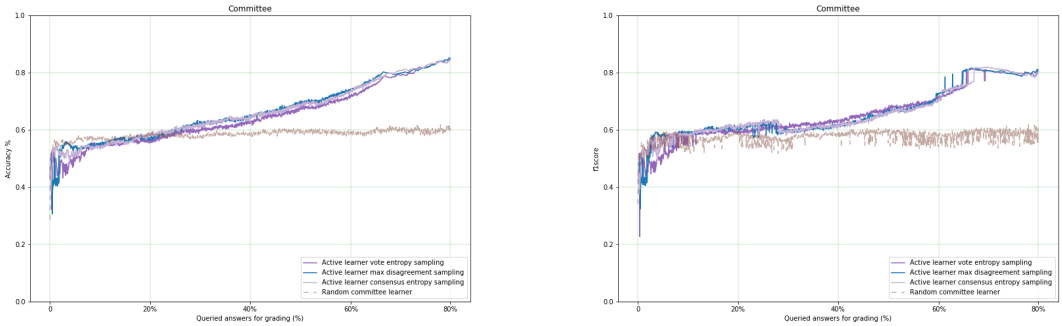


Figure 6.6: Results in terms of F1 score for different models with uncertainty based query strategies.



(a) Results in terms of accuracy for committee based query strategies.

(b) Results in terms of F1 score for committee based query strategies.

Figure 6.7: Committee-based multi-class classification.

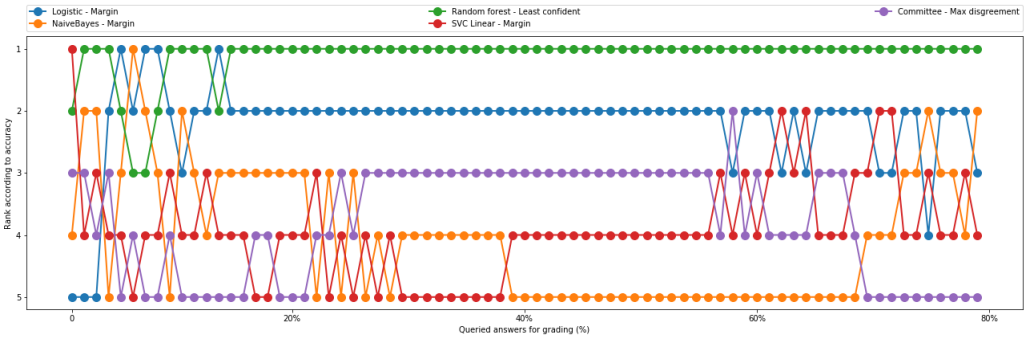


Figure 6.8: Bump chart of model performance in multi-class classification.

6.2 Experiment 2: Mohler'11 Dataset using Bag-of-words (BOW) Features

Instead of Sultan'16 features, Bag-of-words(BOW) of the answers were used as features for this experiment. Extracting this feature doesn't require reference answers of the questions as the entire BOW vectors were created from the student answers alone. Every answer was encoded into a vector of length 2138 and this vector was used as input to our models during training and prediction. The linear SVC and RBF kernel SVC models were ignored in this experiment as they consume a lot of time to select each query since the dimension of the feature space is very high. Considering the future implementation of this project as an interactive task, time becomes a key factor to be considered.

6.2.1 Binary Classification

The experimental results of different model-query strategy combinations in a binary classification setting are shown in Fig 6.9 and 6.11a. The F1 score analysis of this experiment is shown in Fig 6.10 and Fig 6.11b.

We observe that all the active learning strategies in their respective models performed comparably to each other. Hence, we selected the margin uncertainty sampling for the naive Bayes and logistic regression models, least confident uncertainty for random forest classifier, and vote entropy sampling for the committee based model to have a closer look at their performance over increasing number of queries using a bump chart. This chart shows the performance of every model ranked by their accuracies which is shown in Fig 6.12.

We can infer from the bump chart that the naive Bayes with margin uncertainty sampling performs better in the first half and random forest with least confident sampling outperforms naive Bayes in the second half. When considering the overall chart, it seems wise to select naive Bayes classifier with margin based sampling for this setting.

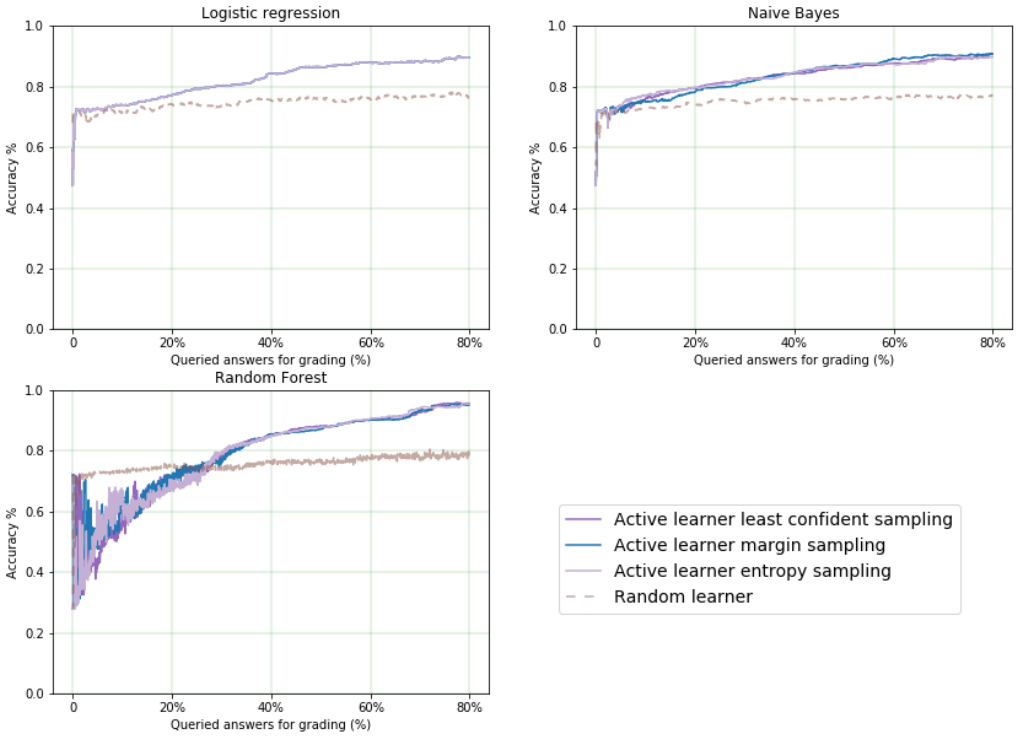


Figure 6.9: Results in terms of accuracy for different models with uncertainty based query strategies.

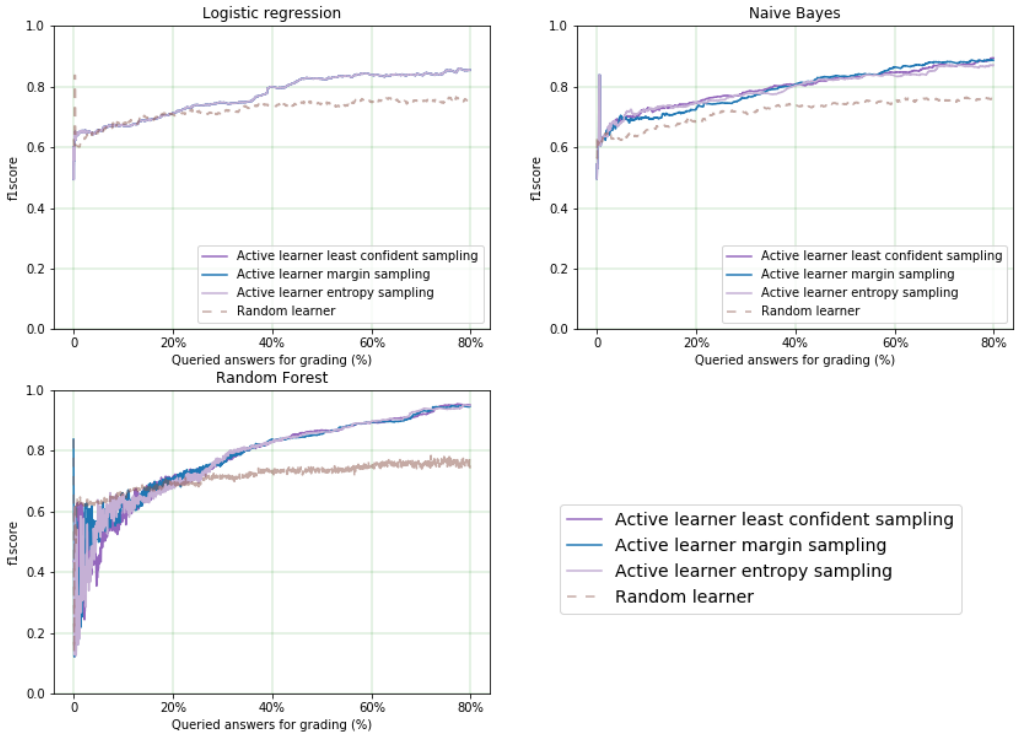
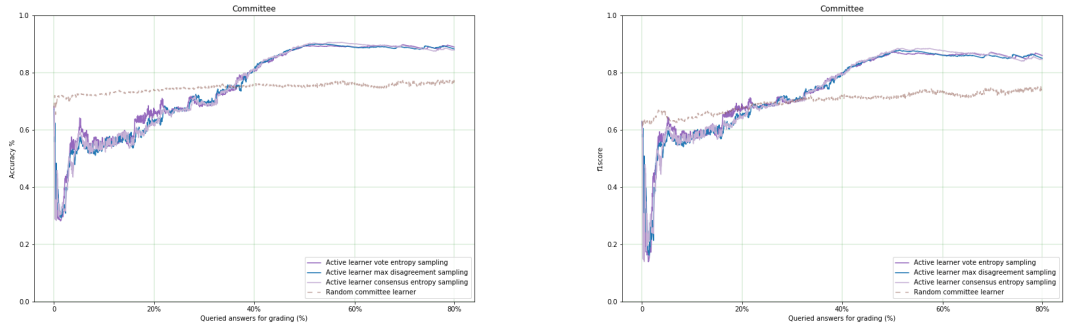


Figure 6.10: Results in terms of F1 score for different models with uncertainty based query strategies.



(a) Results in terms of accuracy for committee based query strategies.

(b) Results in terms of F1 score for committee based query strategies.

Figure 6.11: Committee-based binary classification.

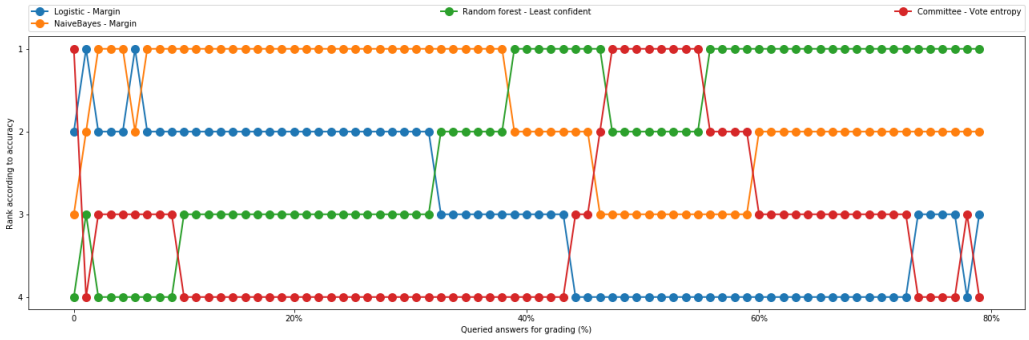


Figure 6.12: Bump chart of model performance in binary classification.

6.2.2 Multi-class Classification

The experimental results of different model-query strategy combinations in a multi-class classification setting are shown in Fig 6.13 and 6.15a. The results were also analyzed using the F1 score which are shown in Fig 6.14 and Fig 6.15b.

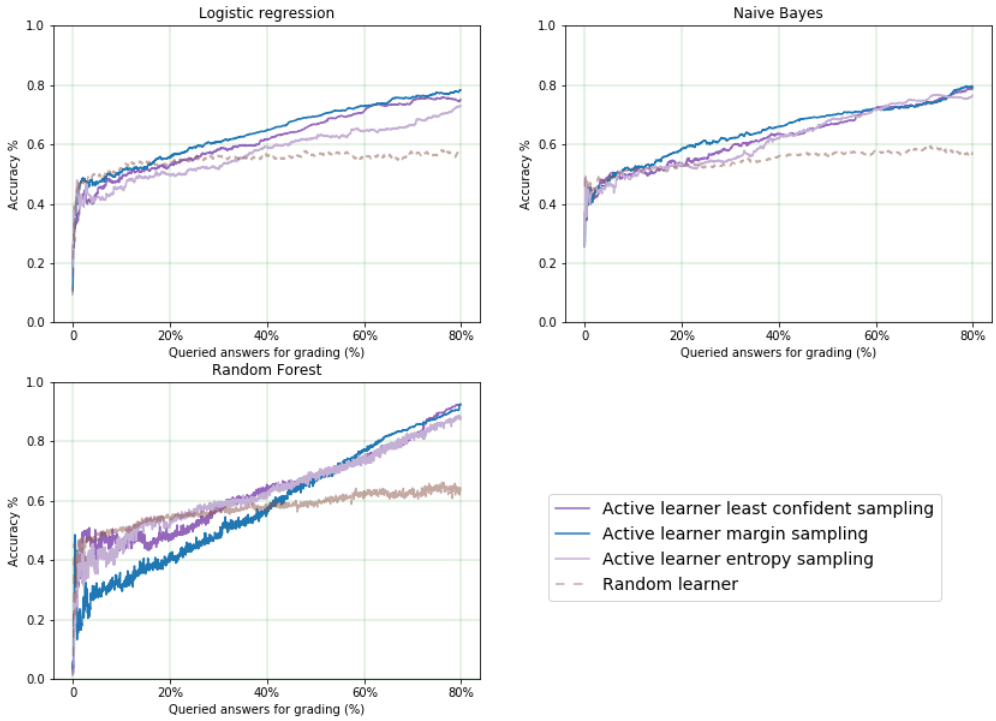


Figure 6.13: Results in terms of accuracy for different models with uncertainty based query strategies.

Considering the accuracy and the F1 scores, Logistic Regression with margin uncertainty sampling, Naive Bayes with margin uncertainty sampling, and Random Forest with least confident uncertainty sampling were selected for comparison using a bump chart. In case of committee, max disagreement sampling performed better and is added to the bump chart comparison. This chart shows the performance of every model ranked by their accuracies which is shown in Fig 6.16. The bump chart

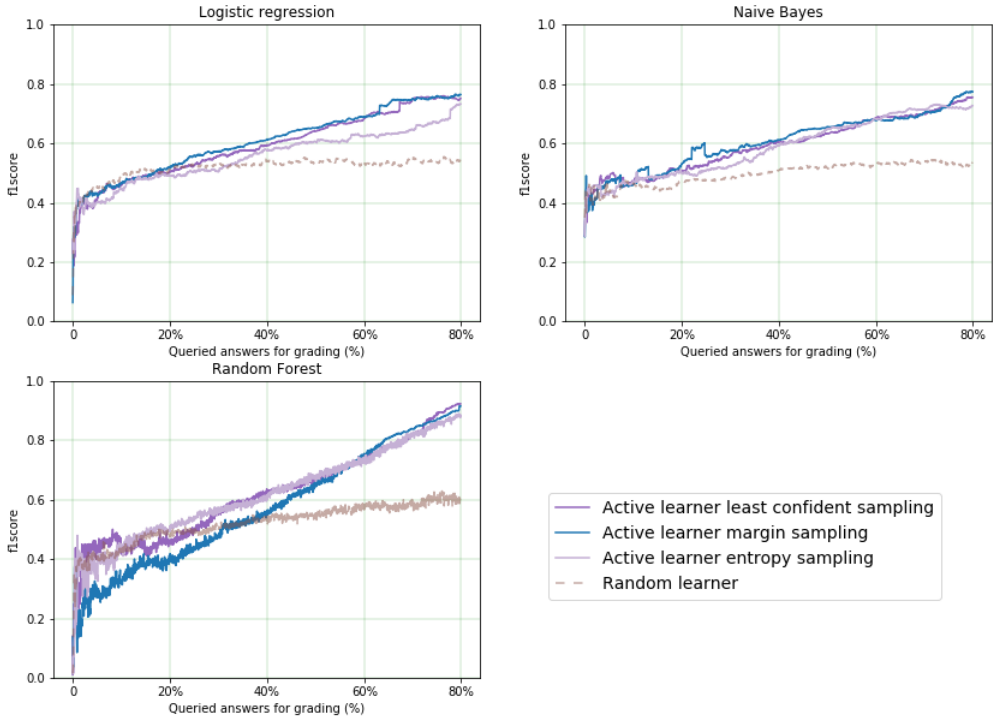


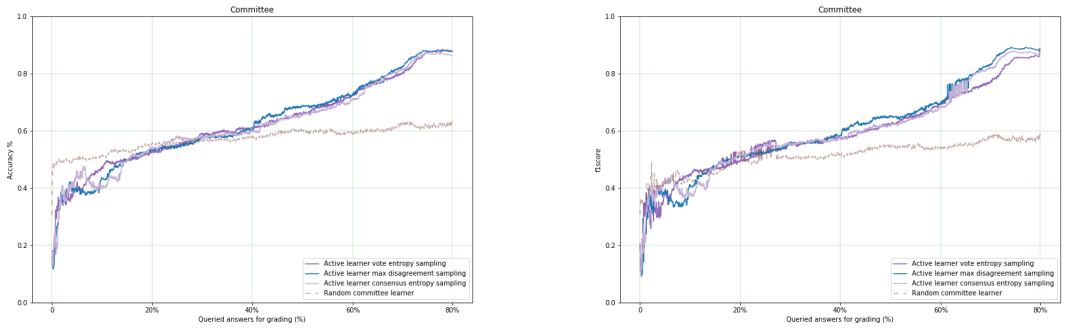
Figure 6.14: Results in terms of F1 score for different models with uncertainty based query strategies.

shows that the naive Bayes classifier performs well over a long range. When the query percentage reaches 70%, other models become comparable in performance.

The time comparison of different models is tabulated in Table 6.2. Considering the bump plot and the time comparison, it could be inferred that naive Bayes classifier with margin based uncertainty sampling fits best for this setting.

	Time taken per query (secs)					
Models	**least confident uncertainty**	**margin based uncertainty**	**entropy based uncertainty**	**vote entropy**	**max disagreement**	**consensus entropy**
Logistic Regression	0.0675	0.0720	0.0667	1.1519	1.1317	1.1239
Naive Bayes	0.0363	0.0361	0.0360			
Random Forest (100 trees)	0.9366	0.9288	0.9372			

Table 6.2: Time comparison between different models for multi-class classification.



(a) Results in terms of accuracy for committee based query strategies.

(b) Results in terms of F1 score for committee based query strategies.

Figure 6.15: Committee-based multi-class classification.

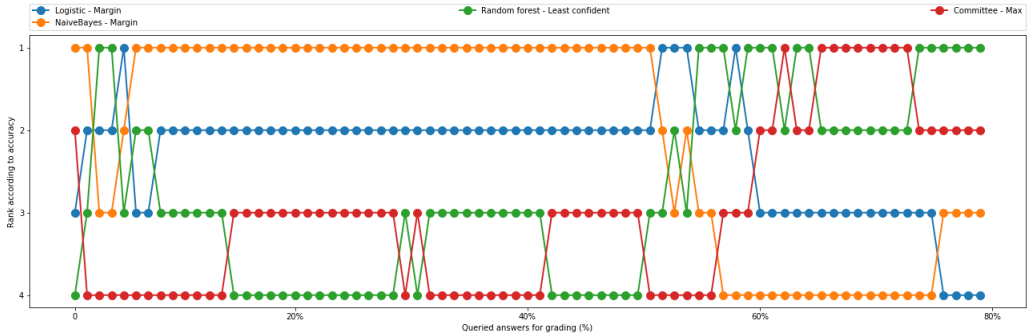


Figure 6.16: Bump chart of model performance in multi-class classification.

6.3 Experiment 3: Mohler'11 Dataset using Term Frequency-Inverse Document Frequency (tf-idf) Features

In this experiment, the Mohler'11 dataset was experimented using term frequency-inverse document frequency (tf-idf) features. Extracting this feature doesn't require reference answers of the questions as the entire tf-idf vectors were created from the student answers alone. Every answer was encoded into a vector of length 2138 and this vector was used as input to our models during training and prediction. Similar to the BOW features, tf-idf features also make up a huge feature space and thus, linear SVC and kernel based SVC were ignored in this experiment due to the time constraints.

6.3.1 Binary Classification

The experimental results of different model-query strategy combinations in a binary classification setting are shown in Fig 6.17 and 6.19a. The F1 score analysis of this experiment is shown in Fig 6.18 and Fig 6.19b.

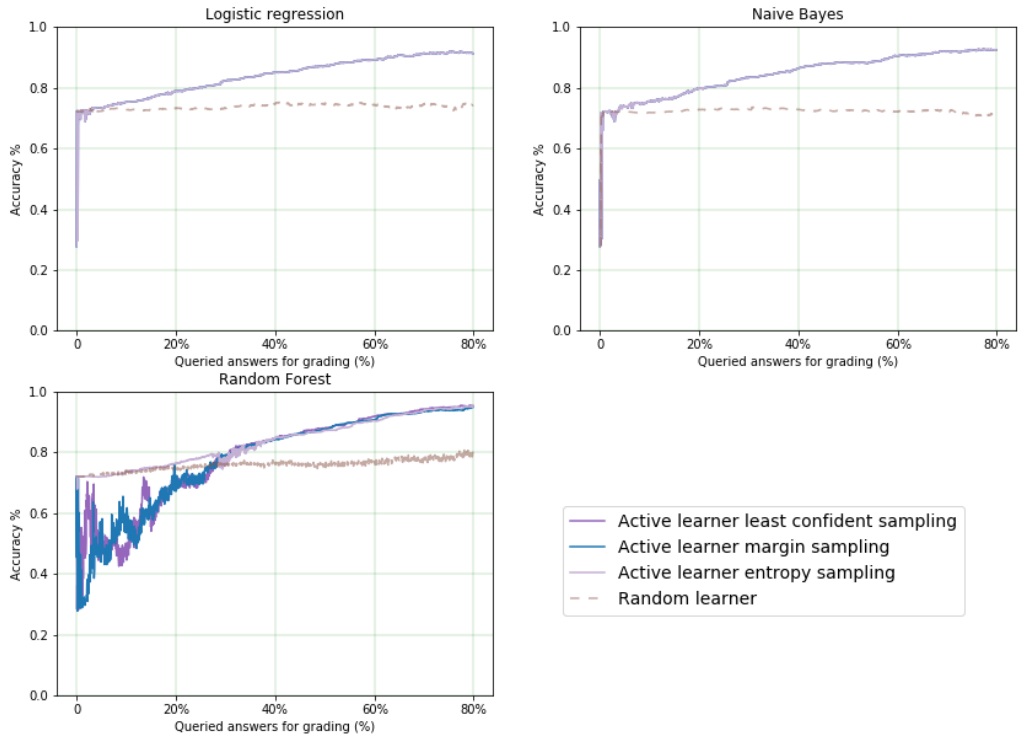


Figure 6.17: Results in terms of accuracy for different models with uncertainty based query strategies.

We observe that all the active learning strategies in their respective models performed comparably to each other. Hence, we selected the margin uncertainty sampling for the naive Bayes, logistic regression, and random forest classifier models to have a closer look at their performance over increasing number of queries using a bump chart. This chart shows the performance of every model ranked by their accuracies which is shown in Fig 6.20. Naive Bayes classifier performs well over a

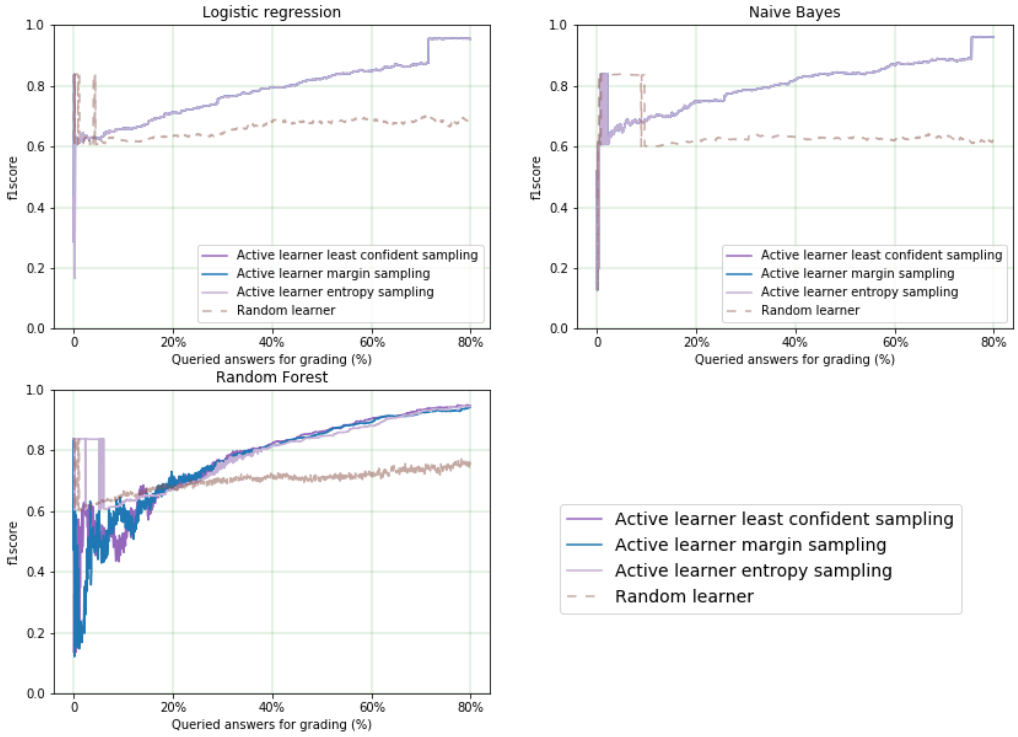
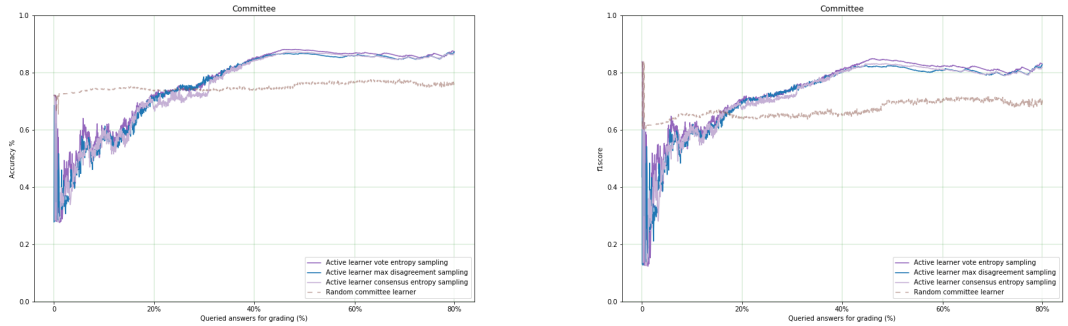


Figure 6.18: Results in terms of F1 score for different models with uncertainty based query strategies.

long range consistently . Random forest performs well at the end, but only after a high number of queries.



(a) Results in terms of accuracy for committee based query strategies.

(b) Results in terms of F1 score for committee based query strategies.

Figure 6.19: Committee-based binary classification.

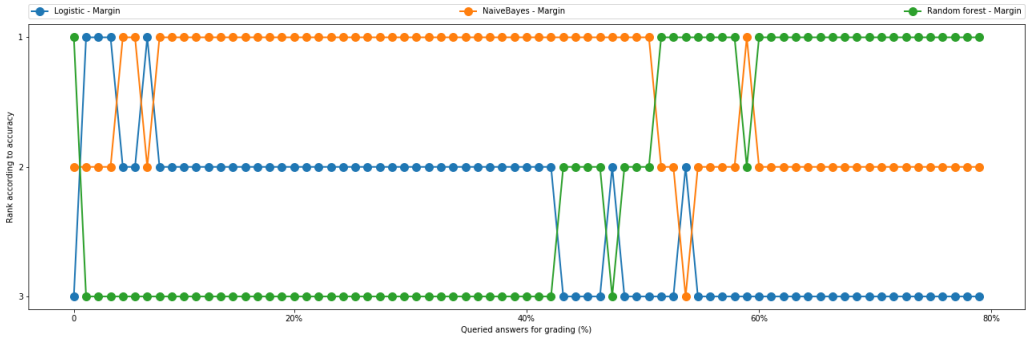


Figure 6.20: Bump chart of model performance in binary classification.

6.3.2 Multi-class Classification

The experimental results of different model-query strategy combinations in a multi-class classification setting are shown in Fig 6.21 and 6.23a. The results were also analyzed using the F1 score which are shown in Fig 6.22 and Fig 6.23b.

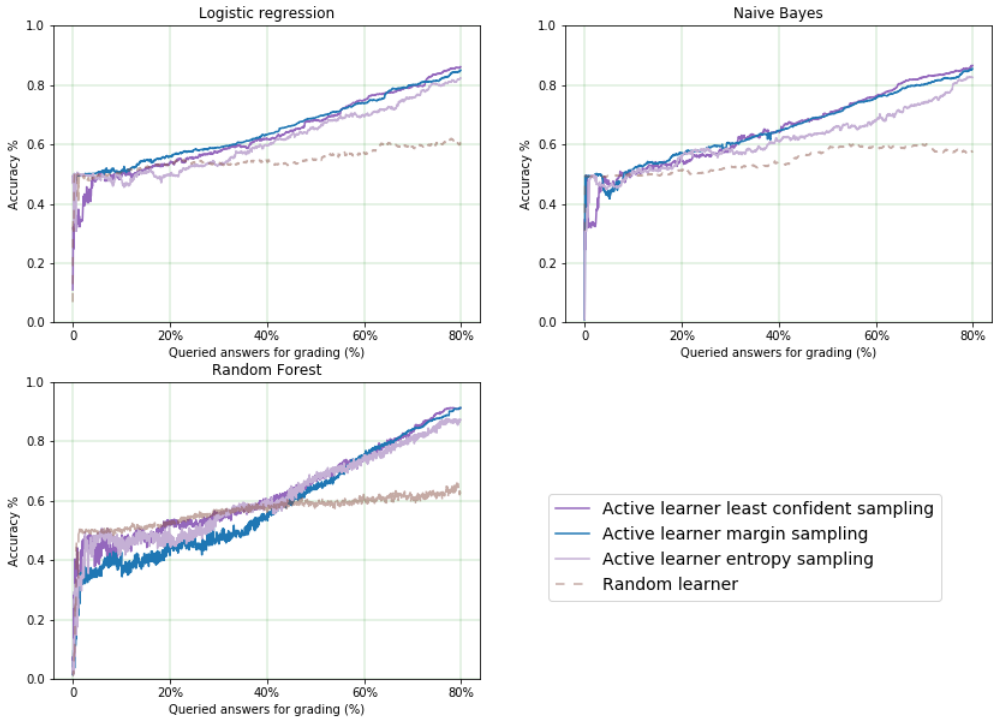


Figure 6.21: Results in terms of accuracy for different models with uncertainty based query strategies.

Considering the accuracy and the F1 scores, Logistic Regression with margin uncertainty sampling, Naive Bayes with margin uncertainty sampling, and Random Forest with least confident uncertainty sampling were selected for comparison using a bump chart. In case of committee, max disagreement sampling performed better and is added to the bump chart comparison. This chart shows the performance of every model ranked by their accuracies which is shown in Fig 6.24. The bump chart

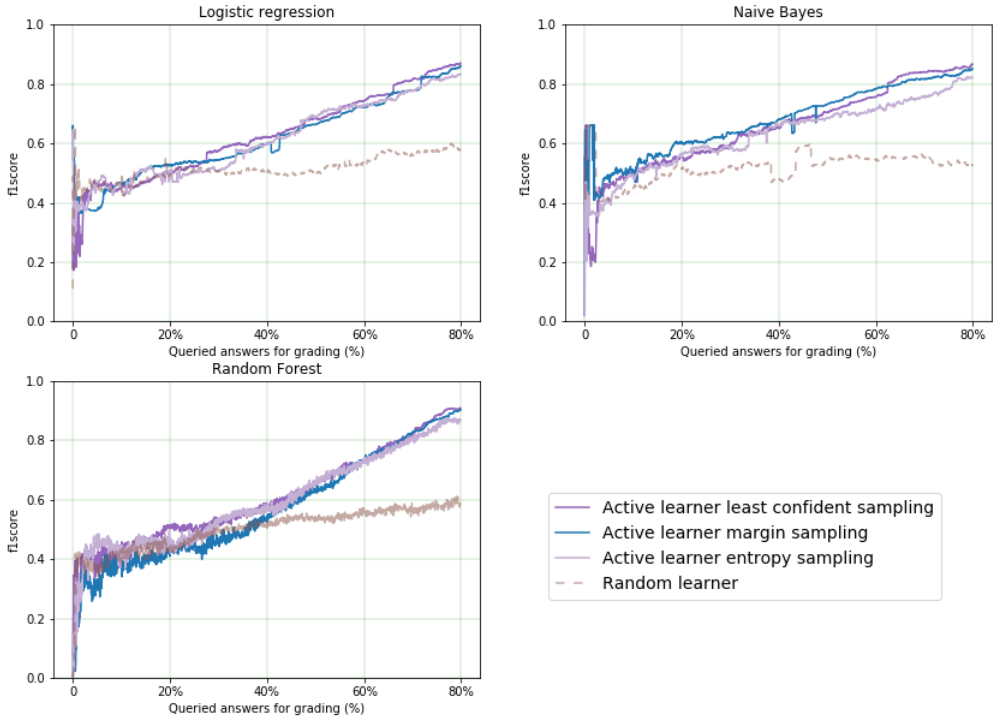


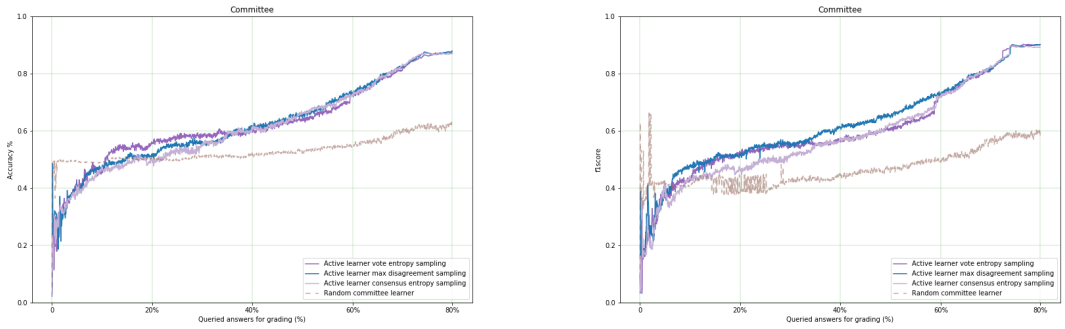
Figure 6.22: Results in terms of F1 score for different models with uncertainty based query strategies.

shows that the naive Bayes classifier performs well over a long range. When the query percentage reaches 65%, other models become comparable in performance.

The time comparison of different models is tabulated in Table 6.3. Considering the bump plot and the time comparison, it could be inferred that naive Bayes classifier with margin based uncertainty sampling fits best for this setting.

	Time taken per query (secs)					
Models	**least confident uncertainty**	**margin based uncertainty**	**entropy based uncertainty**	**vote entropy**	**max disagreement**	**consensus entropy**
Logistic Regression	0.0395	0.0388	.0399	1.0785	1.0671	1.0725
Naive Bayes	0.0245	0.0243	0.0249			
Random Forest (100 trees)	0.9430	0.9340	0.9538			

Table 6.3: Time comparison between different models for multi-class classification.



(a) Results in terms of accuracy for committee based query strategies.

(b) Results in terms of F1 score for committee based query strategies.

Figure 6.23: Committee-based multi-class classification.

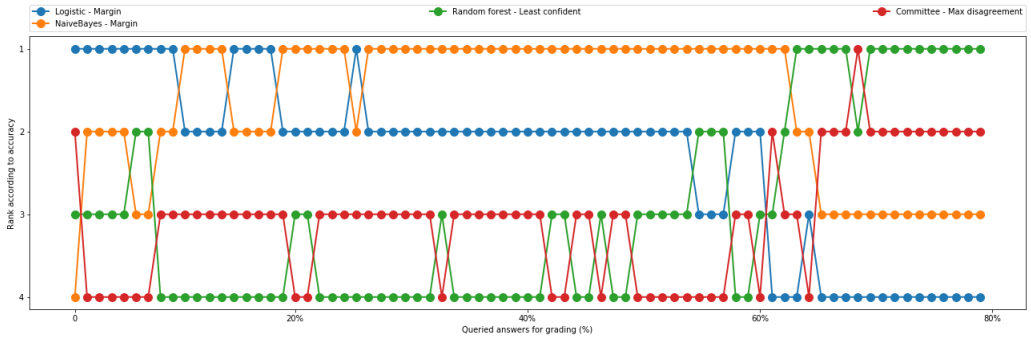


Figure 6.24: Bump chart of model performance in multi-class classification.

6.4 Experiment 4: Neural Network Dataset using Features from Sultan'16

Similar ot the Mohler'11 dataset, the features mentioned in [73] were extracted from the in-house Neural Network exam dataset and used in this experiment. Analysis on binary classification was not performed with this dataset as the grade distribution contained just 3 grades (0,1,2).

6.4.1 Multi-class Classification

The experimental results of different model-query strategy combinations in a multi-class classification setting are shown in Fig 6.25 and 6.27a. The results were also analyzed using the F1 score which are shown in Fig 6.26 and Fig 6.27b.

Taking the accuracy and the F1 scores into account, Logistic Regression with margin uncertainty sampling, Naive Bayes with least confident uncertainty sampling, Random Forest with least confident uncertainty sampling, and SVC-linear classifier with margin uncertainty sampling were selected for comparison using a bump chart Fig 6.28. The committee based query strategies didn't perform comparably to the uncertainty based query strategies and were ignored in the bump chart. The bump chart shows that the random forest with least confident sampling works well in the initial stages and after 45% queries, logistic regression with margin sampling performs well. Hence, it could be inferred that both logistic regression and random forest work well for this dataset with Sultan'16 features.

The time comparison of different models is tabulated in Table 6.4. Considering the bump plot and the time comparison, it could be inferred that random forest classifier with least confident uncertainty sampling fits best for this setting. Though it takes approximately 0.3 seconds to process each query, that time would be negligible for the task of assisted-grading.

	Time taken per query (secs)					
Models	**least confident uncertainty**	**margin based uncertainty**	**entropy based uncertainty**	**vote entropy**	**max disagreement**	**consensus entropy**
Logistic Regression	0.0069	0.0070	0.0067	0.2169	0.2189	0.2150
Naive Bayes	0.0057	0.0060	0.0069			
Random Forest (100 trees)	0.3407	0.3323	0.3028			
SVC - Linear	0.0206	0.0203	0.0216	-	-	-
SVC - RBF	0.0473	0.0495	0.0444	-	-	-

Table 6.4: Time comparison between different models for multi-class classification.

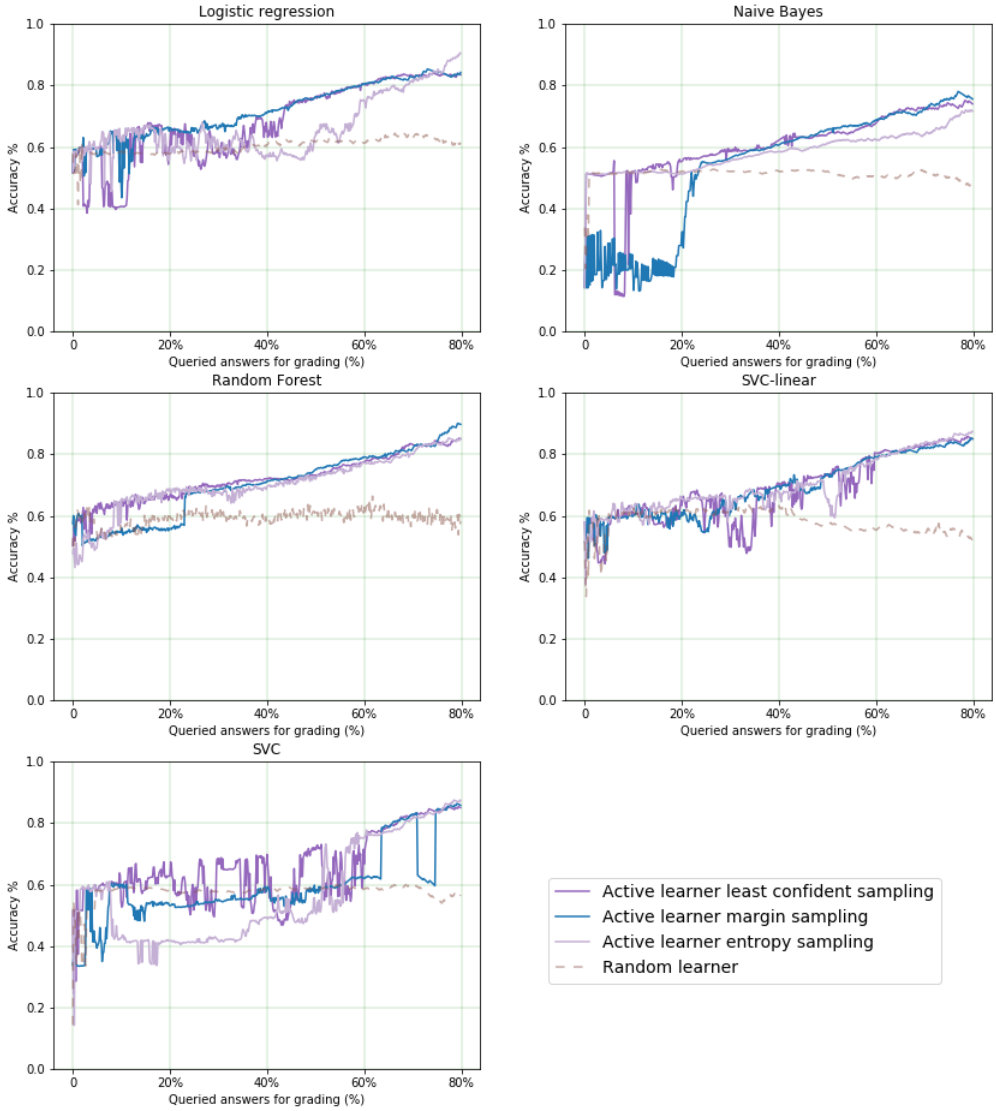


Figure 6.25: Results in terms of accuracy for different models with uncertainty based query strategies.

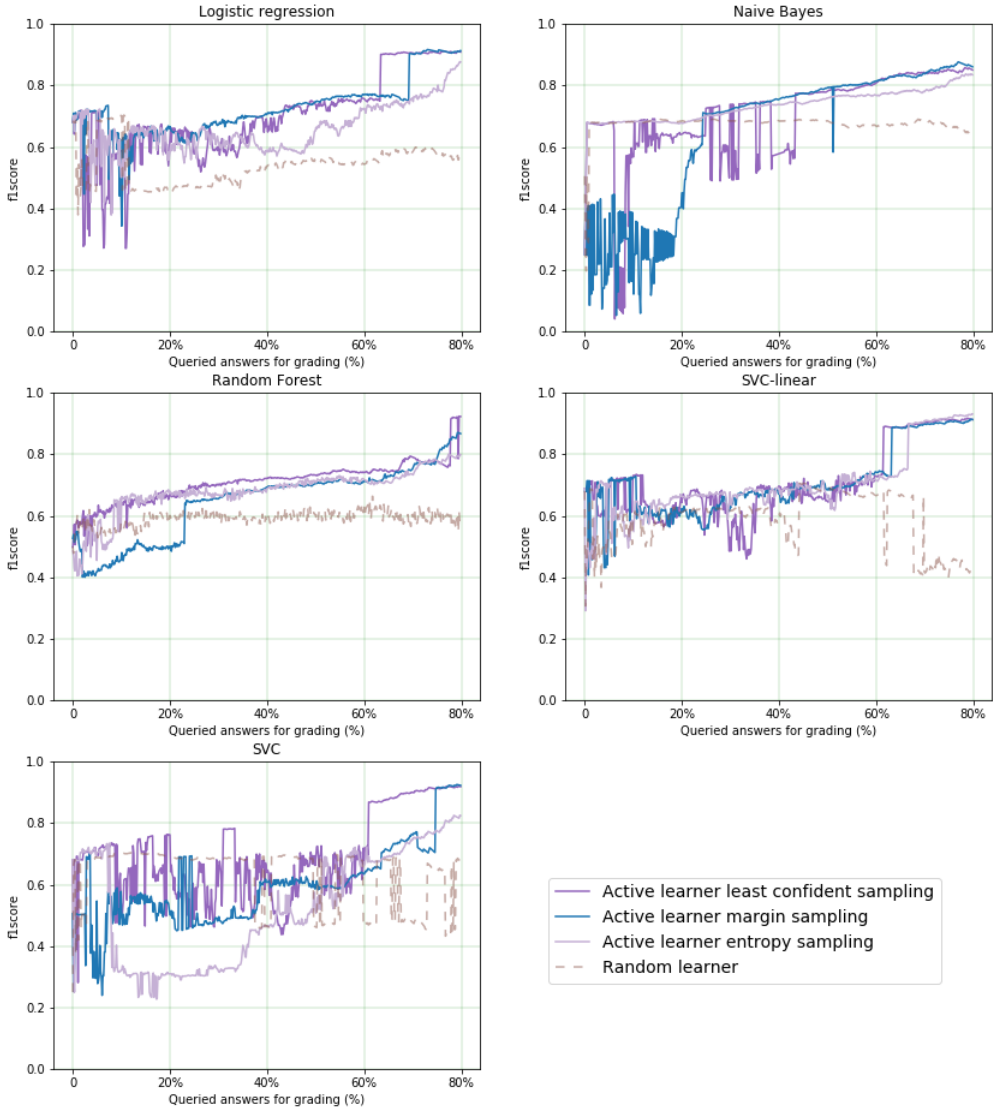
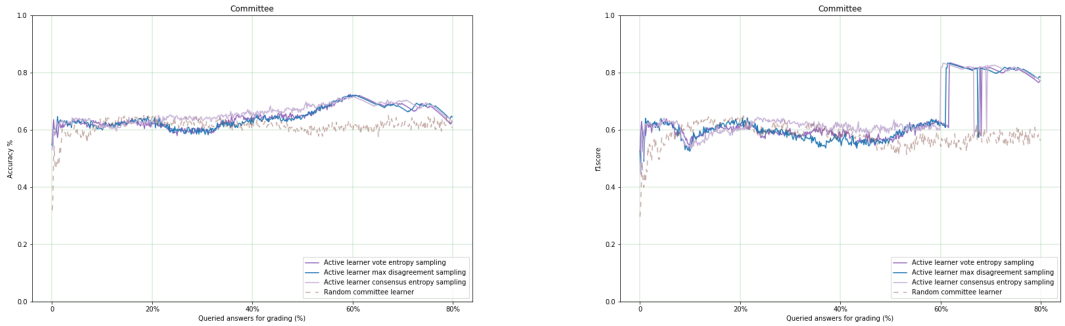


Figure 6.26: Results in terms of F1 score for different models with uncertainty based query strategies.



(a) Results in terms of accuracy for committee based query strategies.

(b) Results in terms of F1 score for committee based query strategies.

Figure 6.27: Committee-based multi-class classification.

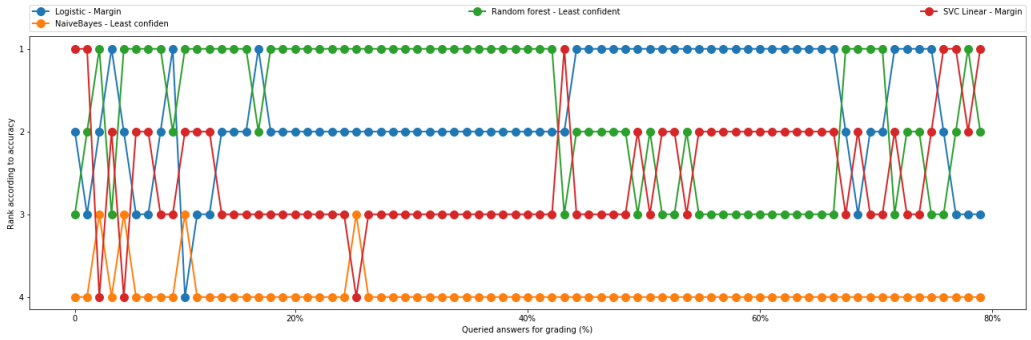


Figure 6.28: Bump chart of model performance in multi-class classification.

6.5 Experiment 5: Neural Network Dataset using Bag-of-words (BOW) Features

Bag-of-words (BOW) of the answers were used as features for this experiment. Extracting this feature doesn't require reference answers of the questions as the entire BOW vectors were created from the student answers alone. Every answer was encoded into a vector of length 2540 and this vector was used as input to our models during training and prediction. The linear SVC and RBF kernel SVC models were ignored in this experiment as they consume a lot of time. Considering the future implementation of this project as an interactive task, time becomes a key factor to be considered.

6.5.1 Multi-class Classification

The experimental results of different model-query strategy combinations in a multi-class classification setting are shown in Fig 6.29 and 6.31a. The results were also analyzed using the F1 score which are shown in Fig 6.30 and Fig 6.31b. The accuracy plots show that it is able to reach 70% of accuracy with just 20% queries in all the models. The accuracy improves with a small slope in the range of 25 - 80% queries. It could be inferred from this that active learning is a better choice for this task as it performs well with less data.

Considering the accuracy and the F1 scores, Logistic Regression with margin uncertainty sampling, Naive Bayes with least confident uncertainty sampling, and Random Forest with margin uncertainty sampling were selected for comparison using a bump chart Fig 6.32. In case of committee, max disagreement sampling performed better and is added to the bump chart comparison. The bump chart shows that the random forest classifier performs well over a long range.

The time comparison of different models is tabulated in Table 6.5. Considering the bump plot and the time comparison, it could be inferred that random forest classifier with margin based uncertainty sampling fits best for this setting.

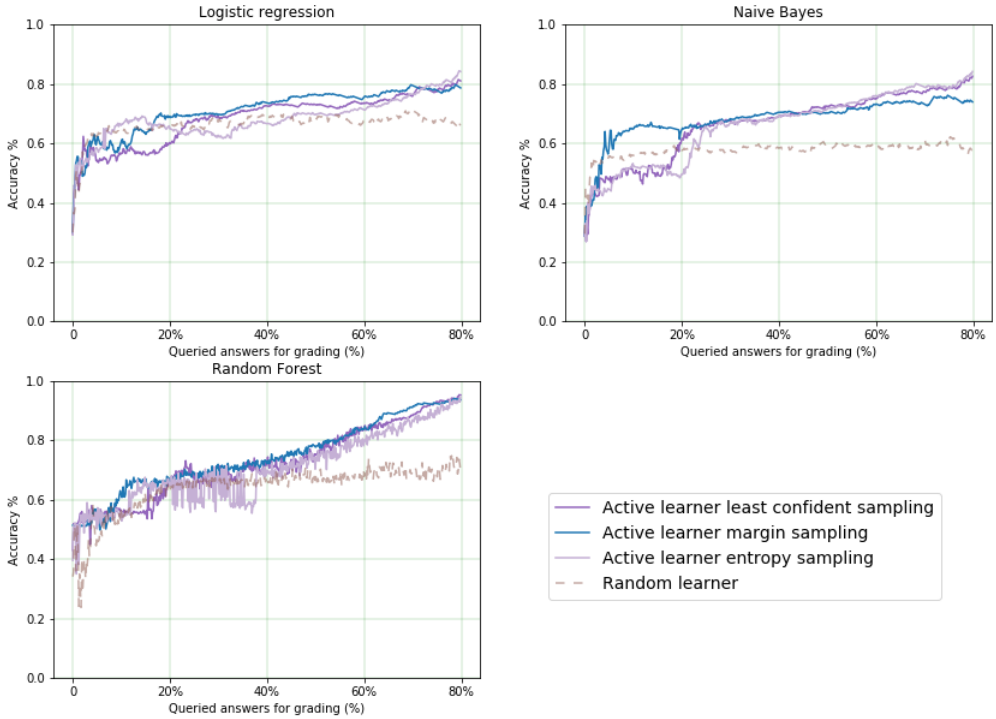


Figure 6.29: Results in terms of accuracy for different models with uncertainty based query strategies.

	Time taken per query (secs)					
Models	**least confident uncertainty**	**margin based uncertainty**	**entropy based uncertainty**	**vote entropy**	**max disagreement**	**consensus entropy**
Logistic Regression	0.0269	0.0261	0.0260	0.3481	0.3450	0.3276
Naive Bayes	0.0199	0.0187	0.0184			
Random Forest (100 trees)	0.4545	0.4214	0.3952			

Table 6.5: Time comparison between different models for multi-class classification.

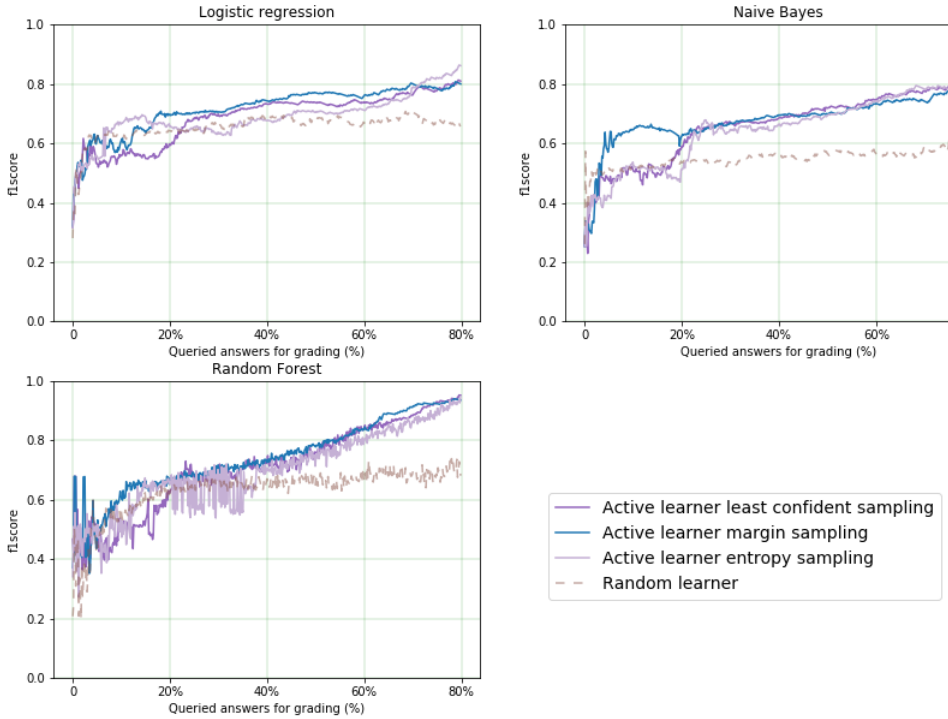
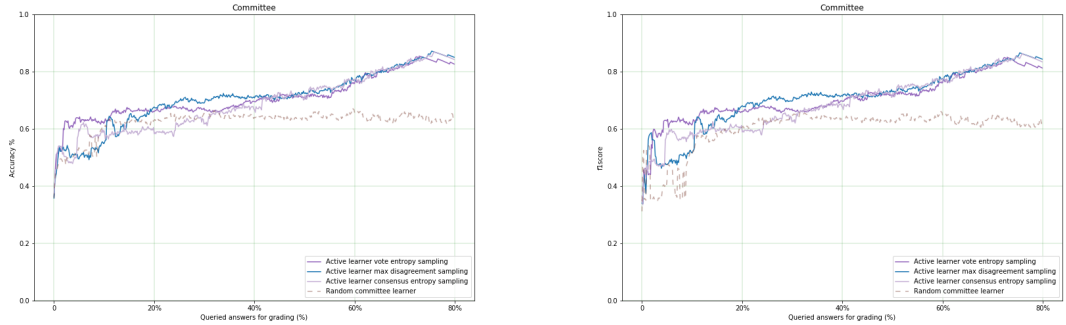


Figure 6.30: Results in terms of F1 score for different models with uncertainty based query strategies.



(a) Results in terms of accuracy for committee based query strategies.

(b) Results in terms of F1 score for committee based query strategies.

Figure 6.31: Committee-based multi-class classification.

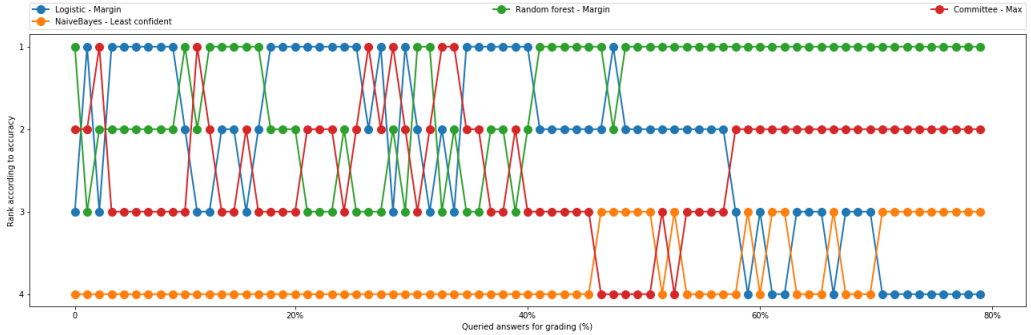


Figure 6.32: Bump chart of model performance in multi-class classification.

6.6 Experiment 6: Neural Network Dataset using Term Frequency-Inverse Document Frequency (tf-idf) Features

Neural Network dataset was experimented using term frequency-inverse document frequency (tf-idf) features. Every answer was encoded into a vector of length 2540 and this vector was used as input to our models during training and prediction. Similar to the BOW features, tf-idf features also make up a huge feature space and thus, linear SVC and kernel based SVC were ignored in this experiment due to the time constraints.

6.6.1 Multi-class Classification

The experimental results of different model-query strategy combinations in a multi-class classification setting are shown in Fig 6.33 and 6.35a. The results were also analyzed using the F1 score which are shown in Fig 6.34 and Fig 6.35b.

Considering the accuracy and the F1 scores, Logistic Regression with least confident uncertainty sampling, Naive Bayes with least confident uncertainty sampling, and Random Forest with margin uncertainty sampling were selected for comparison using a bump chart Fig 6.36. In case of committee, vote entropy sampling performed better and is added to the bump chart comparison. The bump chart shows that the random forest classifier clearly dominates over a long range. All the models perform comparably in the initial stages.

The time comparison of different models is tabulated in Table 6.6. Considering the bump plot and the time comparison, it could be inferred that random forest classifier with margin based uncertainty sampling fits best for this setting.

	Time taken per query (secs)					
Models	**least confident uncertainty**	**margin based uncertainty**	**entropy based uncertainty**	**vote entropy**	**max disagreement**	**consensus entropy**
Logistic Regression	0.0395	0.0388	.0399	1.0785	1.0671	1.0725
Naive Bayes	0.0245	0.0243	0.0249			
Random Forest (100 trees)	0.9430	0.9340	0.9538			

Table 6.6: Time comparison between different models for multi-class classification.

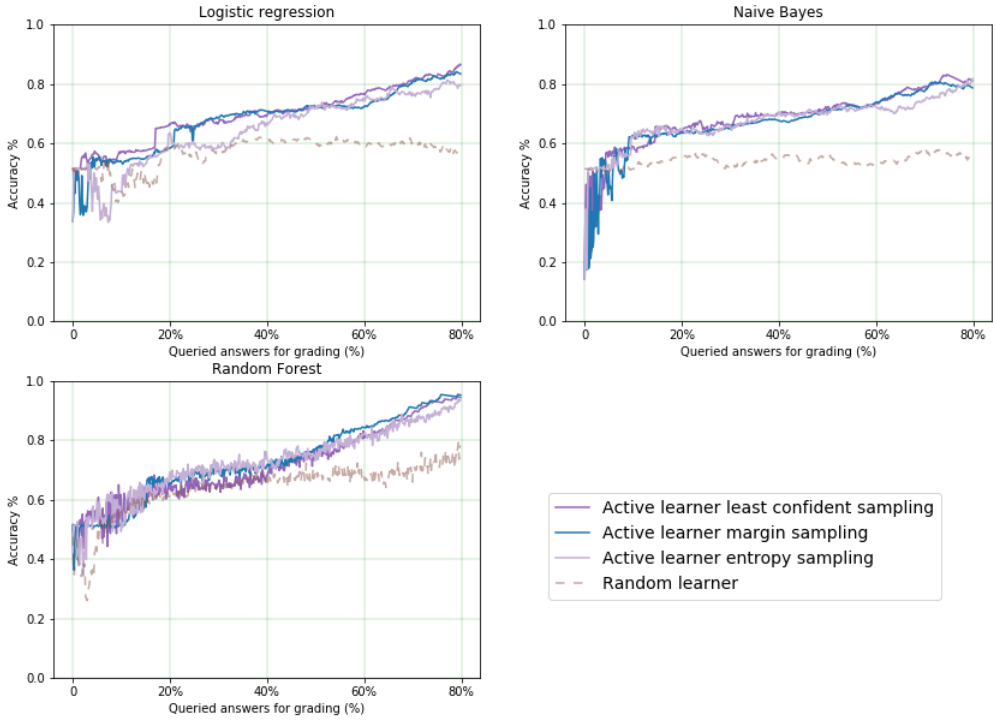


Figure 6.33: Results in terms of accuracy for different models with uncertainty based query strategies.

6.7 Experiment 7: SemEval Dataset using Features from Sultan'16

Sultan'16 features were used in this experiment. The task of grading was split into binary classification and multi-class classification. The objective of binary classification is to identify correct and incorrect answers whereas multi-class classification's objective is to assign the correct grade for each answer.

6.7.1 Binary Classification

The experimental results of different model-query strategy combinations in a binary classification setting are shown in Fig 6.37 and 6.39a. As the class distri-

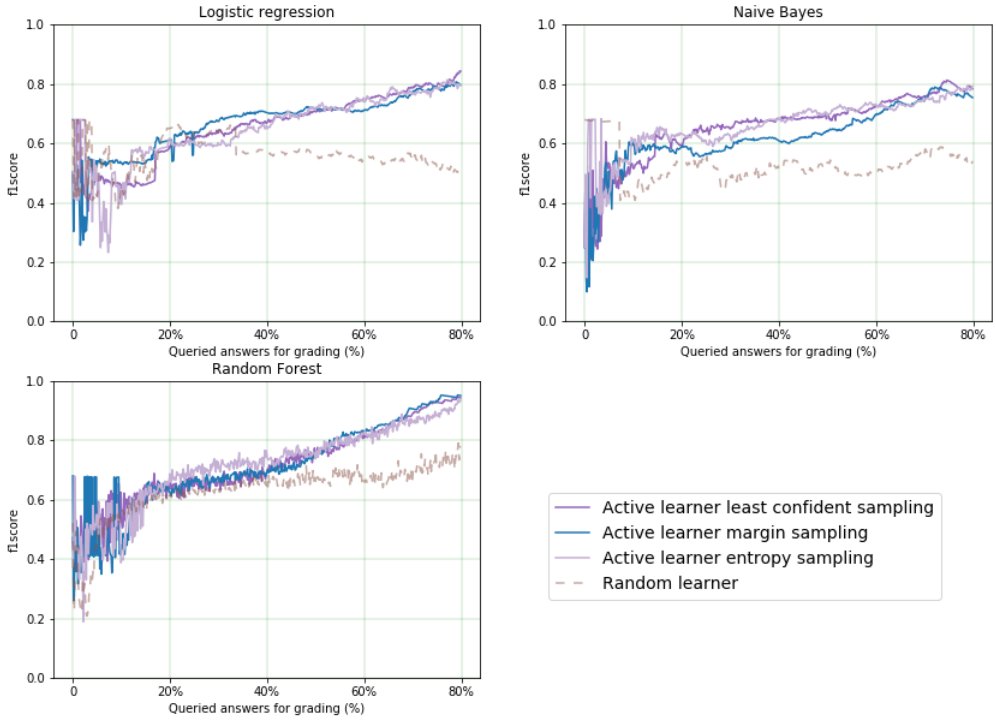
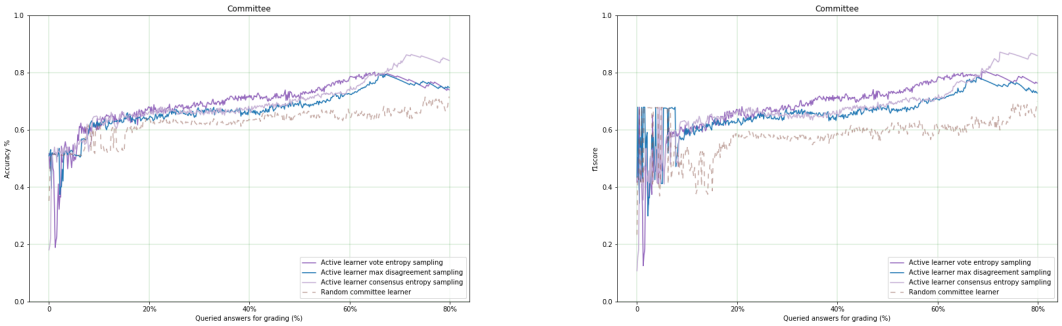


Figure 6.34: Results in terms of F1 score for different models with uncertainty based query strategies.

bution of this dataset is uneven, accuracy alone would not be enough to judge the performance of the models. Hence, the results were also analyzed using the F1 score which are shown in Fig 6.38 and Fig 6.39b.

Based on these graphs, we observe that all the active learning query strategies outperform the passive learning. We also observe that all the active learning strategies in their respective models performed comparably to each other. Hence, we selected the margin based uncertainty sampling for each model in order to compare them using a bump chart. This chart shows the performance of every model ranked by their accuracies which is shown in Fig 6.40. The bump chart for binary classification shows that all logistic regression with margin uncertainty sampling works well for this setting, and is a clear winner performing consistently better over a long range.



(a) Results in terms of accuracy for committee based query strategies.

(b) Results in terms of F1 score for committee based query strategies.

Figure 6.35: Committee-based multi-class classification.

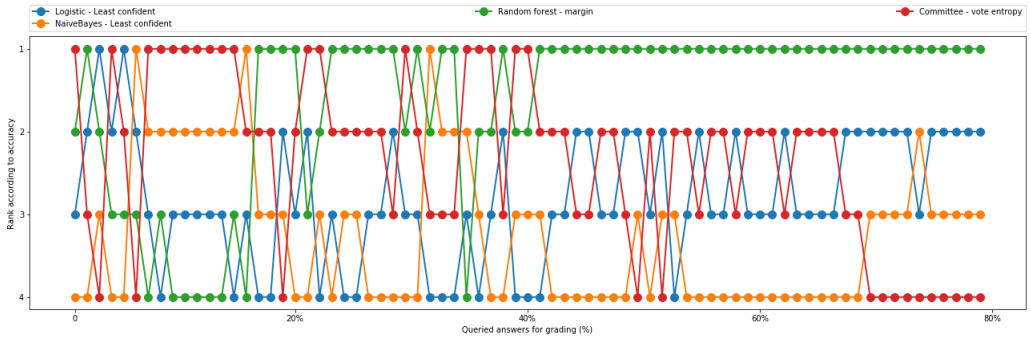


Figure 6.36: Bump chart of model performance in multi-class classification.

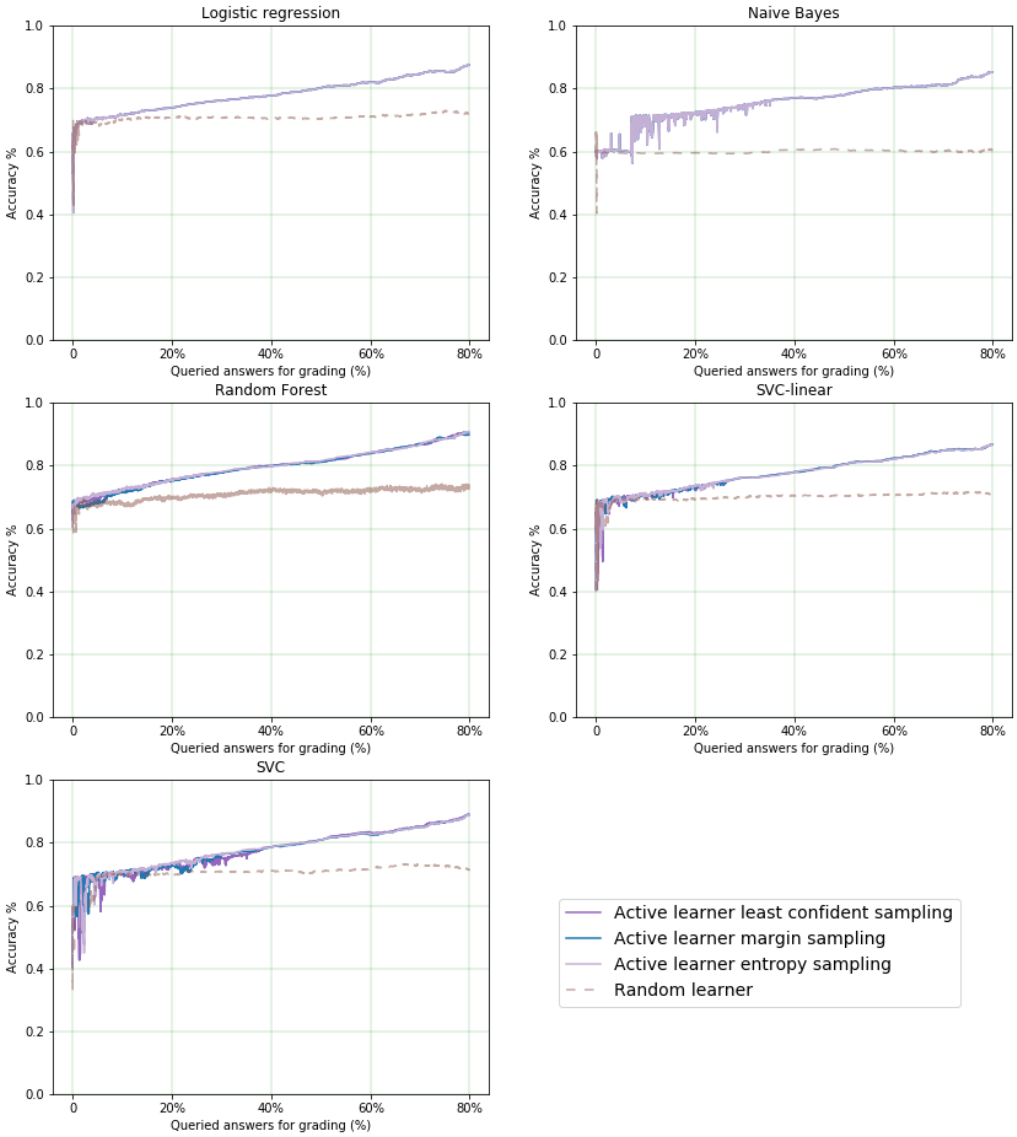


Figure 6.37: Results in terms of accuracy for different models with uncertainty based query strategies.

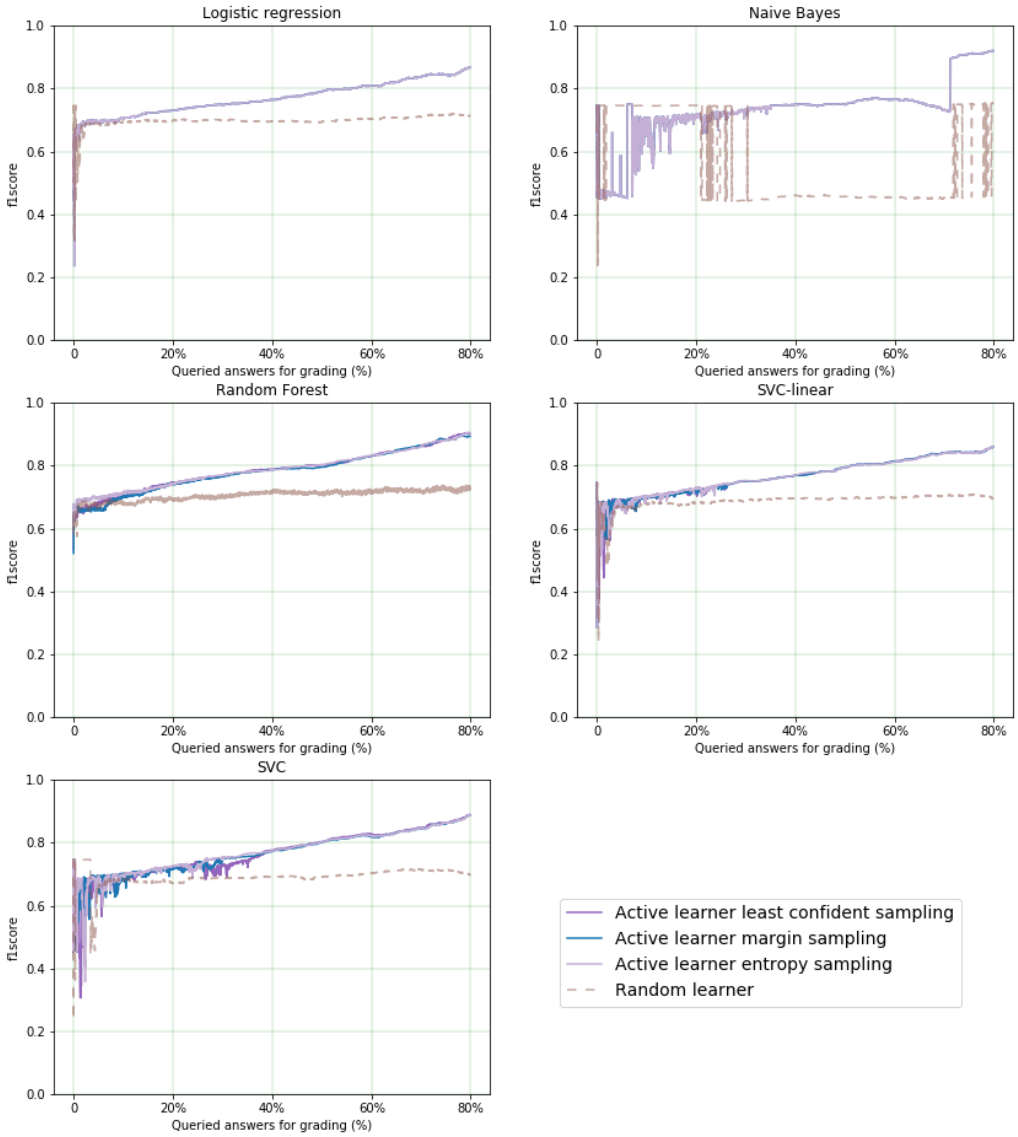
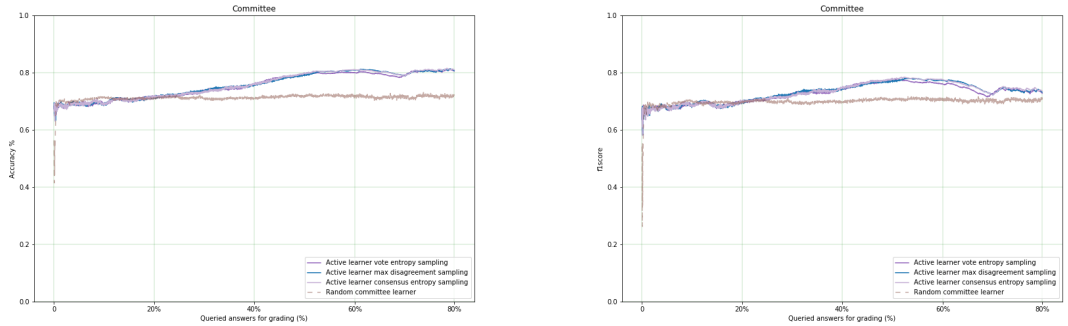


Figure 6.38: Results in terms of F1 score for different models with uncertainty based query strategies.



(a) Results in terms of accuracy for committee based query strategies.

(b) Results in terms of F1 score for committee based query strategies.

Figure 6.39: Committee-based binary classification.

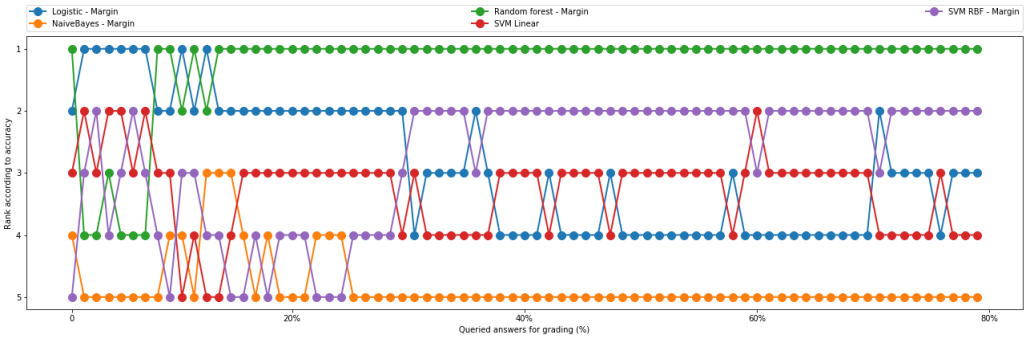


Figure 6.40: Bump chart of model performance in binary classification.

6.7.2 Multi-class Classification

The experimental results of different model-query strategy combinations in a multi-class classification setting are shown in Fig 6.41 and 6.43a. The results were also analyzed using the F1 score which are shown in Fig 6.42 and Fig 6.43b.

We observe that all the active learning query strategies outperform the passive learning (random sampling). Considering the accuracy and the F1 scores, logistic regression with margin uncertainty sampling, naive Bayes with margin uncertainty sampling, random forest with margin uncertainty sampling, and linear SVM with least confident uncertainty sampling were selected for comparison using a bump chart. In case of committee, max disagreement sampling performed better and is added to the bump chart comparison. This chart shows the performance of every model ranked by their accuracies which is shown in Fig 6.44.

The bump chart shows that the random forest classifier model with margin uncertainty sampling and logistic regression classifier with margin uncertainty sampling are preferable for this setting. Random forest classifier shows consistent and good performance after 40% of the queries.

The time comparison of different models is tabulated in Table 6.7. Based on the bump plot and time comparison, it could be inferred that random forest with least confident uncertainty sampling is best for SemEval 2013 dataset with Sultan'16 features.

	Time taken per query (secs)					
Models	**least confident uncertainty**	**margin based uncertainty**	**entropy based uncertainty**	**vote entropy**	**max disagreement**	**consensus entropy**
Logistic Regression	0.0530	0.0506	0.0519	0.8570	0.8132	0.8008
Naive Bayes	0.0217	0.0199	0.0217			
Random Forest (100 trees)	0.7391	0.5579	0.5538			
SVC Linear	0.6159	0.5972	0.6064	-	-	-
SVC RBF	1.9078	1.8223	1.9001	-	-	-

Table 6.7: Time comparison between different models for multi-class classification.

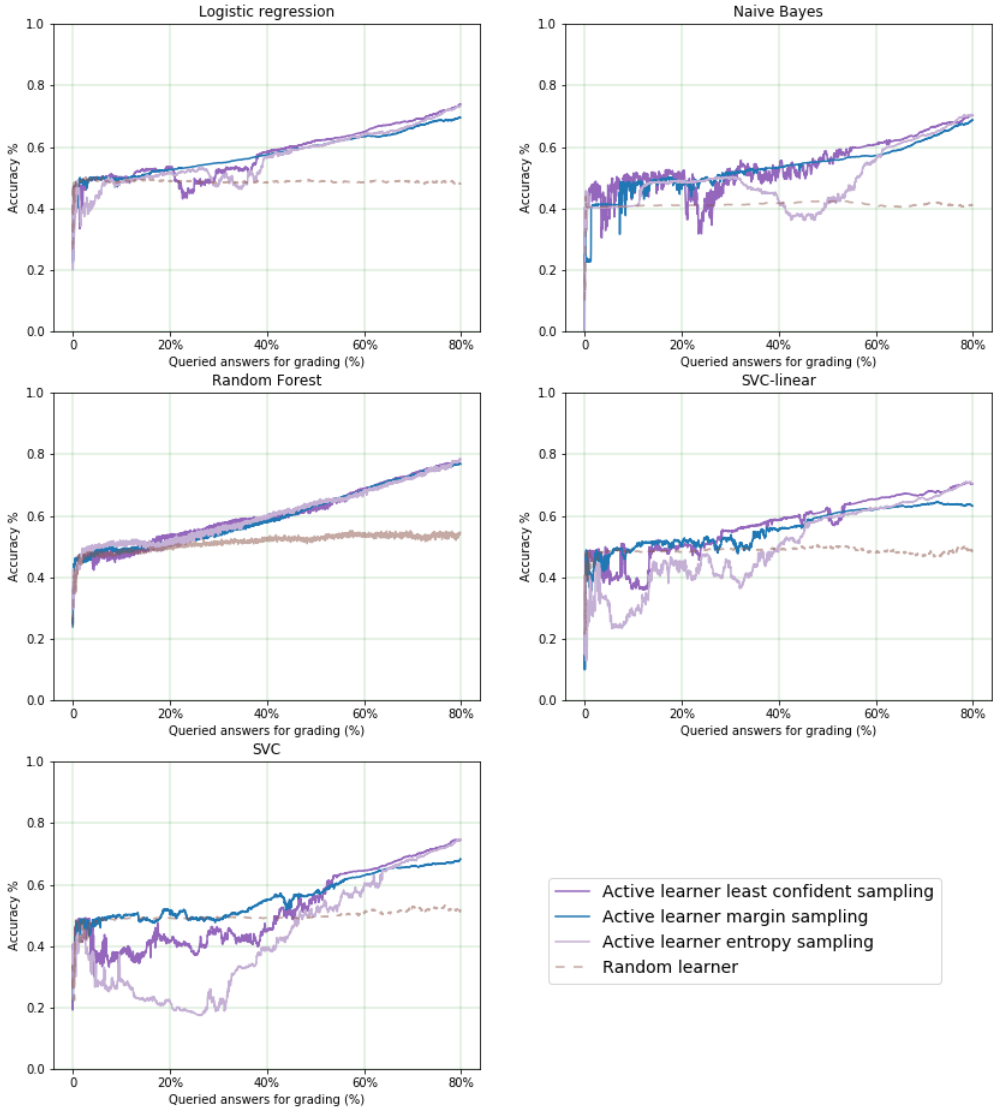


Figure 6.41: Results in terms of accuracy for different models with uncertainty based query strategies.

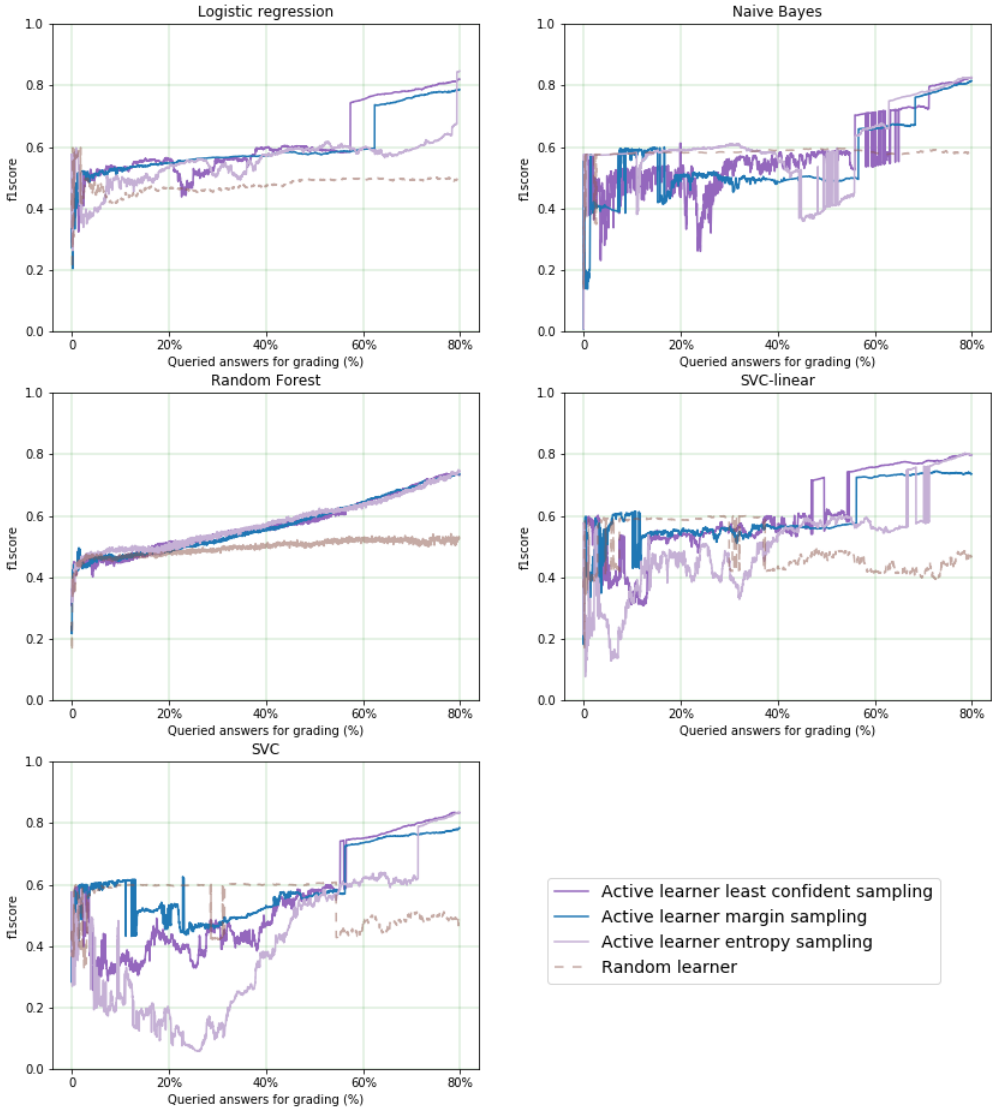
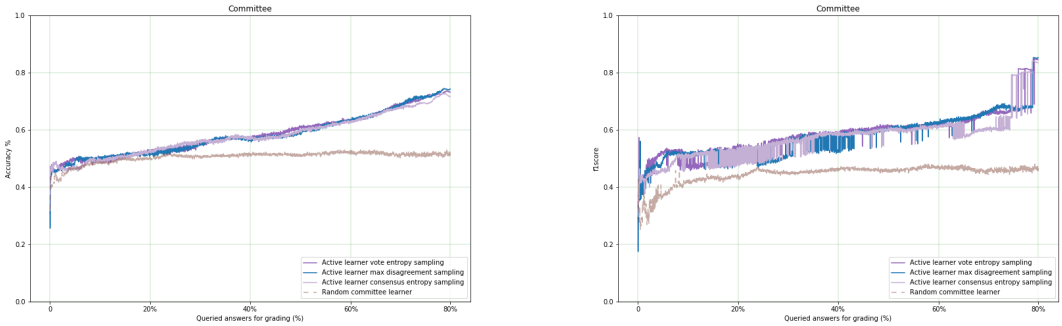


Figure 6.42: Results in terms of F1 score for different models with uncertainty based query strategies.



(a) Results in terms of accuracy for committee based query strategies.

(b) Results in terms of F1 score for committee based query strategies.

Figure 6.43: Committee-based multi-class classification.

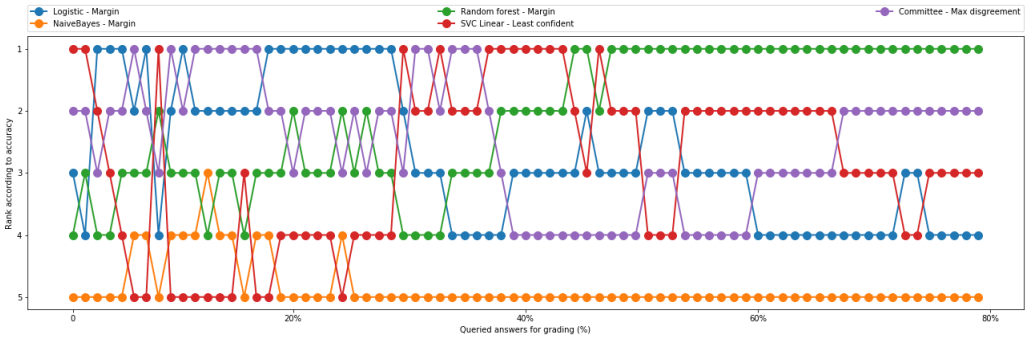


Figure 6.44: Bump chart of model performance in multi-class classification.

6.8 Experiment 8: SemEval 2013 Dataset using Bag-of-words (BOW) Features

Bag-of-words(BOW) of the answers were used as features for this experiment. Extracting this feature doesn't require reference answers of the questions as the entire BOW vectors were created from the student answers alone. Every answer was encoded into a vector of length 1908 and this vector was used as input to our models during training and prediction. The linear SVC and RBF kernel SVC models were ignored in this experiment as they consume a lot of time to select each query since the dimension of the feature space is very high. Considering the future implementation of this project as an interactive task, time becomes a key factor to be considered.

6.8.1 Binary Classification

The experimental results of different model-query strategy combinations in a binary classification setting are shown in Fig 6.45 and 6.47a. The F1 score analysis of this experiment is shown in Fig 6.46 and Fig 6.47b.

We observe that all the active learning strategies in their respective models performed comparably to each other. Hence, we selected the margin uncertainty sampling for the random forest and logistic regression models, entropy uncertainty for naive Bayes classifier, and vote entropy sampling for the committee based model to have a closer look at their performance over increasing number of queries using a bump chart. This chart shows the performance of every model ranked by their accuracies which is shown in Fig 6.48.

We can infer from the bump chart that the random forest with margin uncertainty sampling performs well in the overall range. When considering the overall chart, it seems wise to select random forest classifier with margin based sampling for this setting.

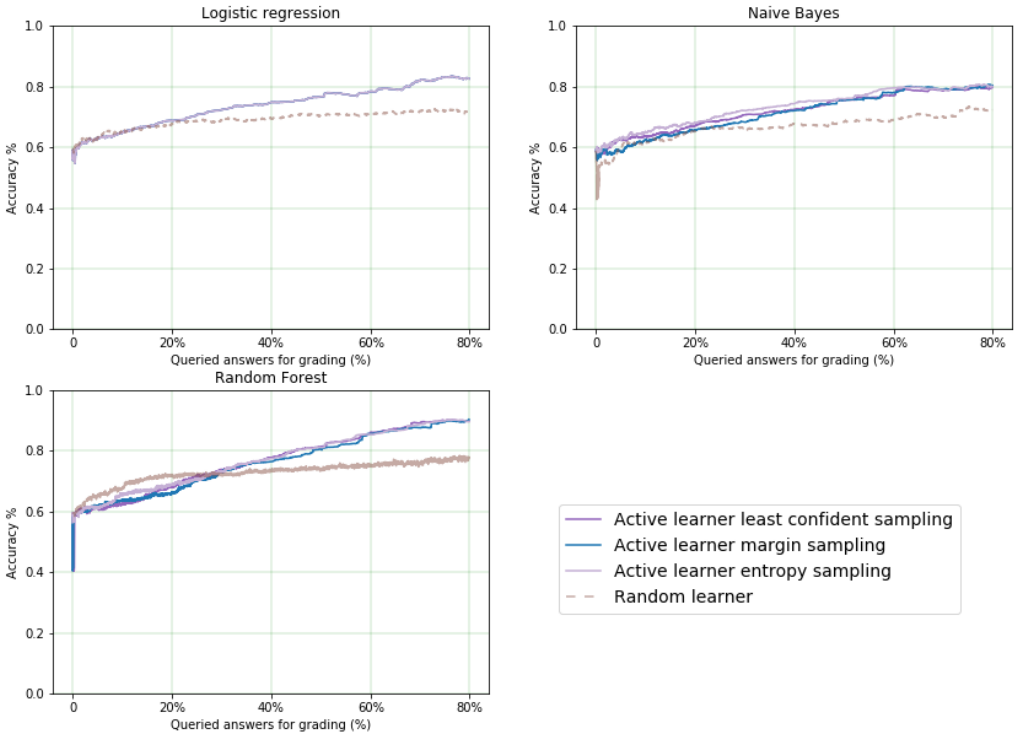


Figure 6.45: Results in terms of accuracy for different models with uncertainty based query strategies.

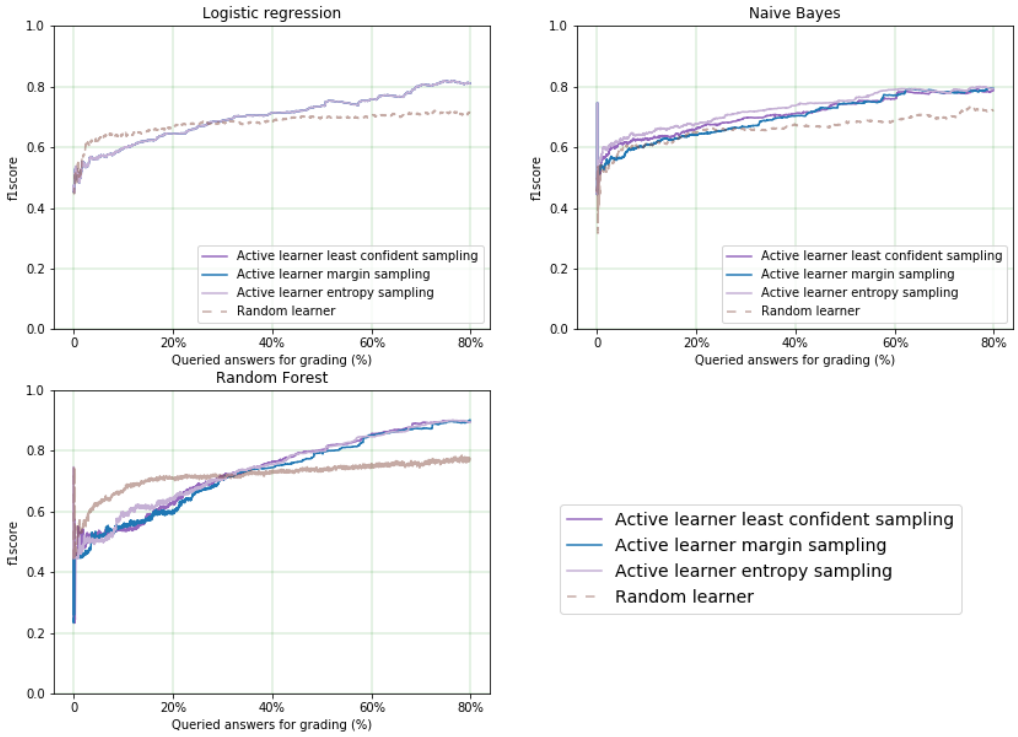
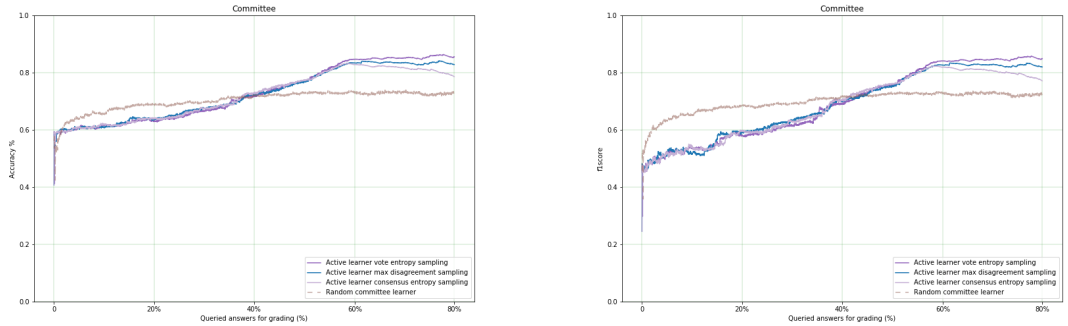


Figure 6.46: Results in terms of F1 score for different models with uncertainty based query strategies.



(a) Results in terms of accuracy for committee based query strategies.

(b) Results in terms of F1 score for committee based query strategies.

Figure 6.47: Committee-based binary classification.

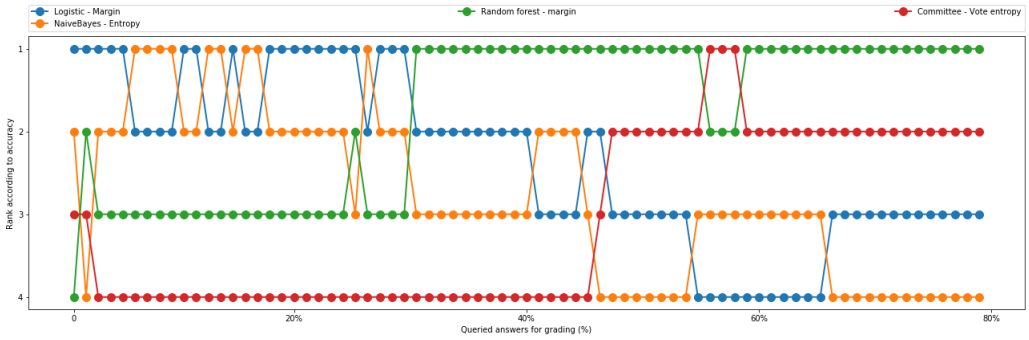


Figure 6.48: Bump chart of model performance in binary classification.

6.8.2 Multi-class Classification

The experimental results of different model-query strategy combinations in a multi-class classification setting are shown in Fig 6.49 and 6.51a. The results were also analyzed using the F1 score which are shown in Fig 6.50 and Fig 6.51b.

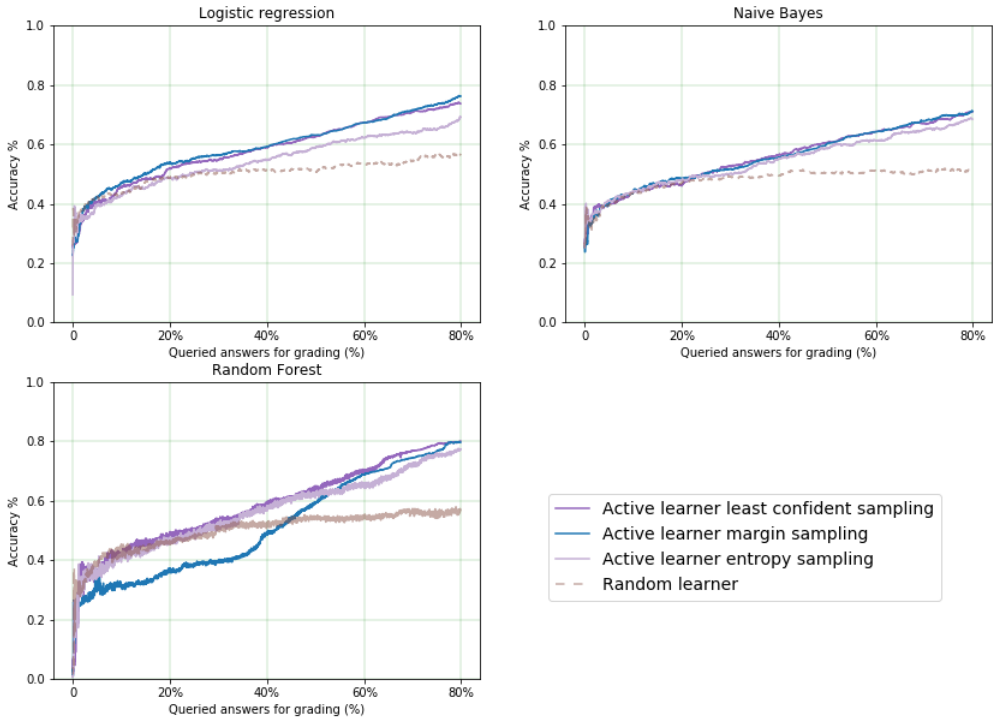


Figure 6.49: Results in terms of accuracy for different models with uncertainty based query strategies.

Considering the accuracy and the F1 scores, Logistic Regression with margin uncertainty sampling, Naive Bayes with margin uncertainty sampling, and Random Forest with least confident uncertainty sampling were selected for comparison using a bump chart. In case of committee, max disagreement sampling performed better and is added to the bump chart Fig 6.52 comparison. The bump chart shows that the logistic regression classifier and random forest perform compably to each other.

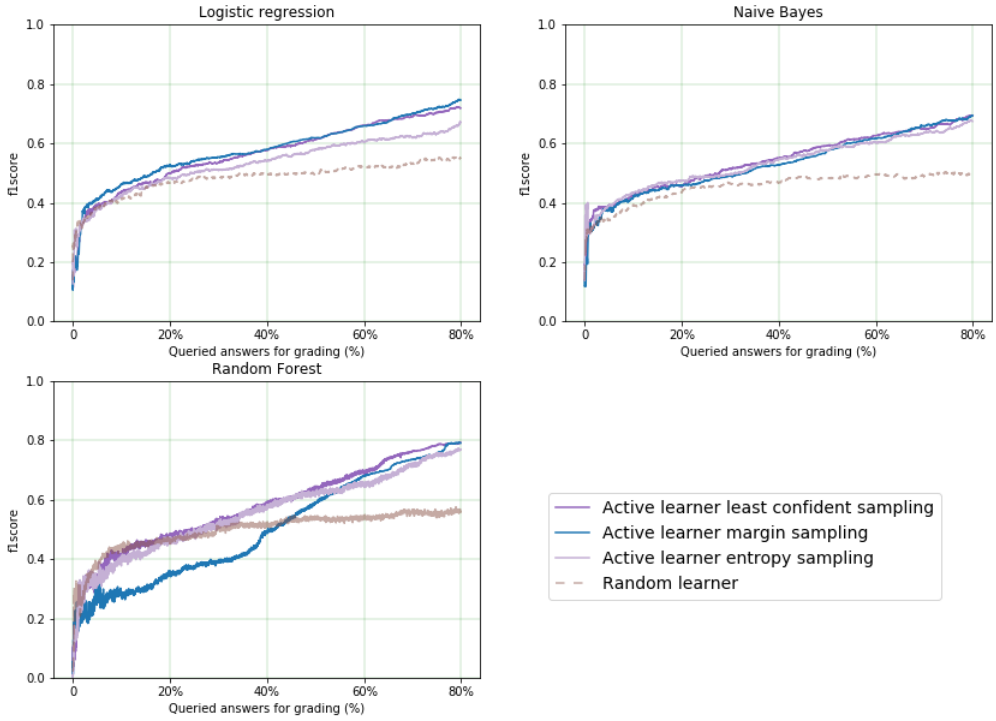


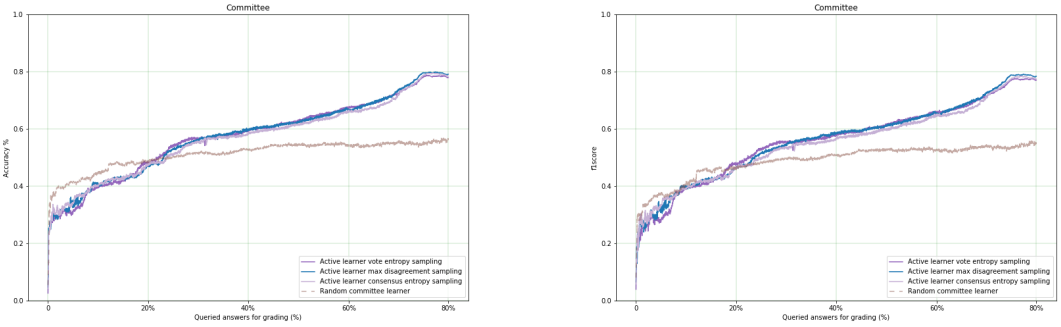
Figure 6.50: Results in terms of F1 score for different models with uncertainty based query strategies.

Logistic regression performs well during the first half queries and random forest performs well during the last stages.

	Time taken per query (secs)					
Models	**least confident uncertainty**	**margin based uncertainty**	**entropy based uncertainty**	**vote entropy**	**max disagreement**	**consensus entropy**
Logistic Regression	0.2076	0.1573	0.1423	3.7620	3.5757	3.5843
Naive Bayes	0.0938	0.0915	0.0910			
Random Forest (100 trees)	3.4522	3.3247	3.1755			

Table 6.8: Time comparison between different models for multi-class classification.

The time comparison of different models is tabulated in Table 6.8. Considering the bump plot and the time comparison, it could be inferred that logistic regression classifier with margin based uncertainty sampling fits best for this setting as random



(a) Results in terms of accuracy for committee based query strategies.

(b) Results in terms of F1 score for committee based query strategies.

Figure 6.51: Committee-based multi-class classification.

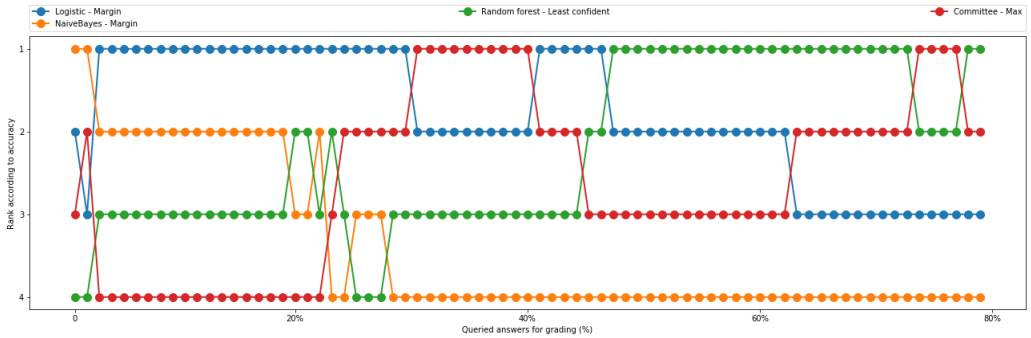


Figure 6.52: Bump chart of model performance in multi-class classification.

forest takes more than 3 seconds for every query which is a long waiting time in a practical application.

6.9 Experiment 9: SemEval 2013 Dataset using Term Frequency-Inverse Document Frequency (tf-idf) Features

In this experiment, the SemEval dataset was experimented using term frequency-inverse document frequency (tf-idf) features. Every answer was encoded into a vector of length 1908 and this vector was used as input to our models during training and prediction. Similar to the BOW features, tf-idf features also make up a huge feature space and thus, linear SVC and kernel based SVC were ignored in this experiment due to the time constraints.

6.9.1 Binary Classification

The experimental results of different model-query strategy combinations in a binary classification setting are shown in Fig 6.53 and 6.55a. The F1 score analysis of this experiment is shown in Fig 6.54 and Fig 6.55b.

We observe that all the active learning strategies in their respective models performed comparably to each other. Hence, we selected the margin uncertainty sampling for the naive Bayes, logistic regression, and random forest classifier models to have a closer look at their performance over increasing number of queries using a bump chart Fig 6.56. Logistic regression classifier performs well initially and random forest performs well after 35% of the queries.

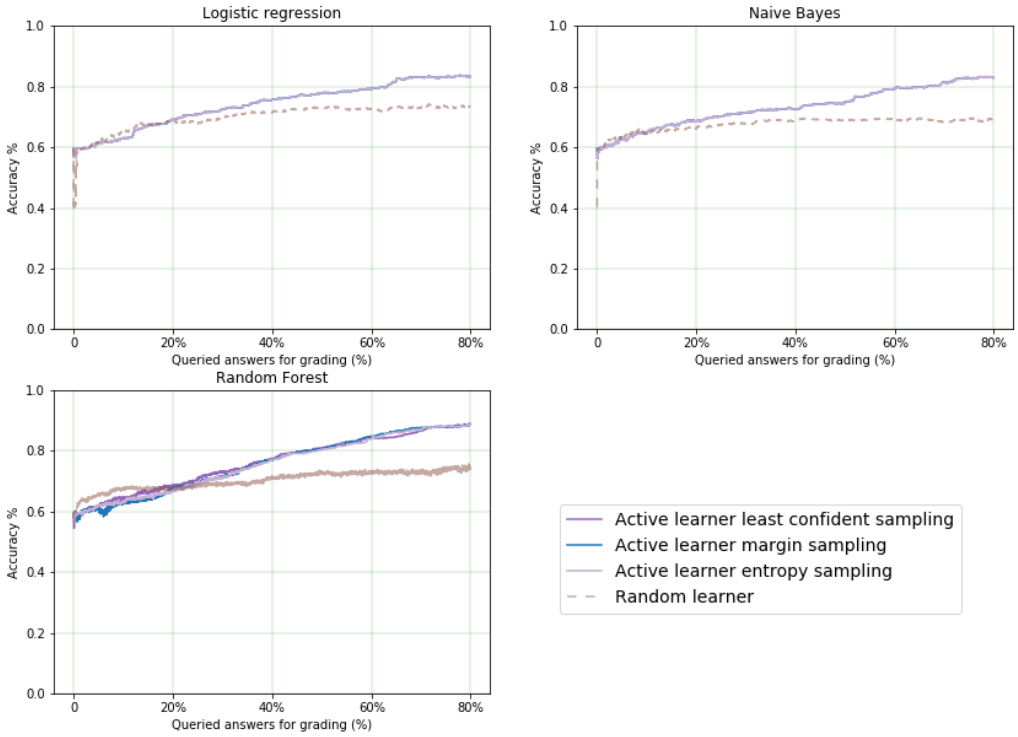


Figure 6.53: Results in terms of accuracy for different models with uncertainty based query strategies.

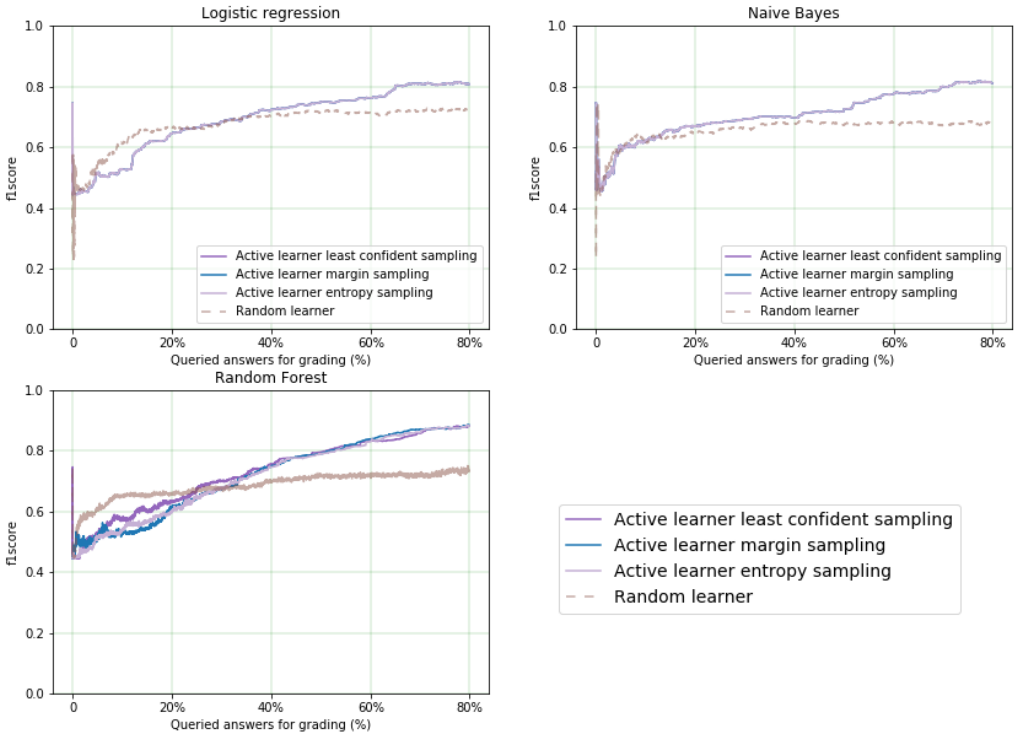
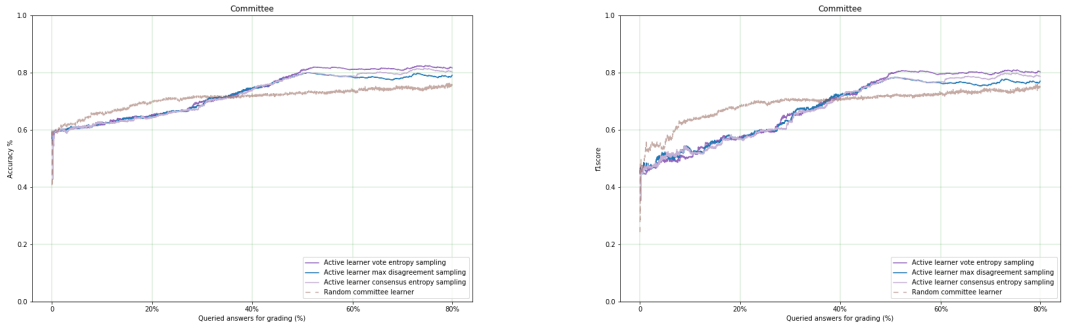


Figure 6.54: Results in terms of F1 score for different models with uncertainty based query strategies.



(a) Results in terms of accuracy for committee based query strategies.

(b) Results in terms of F1 score for committee based query strategies.

Figure 6.55: Committee-based binary classification.

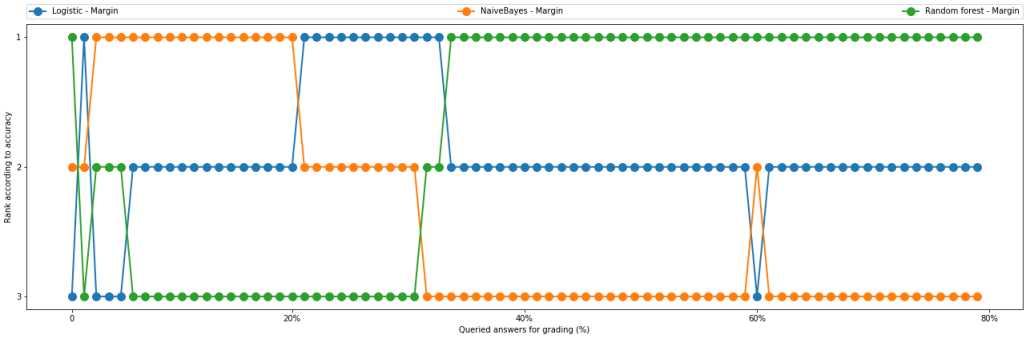


Figure 6.56: Bump chart of model performance in binary classification.

6.9.2 Multi-class Classification

The experimental results of different model-query strategy combinations in a multi-class classification setting are shown in Fig 6.57 and 6.59a. The results were also analyzed using the F1 score which are shown in Fig 6.58 and Fig 6.59b.

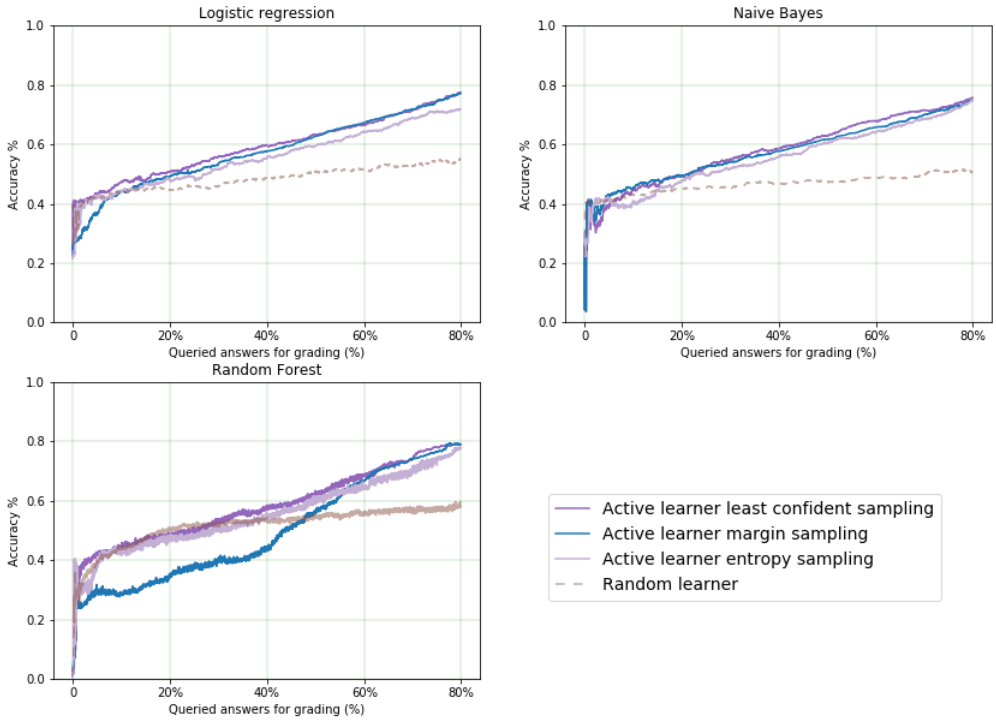


Figure 6.57: Results in terms of accuracy for different models with uncertainty based query strategies.

Considering the accuracy and the F1 scores, Logistic Regression with margin uncertainty sampling, Naive Bayes with margin uncertainty sampling, and Random Forest with margin uncertainty sampling were selected for comparison using a bump chart. In case of committee, max disagreement sampling performed better and is added to the bump chart comparison. This chart shows the performance of every model ranked by their accuracies which is shown in Fig 6.60. The bump chart shows

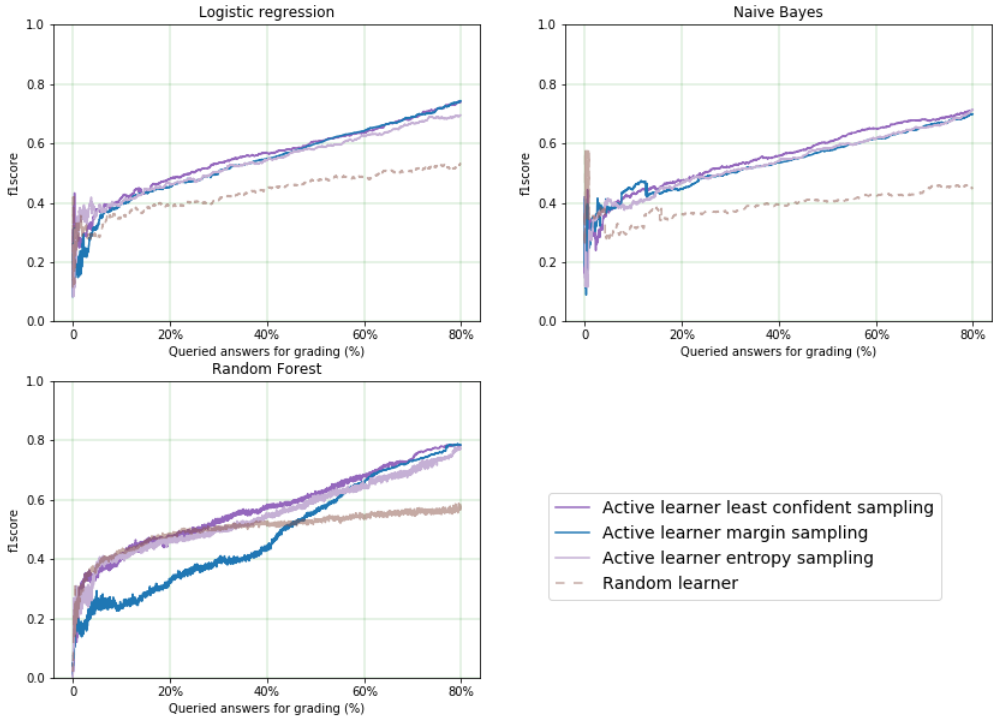


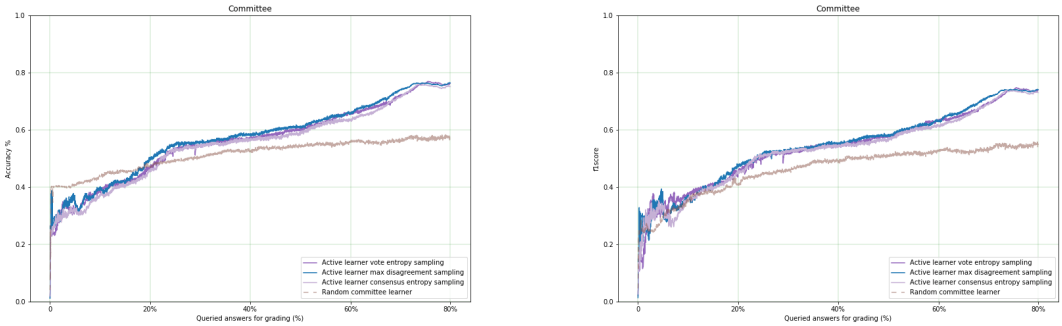
Figure 6.58: Results in terms of F1 score for different models with uncertainty based query strategies.

that the committee of models and logistic regression models perform comparable to each other over the whole range of queries.

The time comparison of different models is tabulated in Table 6.9. Considering the bump plot and the time comparison, it could be inferred that logistic regression classifier with margin based uncertainty sampling fits best for this setting. Committee based learning is ignored as it takes a lot of time.

	Time taken per query (secs)					
Models	**least confident uncertainty**	**margin based uncertainty**	**entropy based uncertainty**	**vote entropy**	**max disagreement**	**consensus entropy**
Logistic Regression	0.1322	0.1121	0.0997	4.0031	3.7025	3.6239
Naive Bayes	0.0630	0.0612	0.0627			
Random Forest (100 trees)	3.6896	3.5362	3.3289			

Table 6.9: Time comparison between different models for multi-class classification.



(a) Results in terms of accuracy for committee based query strategies.

(b) Results in terms of F1 score for committee based query strategies.

Figure 6.59: Committee-based multi-class classification.

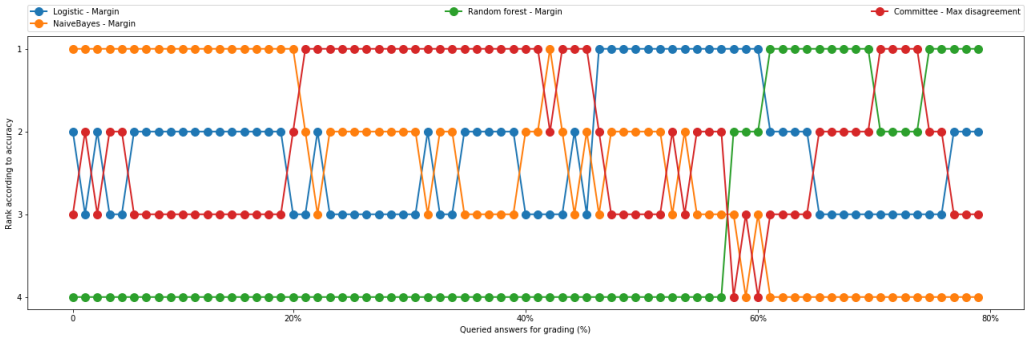


Figure 6.60: Bump chart of model performance in multi-class classification.

6.10 Experiment 10: Influence of seed and batch size

Batch size is one of the important setting which influences the performance of active learning. To evaluate the effectiveness of different batch sizes, the active learning setting that performed best on the previous experiments on the SemEval 2013 dataset was used in this experiment. Random forest model consisting of 100 trees with least confident uncertainty sampling was trained on 40% of SemEval's train dataset and then this model is used to evaluate the performance on the students' answers. The active learning is done with varying batch sizes and the accuracy on unseen answers test set have been tabulated in table 6.10.

Batch Size	1	3	5	7	10	20	50
Accuracy	**0.537**	0.492	0.505	0.494	0.511	0.516	0.518

Table 6.10: Accuracy for different batch sizes.

We can infer that querying the human expert the grades for one question at a time yields more accuracy than using higher batch sizes. The reason behind this could be that active learning choses samples from different regions in the sample space while querying one by one which in turns helps it to fix a better decision boundary between the classes. In contrast, querying in batches of size more than one may try to query samples from the same neighborhood which results in its poor performance [44].

Seeding is another important aspect of active learning as it defines the samples from which to learn the model initially. Different strategies for seeding exists such as random seeding (randomly selecting the samples), one sample per class, and multiple samples per class. We opted for one sample per class in our experiments as work by Horbach et al [44]., claimed that equal seeding has more impact on the performance of the machine learning model. Though one sample per class could easily be obtained from an existing dataset, efficient methods to find the seeds would be a challenge for a new set of answers and questions. Clustering the sample space with the grades as cluster numbers and seeding one sample per cluster could be efficient in such a setting.

6.11 Experiment 11: Question-wise analysis of the best setting.

The best active learning setting based on the previous experiments on the SemEval 2013 dataset was used in this experiment to observe the model's performance on different questions. The random forest model consisting of 100 trees with least confident uncertainty sampling was trained on 20% of SemEval's train dataset and then this model is used to evaluate the performance on the students' answers for every question. Fig 6.61 shows the accuracy obtained on every question using this model.

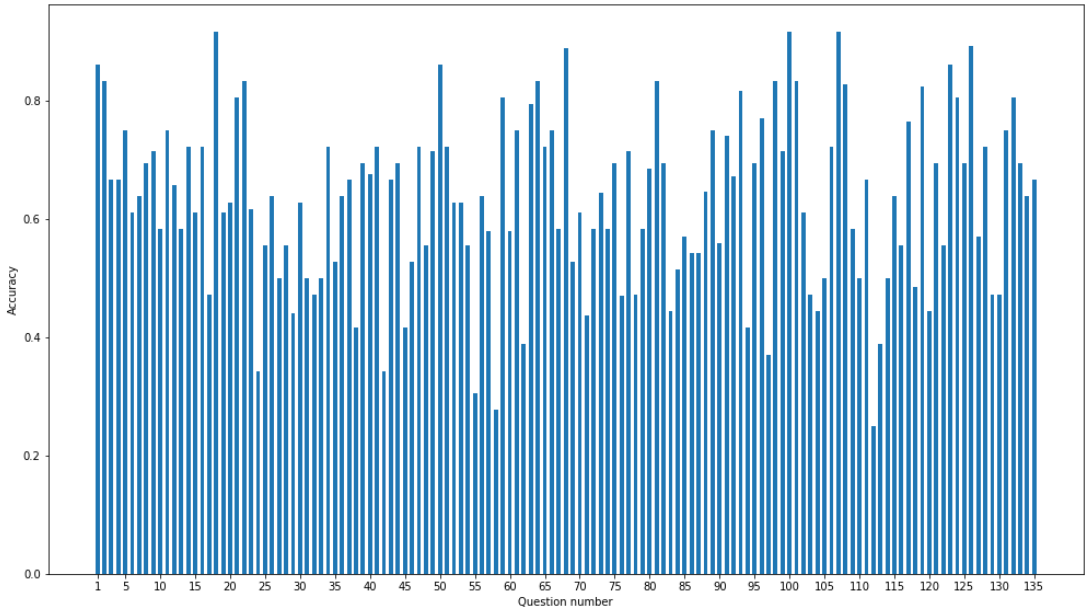


Figure 6.61: Accuracy obtained by active learning on every question of SemEval dataset.

It can be seen from the graph that the model doesn't perform consistently on every question. Following are the questions and their reference answers for which the model achieved more than 90% accuracy.

Question: Andi and Scott decided to investigate solar water heaters using collectors of different colors (but the same size) as one variable and covered versus uncovered as a second variable. Look at the graph of their data and answer the questions below. What does the graph tell you about the effect of using a cover?

Reference answer: Water heats up faster when covered.

Question: Paula wants to test a food to see if it contains acid and/or sugar. When Paula added water and yeast to a sample of the food and put the mixture in a warm water bath, the mixture began to fizz and bubble. What does this chemical reaction indicate?
Reference answer: The food contains sugar.

The questions are of closed type where the answers are very short and decisive. The features used in this experiment are capable of capturing such answers which contain similar wordings as the reference answers of paraphrases of it. This could be a possible reason behind the better performance of this model on these questions.

Following are the questions on which the model achieved less than 30% of accuracy.

Question: The graph shows temperature data from 2 containers. One container had 100 milliliters of water and the other had 100 milliliters of dry soil. Each container was placed in the sun for 20 minutes and then in the shade for 20 minutes. Which container, A or B, had the dry soil? Explain how the graph helped you decide which container had the dry soil.
Reference answer: A. Dry soil heats more quickly and cools off more quickly than water. The graph shows A heats and cools more quickly than B, so A must be the dry soil.

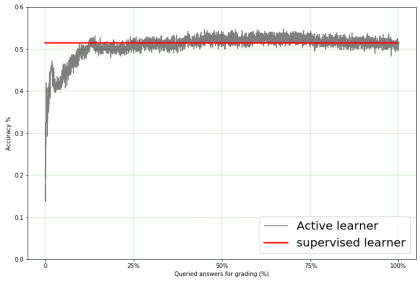
Question: Denise made a circuit to light a bulb or run a motor. She used a special switch. Below is the schematic diagram of her circuit. The switch is inside the dotted box. Can she run both the light and the motor at the same time? Why or why not?
Reference answer: No. Only one circuit can be complete at a time.

As can be seen from the questions above, these are mostly open type questions and the answers to these questions are heavily dependent on the students' perspectives. In contrast to the definitive quesitons, explaining why something works and interpreting an illustration vary much from person to person. This explains why the models were not successful in computing the grades for the answers to these questions.

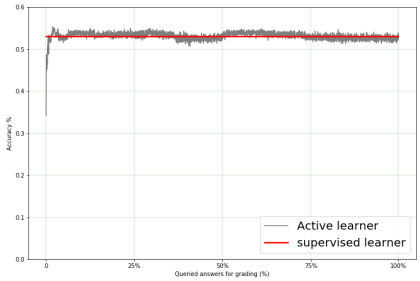
6.12 Experiment 12: Comparison of Active Learning with Supervised Learning.

SemEval 2013 dataset's three different test sets are used for the comparison of active learning with supervised learning. The training set includes the same dataset which is used for Experiments 7.7,7.8, and 7.9 and the test set consists of unseen answers to the same question which appeared in the training set, answers for unseen questions that belong to the same domain, and answers for questions from a different domain. Based on the previous experiments' results, features from Sultan et al [73]., were used to train a random forest classifier.

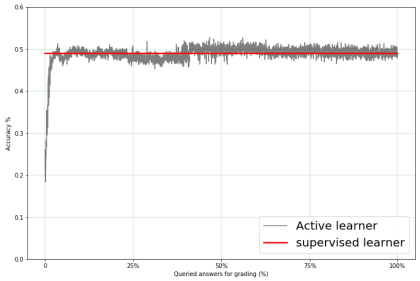
Least confident uncertainty based active learning strategy is used here as it has performed well in most of the scenarios with Sultan's features. The objective of this experiment is to observe the effectiveness of active learning with less training data compared to the supervised learning. Fig 6.62 shows how active learning performs compared to the supervised learning in three test datasets.



(a) Comparison for unseen answers.



(b) Comparison for unseen questions.



(c) Comparison for unseen domain.

Figure 6.62: Active learning vs. supervised learning.

As evident from the Fig 6.62 we can see that active learning reaches the same performance as supervised learning with less than 20% of the data. It is surprising that the model works better in unseen questions than in the unseen answers (for the same questions as in the training set). However the performance is inferior in the unseen domain case. Active learning performs efficiently as it choses the best samples to learn from earlier in the learning phase. As supervised learning doesn't choose the samples to learn from it needs a lot of data to achieve a good performance.

6.13 Discussions

Based on the experiments 6.1 to 6.9 the active learning strategy and the machine learning model which worked best in every dataset has been tabulated in table 6.11. Since all the models performed camparably to each other, no model was finalized for the Mohler dataset with Sultan features in binary classification task.

		Sultan	BOW	TF-IDF
Mohler	Binary	-	Naive Bayes - Margin	Naive Bayes - Margin
	Multi	Random Forest - Least Confident	Naive Bayes - Margin	Naive Bayes - Margin
NN	Multi	Random Forest - Least Confident	Random Forest - Margin	Random Forest - Margin
Sem-Eval	Binary	Logistic Regression - Margin	Random Forest - Margin	Random Forest - Margin
	Multi	Random Forest - Least Confident	Logistic Regression - Margin	Logistic Regression - Margin

Table 6.11: Best active learning settings on different datasets.

The main observations from experiments in this chapter are;

1. All active learning strategies outperformed random sampling which reinforces the fact that active learning strategies have much impact for this task of short answer grading.

2. Least confident based uncertainty sampling worked well in random forest classifier when features from Sultan et al [73]., were used. Sultan's features were found to perform better than BOW and tf-idf in SemEval and Mohler's dataset. This could be because Sultan's features capture the semantic properties of the text in addition to the syntactic features, while BOW and tf-idf work in the syntactic level. The shortcomings of Sultan's features are;

   - the task of getting the features is time-consuming. This hinders instant deployment of the grading system using these features on a new dataset in the GUI.

   - the features require reference answers for every questions to compute the similarity score and word alignment score. The quality of the reference answer has a very high impact on the results of the grading system.

3. Margin based uncertainty sampling worked well when bag of words or tf-idf features were used. Different models are seen to perform well in different datasets and thus selection of the best model heavily depends on the dataset on which it is used. Despite the model being used, margin based uncertainty sampling works well for features in high dimension.

4. Based on the time taken by every model which have been tabulated in the experiments above, naive Bayes and logistic regression take very less time when compared to random forests as it is an ensemble of 100 trees.

5. In addition to the query strategy, batch size and seeding are imperative to better performance of active learning. From experiment 6.10, it can be seen that batch size of 1 and equal seeding is suitable for short answer grading.

6. Question-wise analysis on the SemEval 2013 dataset from experiment 6.11 shows that the model with the best active learning setting doesn't work consistently on all types of questions. The model performs well on closed type of questions and worse on open type questions (the answers to which are very subjective in nature).

7. Experiment 6.12 was performed to prove that active learning can perform comparably to supervised learning with much less data. The results confirm that active learning performs with same level of performance as supervised learning and that it can not achieve more accuracy than that could be achieved with supervised learning.

8. Class imbalance is a common problem in most of the machine learning problems. Active learning is seen as better alternative to supervised learning for problems in which the data has moderate skewness [36]. This could also be seen as a reason behind active learning performing well with less data when compared to supervised learning in datasets with moderate skewness. However, high degree of skewness severely affects the learning models' performance as in the Mohler'11 dataset. The failure of active learning in selecting efficient samples under extreme class imbalance is also seen in other works [24].

9. Uncertainty based sampling performed better than committee based sampling in most of the experiments and committee of models took more time per query than using a single model. We recommend a study of best set of machine learning models to be included in a committee which would work well with reasonable time per query as a future work.

7

AI Assisted Grading System

The best active learning strategies (query strategy, and seed selection), features extracted from the answers, and machine learning model which were determined from the experimental results were incorporated into a Graphical User Interface (GUI) to be used by teachers or professors while grading. Initially, the system queries the grades for certain percentage of the total answers from the human grader, and then displays the notebooks in 'question-wise' view. The purpose of question-wise view is to make the task of assessing the answers easier for the humans. This GUI along with the functionality of each of its components have been discussed below.

7.1 Software Architecture

Fig 7.1 shows the architecture of the GUI implemented in this project. This system reads all the notebook files of the students from a specified directory, converts them into a Pandas dataframe, extracts the features from the answers needed for the machine learning algorithm, learns the model for the task of grading, gets the grades confirmed by a human and stores the grades along with human expert's feedbacks which is saved in another directory as Jupyter Notebook files. Tasks which include getting the notebook files, processing them through the machine learning algorithm and saving the grades with feedbacks occur in the backend while the display of the questions, answers, grades and feedbacks is taken care of by the front end.

The functionalities of the front end is done using Vue.js. This framework is used to create a web-based graphical user interface using the JavaScript programming

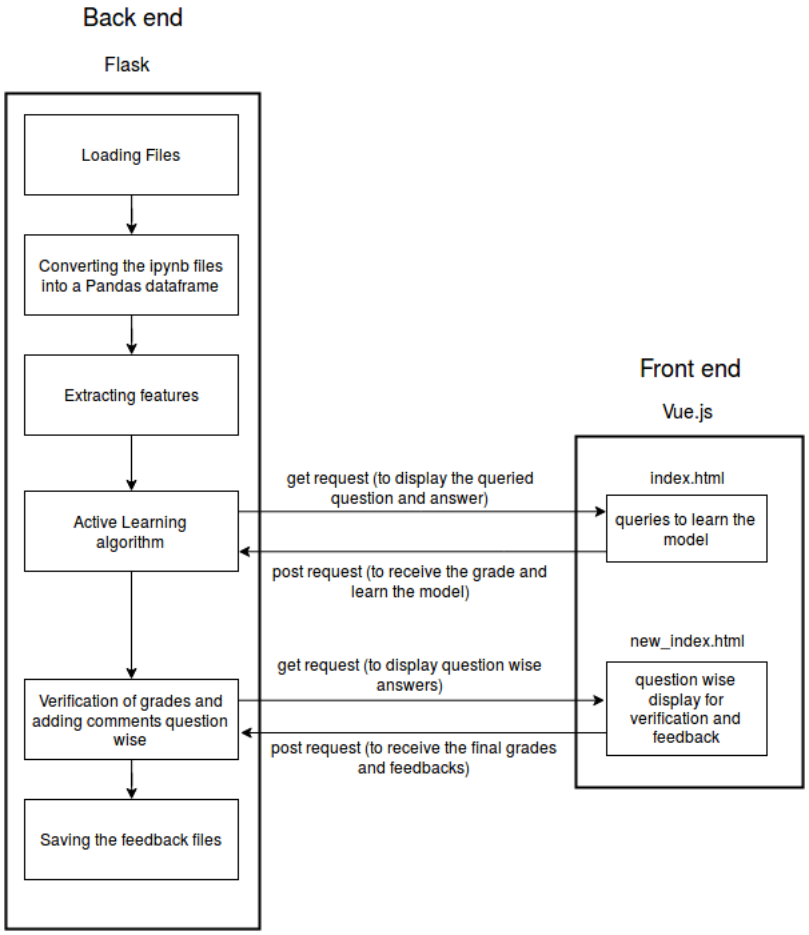


Figure 7.1: Architecture of the AI-assisted grading system (GUI).

language. In addition, the elements of the page and the display features were written using HTML5 and CSS. Flask is a web development microframework which uses Python programming language. Since the machine learning models, and active learning settings were coded in Python, Flask came in handy to deal with the back end operations.

Once the notebooks are converted into the Pandas dataframe along with its features, the whole dataset is fed into the active learning algorithm. Fig 7.2 shows the stage where the queried answers appear in the GUI interface for the professor to grade. Certain percentage of the answers are queried from the professor in this manner and the model is learned using these grades.

Once professor is done grading the initial set of answers queried through active

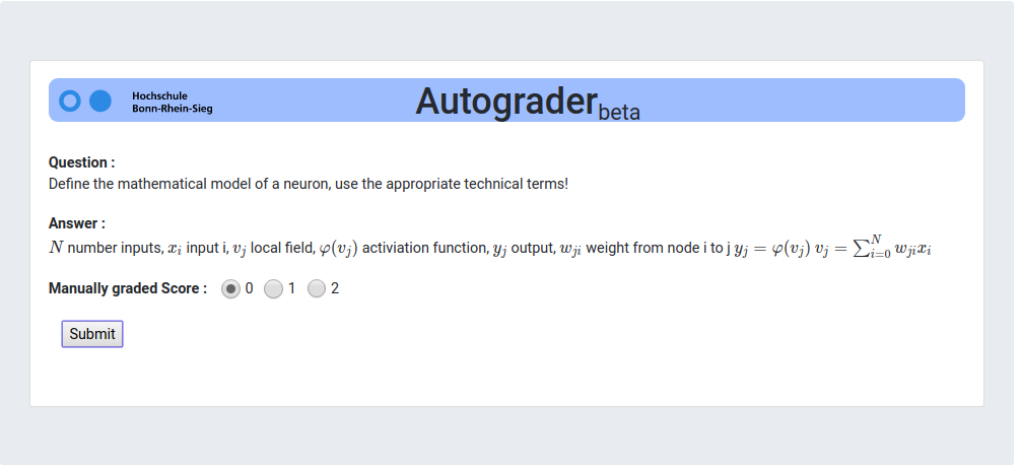
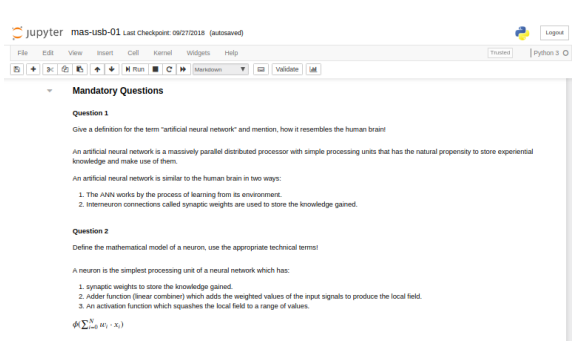


Figure 7.2: Querying stage of active learning in GUI.

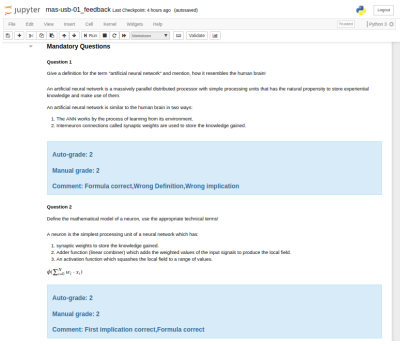
learning, the autograded scores are displayed along with the answers question by question. Fig 7.3 shows this stage where all the students' answers to the first question for the professor to check. The autograded scores are binded to the manual grade radio buttons to ease the effort of professors while grading. The proffesors could simply leave the grades as it is if he thinks that they are right or he has the option to change the grades in the manual grade section. In addition, the professor can add further feedbacks by checking the relevant opinions given under every question.

Once the professor has gone through all the questions, all the answers are saved back into jupyter notebook format. Fig 7.4a shows the initial format of the submitted students' answers in the notebook and Fig 7.4b shows the final format of the notebook once the final grades and feedbacks have been added to respective answers.

Figure 7.3: Question-wise view of answers after autograding stage.



(a) Submitted student's answers in the notebook.



(b) Final format of the notebook with grades and feedbacks

Figure 7.4: Initial and final versions of the notebook.

7.2 User Experience

7.2.1 Survey

Two users were asked to try grading a small percentage of answers and the survey answered by them is used to improve the interface and its functionalities. The survey questions and their answers by the users is given below.

User 1:

- How easy is our grading to use ?
  In a scale of 1-10 , I would rate it as 6.

- Did AI assisted grading reduce the effort of grading ?
  Yes. The difference was very clear. In manual grading I need to select the grades for all the answers and have to submit the grade. While in case of AI assisted grading most of the times I just need to verify the graded answer. Hence lots of effort for grading is reduced.

- How about the readability of Questions and answers ?
  The readability of question and answers was fine. Still, Various fonts can be tried to help the user find the better font for reading.

- How about the positioning and sizing of various elements (radio button,submit button)?
  Size of the radio button can be increased. Submit button can be on the right side as most of the users are right handed.

- What features you think need to be added to enhance the experience ?
  Same question with different student answers can be displayed in a single page.

User 2:

- How easy is our grading to use ?
  In a scale of 1-10 , I would rate it as 7.

- Did AI assisted grading reduce the effort of grading ?
  Yes.
- How about the readability of Questions and answers ?
  Readability was satisfactory.
- How about the positioning and sizing of various elements (radio button,submit button)?
  Positioning can be improved a bit as I have to move the cursor here and there for selection. Instead it would be great if the movement of the cursor was in a sequential manner.
- What you think need to be added to enhance the experience ?
  Lot can be added to make the user feel comfortable. Various display options can be created. Autograding score can be embedded in other options other than radio button. Shortcuts can be created.

7.2.2 Experiment: Number of clicks

A small experiment was conducted to realize the efficiency of using such an interface to grade the assignments with active learning in the background. One version of this experiment would be done by counting the total number of clicks while the grader grades all the answers for all the quesitons and the second version would be done with the support of active learning in the background, which computes the grades for all the questions once the learning phase is completed. The efficiency is judged by the reduction in the number of clicks required to grade all the answers in both the settings.

The experiments were done on all three datasets. All the students' answers in the notebooks are displayed question-wise in the active learning-free version. This would reflect the effort required to grade the assignments without using any assistance of any sort. In the version which uses active learning, the model is trained on the grades given by the grader for 25% of the answers and then the answers are displayed question-wise along with the computed score for every answer.

The grader has the choice of changing the grade if he thinks that the suggested score by the model is wrong and this grade would be saved in a database for later

converison into notebook. If the grader feels that the grade suggested by the model is reasonable, he could simply move on to the next student's answer to the same question. Viewing the answers question by question enables easier judgement of students' answers for the grader and faster grading. Random forest classifier with 100 trees and margin based uncertainty sampling was used in the learning process. The number of clicks made in both the learning-free version and learning assisted versions have been tabulated in Table 7.1.

Datasets	Clicks with active learning	Clicks without active learning
Neural network	338	680
SemEval 2013	3527	5104
Mohler'11	1386	2352

Table 7.1: Number of clicks required to grade the answers with and without active learning.

As can be seen from the table above, the grading process assisted by active learning massively reduces the effort and time of the grader. In addition, the grader could spend more time in constructing useful feedbacks for the students by ticking the pertinent check boxes in the feedback section of the interface below every answer.

8

Conclusions

8.1 Contributions

The main contributions of this work can be enumerated as follows;

1. A detailed study and evaluation of different active learning strategies, features extracted from the student answers, machine learning models and active learning settings on three different datasets.

2. A web-based graphical user interface (GUI) which can be used for the task of grading the answers with the help of active learning. This system is flexible as it can work with different machine learning classifiers, and different active learning settings on the students' answers in a Jupyter notebook format. With little modifications it could be fit to work on any dataset in a csv or Pandas dataframe format. This will be made available as an open-source project to which future researchers can contribute or make use of.

3. Mohler'11, Neural network, and SemEval 2013 datasets were cleaned in preparation to be used in a classifier and features such as word embeddings, cosine similarity between student and reference answers, aligned words, length ratio, bag of words, and tf-idf were also added. These datasets are available in csv, and Pandas Dataframe formats which could be used in future research works.

8.2 Lessons learned

We overcame many challenges during the implementation of this project. These are enumerated below.

1. Answers with more latex codes were difficult to make sense of, and this affected the performance a lot.

2. Questions with answers in English and German languages made extracting features more complex as the new German words would add to the dimensions of bag of words and tf-idf encodings which is of insignificant use.

3. It was hard to suggest a single active learning setting which would work for different datasets. Different combinations of query strategies and machine learning models worked well on different datasets and this made it finalizing a single best setting.

8.3 Future work

During the period of our research, we realized many avenues to be explored in every stages of the experimental pipeline, which are listed below.

1. Features are an imperative aspect of any machine learning task. Thus, the best features to be extracted for better performance of the grading system is still an open research area. Features such as sentence embeddings and latex embeddings could be a good place to start while thinking of feature engineering. Making sense of latex codes proves vital for the performance of math related assessment answers. Active learning can achieve the supervised learning's performance using lesser data when better features are used for learning.

2. Efficient active learning strategies to deal with class imbalance is an active research topic. As skewed grade distribution is ubiquitous in most of the real world scenarios, better active learning settings to deal with such skewness could be looked into as a solution for better performance.

3. An efficient method for giving feedbacks should be worked on. The current version enables the grader to give feedbacks one by one. Clustering the similar

answers and giving the same feedback to all of them would be a better choice and would reduce the effort of the grader.

4. More functionalities could be added to the GUI according to the users' requirements. The current version is a basic one which concentrates on giving the best possible grades for the answers. The design of the interface could be optimized for simpler use.

References

[1] Chatbots with seq2seq. http://suriyadeepan.github.io/2016-06-28-easy-seq2seq/. Accessed: 2018-08-27.

[2] Active learning. https://en.wikipedia.org/wiki/Active_learning_(machine_learning), . Accessed: 2018-05-28.

[3] Active learning curious ai algorithms. https://www.datacamp.com/community/tutorials/active-learning, . Accessed: 2018-05-28.

[4] A gentle introduction to the bag-of-words model. https://machinelearningmastery.com/gentle-introduction-bag-words-model/. Accessed: 2018-11-22.

[5] Overview of text similarity metrics in python. https://towardsdatascience.com/overview-of-text-similarity-metrics-3397c4601f50. Accessed: 2018-11-24.

[6] displacy dependency visualizer. https://explosion.ai/demos/displacy. Accessed: 2018-11-22.

[7] fasttext. https://research.fb.com/fasttext/. Accessed: 2018-08-27.

[8] Glove: Global vectors for word representation. https://nlp.stanford.edu/projects/glove/. Accessed: 2018-08-27.

[9] A general approach to preprocessing text data. https://www.kdnuggets.com/2017/12/general-approach-preprocessing-text-data.html. Accessed: 2018-08-27.

[10] modal: A modular active learning framework for python3. https://modal-python.readthedocs.io/en/latest/. Accessed: 2018-11-22.

[11] Named entity recognition: Applications and use cases. https://towardsdatascience.com/named-entity-recognition-applications-and-use-cases-acdbf57d595e. Accessed: 2018-11-22.

[12] Applications of natural language processing. https://www.lifewire.com/applications-of-natural-language-processing-technology-2495544. Accessed: 2018-08-27.

[13] Natural language toolkit. https://www.nltk.org/, . Accessed: 2018-08-27.

[14] Nltk's list of english stopwords. https://gist.github.com/sebleier/554280, . Accessed: 2018-08-27.

[15] Numpy. http://www.numpy.org/. Accessed: 2018-08-27.

[16] Python data analysis library. https://pandas.pydata.org/. Accessed: 2018-08-27.

[17] Partofspeech tagging. https://en.wikipedia.org/wiki/Part-of-speech_tagging. Accessed: 2018-11-22.

[18] Get busy with word embeddings an introduction. https://www.shanelynn.ie/get-busy-with-word-embeddings-introduction/. Accessed: 2018-08-27.

[19] spacy. https://spacy.io/. Accessed: 2018-08-27.

[20] Finding the most important sentences using nlp and tf-idf. https://hackernoon.com/finding-the-most-important-sentences-using-nlp-tf-idf-3065028897a3, . Accessed: 2018-11-22.

[21] tf-idf. https://en.wikipedia.org/wiki/Tf%E2%80%93idf, . Accessed: 2018-11-22.

[22] word2vec. https://code.google.com/archive/p/word2vec/. Accessed: 2018-08-27.

[23] Evaluation of semantic textual similarity approaches for automatic short answer grading. WS17 H-BRS - Evaluation of Semantic Textual Similarity Approaches for Automatic Short Answer Grading Ploeger, Nair supervising, January 2017/18.

[24] Josh Attenberg and Seyda Ertekin. Class imbalance and active learning. *H. He, & Y. Ma, Imbalanced Learning: Foundations, Algorithms, and Applications*, pages 101–149, 2013.

[25] Lyle F Bachman, Nathan Carr, Greg Kamei, Mikyung Kim, Michael J Pan, Chris Salvador, and Yasuyo Sawaki. A reliable approach to automatic assessment of short answer free responses. *Proceedings of the 19th international conference on Computational linguistics* -, 2:1–4, 2002. doi: 10.3115/1071884.1071907.

[26] Sumit Basu, Chuck Jacobs, and Lucy Vanderwende. Powergrading: a clustering approach to amplify human effort for short answer grading. *Transactions of the Association for Computational Linguistics*, 1:391–402, 2013.

[27] Steven Burrows, Iryna Gurevych, and Benno Stein. *The eras and trends of automatic short answer grading*, volume 25. 2015. ISBN 4059301400268. doi: 10.1007/s40593-014-0026-8.

[28] David Callear, Jenny Jerrams-Smith, and Victor Soh. Caa of short non-mcq answers. 2001.

[29] Kathryn Chaloner and Isabella Verdinelli. Bayesian experimental design: A review. *Statistical Science*, pages 273–304, 1995.

[30] David A Cohn, Zoubin Ghahramani, and Michael I Jordan. Active learning with statistical models. In *Advances in neural information processing systems*, pages 705–712, 1995.

[31] Ido Dagan and Sean P Engelson. Committee-based sampling for training probabilistic classifiers. In *Machine Learning Proceedings 1995*, pages 150–157. Elsevier, 1995.

[32] Tivadar Danka and Peter Horvath. modAL: A modular active learning framework for Python. URL https://github.com/cosmic-cortex/modAL. available on arXiv at https://arxiv.org/abs/1805.00979.

[33] Marie-Catherine De Marneffe and Christopher D Manning. Stanford typed dependencies manual. Technical report, Technical report, Stanford University, 2008.

[34] Dmitriy Dligach and Martha Palmer. Good seed makes a good crop: accelerating active learning using language modeling. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 6–10. Association for Computational Linguistics, 2011.

[35] Myroslava O Dzikovska, Rodney D Nielsen, Chris Brew, Claudia Leacock, Danilo Giampiccolo, Luisa Bentivogli, Peter Clark, Ido Dagan, and Hoa T Dang. Semeval-2013 task 7: The joint student response analysis and 8th recognizing textual entailment challenge. Technical report, NORTH TEXAS STATE UNIV DENTON, 2013.

[36] Seyda Ertekin, Jian Huang, and C Lee Giles. Active learning for class imbalance problem. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 823–824. ACM, 2007.

[37] Rosa L Figueroa, Qing Zeng-Treitler, Long H Ngo, Sergey Goryachev, and Eduardo P Wiechmann. Active learning for clinical text classification: is it better than random sampling? *Journal of the American Medical Informatics Association*, 19(5):809–816, 2012.

[38] Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. Ppdb: The paraphrase database. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 758–764, 2013.

[39] Stuart Geman, Elie Bienenstock, and René Doursat. Neural networks and the bias/variance dilemma. *Neural computation*, 4(1):1–58, 1992.

[40] Yoav Goldberg. Neural network methods for natural language processing. *Synthesis Lectures on Human Language Technologies*, 10(1):1–309, 2017.

[41] Wael H Gomaa and Aly A Fahmy. Short answer grading using string similarity and corpus-based similarity. *International Journal of Advanced Computer Science and Applications (IJACSA)*, 3(11), 2012.

[42] Michael Hahn and Detmar Meurers. Evaluating the meaning of answers to reading comprehension questions a semantics-based approach. In *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*, pages 326–336. Association for Computational Linguistics, 2012.

[43] Derrick Higgins, Jill Burstein, and Daniel Marcu. Evaluating Multiple Aspects of Coherence in Student Essays. *Hlt-Naacl*, pages 185–192, 2004.

[44] Andrea Horbach, Alexis Palmer, and Leibniz Sciencecampus. Investigating Active Learning for Short-Answer Scoring. pages 301–311, 2016.

[45] Sally Jordan and Tom Mitchell. e-assessment for learning? the potential of short-answer free-text questions with tailored feedback. *British Journal of Educational Technology*, 40(2):371–385, 2009.

[46] Dan Jurafsky and James H Martin. *Speech and language processing*, volume 3. Pearson London, 2014.

[47] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

[48] Sachin Kumar, Soumen Chakrabarti, and Shourya Roy. Earth movers distance pooling over siamese lstms for automatic short answer grading. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 2046–2052, 2017.

[49] Claudia Leacock and Martin Chodorow. C-rater: Automated scoring of short-answer questions. *Computers and the Humanities*, 37(4):389–405, 2003. ISSN 00104817. doi: 10.1023/A:1025779619903.

[50] David D Lewis and Jason Catlett. Heterogeneous uncertainty sampling for supervised learning. In *Machine Learning Proceedings 1994*, pages 148–156. Elsevier, 1994.

[51] Ou Lydia Liu, Joseph A Rios, Michael Heilman, Libby Gerard, and Marcia C Linn. Validation of automated scoring of science assessments. *Journal of Research in Science Teaching*, 53(2):215–233, 2016.

[52] Oliver Mason and Ian Grove-Stephensen. Automated free text marking with paperless school. 2002.

[53] Andrew Kachites McCallumzy and Kamal Nigamy. Employing em and pool-based active learning for text classification. In *Proc. International Conference on Machine Learning (ICML)*, pages 359–367. Citeseer, 1998.

[54] Andrew Kachites McCallumzy and Kamal Nigamy. Employing em and pool-based active learning for text classification. In *Proc. International Conference on Machine Learning (ICML)*, pages 359–367. Citeseer, 1998.

[55] Detmar Meurers, Ramon Ziai, Niels Ott, and Janina Kopp. Evaluating answers to reading comprehension questions in context: Results for german and the role of information structure. In *Proceedings of the TextInfer 2011 Workshop on Textual Entailment*, pages 1–9. Association for Computational Linguistics, 2011.

[56] Tom Mitchell, Terry Russell, Peter Broomhead, and Nicola Aldridge. Towards robust computerised marking of free-text responses. 2002.

[57] M. Mohler, R. Bunescu, and R. Mihalcea. Learning to grade short answer questions using semantic similarity measures and dependency graph alignments. pages 752–762, 2011.

[58] Michael Mohler and Rada Mihalcea. Text-to-text Semantic Similarity for Automatic Short Answer Grading. *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics (EACL '09)*, (April):567–575, 2009. doi: 10.3115/1609067.1609130.

[59] Rodney D Nielsen, Wayne H Ward, James H Martin, and Martha Palmer. Annotating students' understanding of science concepts. In *LREC*, 2008.

[60] Rodney D Nielsen, Wayne Ward, and James H Martin. Recognizing entailment in intelligent tutoring systems. *Natural Language Engineering*, 15(4):479–501, 2009.

[61] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[62] Stephen G. Pulman and Jana Z. Sukkarieh. Automatic short answer marking. *EdAppsNLP 05 Proceedings of the second workshop on Building Educational Applications Using NLP*, (June):9–16, 2005. doi: 10.3115/1609829.1609831.

[63] Lakshmi Ramachandran, Jian Cheng, and Peter Foltz. Identifying Patterns For Short Answer Scoring Using Graph-based Lexico-Semantic Text Matching. *Proceedings of the Tenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 97–106, 2015. doi: 10.3115/v1/W15-0612.

[64] Sebastian Raschka. *Python machine learning*. Packt Publishing Ltd, 2015.

[65] Shourya Roy, Sandipan Dandapat, Ajay Nagesh, and Narahari Y. Wisdom of Students: A Consistent Automatic Short Answer Grading Technique. *Proceedings of the 13th International Conference on Natural Language Processing*, pages 178–187, 2016.

[66] Tobias Scheffer, Christian Decomain, and Stefan Wrobel. Active hidden markov models for information extraction. In *International Symposium on Intelligent Data Analysis*, pages 309–318. Springer, 2001.

[67] Burr Settles. Active Learning Literature Survey. *Machine Learning*, 15(2): 201–221, 2010. ISSN 00483931. doi: 10.1.1.167.4245.

[68] Burr Settles and Mark Craven. An analysis of active learning strategies for sequence labeling tasks. In *Proceedings of the conference on empirical methods in natural language processing*, pages 1070–1079. Association for Computational Linguistics, 2008.

[69] Burr Settles, Mark Craven, and Soumya Ray. Multiple-instance active learning. In *Advances in neural information processing systems*, pages 1289–1296, 2008.

[70] Claude Elwood Shannon. A mathematical theory of communication. *Bell system technical journal*, 27(3):379–423, 1948.

[71] Raheel Siddiqi, Christopher J Harrison, and Rosheena Siddiqi. Improving teaching and learning through automated short-answer marking. *IEEE Transactions on Learning Technologies*, 3(3):237–249, 2010.

[72] Md Arafat Sultan, Steven Bethard, and Tamara Sumner. Back to basics for monolingual alignment: Exploiting word similarity and contextual evidence. *Transactions of the Association for Computational Linguistics*, 2:219–230, 2014.

[73] Md Arafat Sultan, Cristobal Salazar, and Tamara Sumner. Fast and Easy Short Answer Grading with High Accuracy. *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1070–1075, 2016. doi: 10.18653/v1/N16-1123.

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

[75] Hao Chuan Wang, Chun Yen Chang, and Tsai Yen Li. Assessing creative problem-solving with automated text grading. *Computers and Education*, 51(4): 1450–1466, 2008. ISSN 03601315. doi: 10.1016/j.compedu.2008.01.006.

[76] Bishan Yang, Jian-Tao Sun, Tengjiao Wang, and Zheng Chen. Effective multi-label active learning for text classification. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 917–926. ACM, 2009.

[77] Torsten Zesch, Michael Heilman, and Aoife Cahill. Reducing annotation efforts in supervised short answer scoring. In *Proceedings of the Tenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 124–132, 2015.