Minimizing The Costs in Generalized Interactive Annotation Learning

Shilpa Arora

CMU-LTI-12-013

Language Technologies Institute
School of Computer Science
Carnegie Mellon University
5000 Forbes Ave., Pittsburgh, PA 15213
www.lti.cs.cmu.edu

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy In Language and Information Technologies

Abstract

Supervised learning involves collecting unlabeled data, defining features to represent an instance, obtaining annotations for the unlabeled instances, and learning a classifier from the annotated data. Each of these steps has an associated cost. In this thesis, our goal is to reduce the total cost for the desired performance in supervised learning. Specifically, we focus on reducing the cost of feature engineering and the total annotation cost.

An instance in supervised learning is represented by a feature vector. For a text instance, a bag-of-words feature representation is commonly used, since word segmentation is straightforward in most languages, and word features have been found to provide good performance for several learning tasks. However, words are limited in the information they provide about the meaning of a text. Hand-crafted structured features based on linguistic annotations, such as parts of speech, semantic roles, syntactic parse trees, etc., have been found to improve performance beyond bag-of-words features. However, such manually engineered features require substantial effort from the expert. In this work, we propose a generic annotation graph representation for linguistic annotations, and use a frequent subgraph mining algorithm to automatically extract structured features from the annotation graphs. For a sentiment classification task and a protein-protein interaction extraction task, we show that these automatically extracted structured features provide a significant improvement in performance over bag-of-words features.

Training a classifier involves learning a function of the features to approximate the target variable. To learn a good approximation, several labeled instances are needed. Labeling an instance may require substantial annotation effort, called the *annotation cost*. In order to reduce the total annotation cost for the desired performance, in addition to labeling the instances, the user could provide information about the features directly. Direct feedback on features has been shown to reduce the total number of labeled instances required to achieve the desired performance. However, such feedback is restricted to simple features, such as words. Linguistic features, hand-crafted or automatically extracted, are often difficult to visualize and present to the user for feedback. To represent an image, features such as pixel values are commonly used. The user may not be familiar with such features to give feedback on them. An alternative is for the user to indicate parts of an instance that are *rationales* for its class label. For example, what sentences in a document, segments in an image, and scenes in a video, are rationales for their class label. Rationales provide indirect feature feedback, since features that overlap with the rationales should be important for the classification task and this indication is only indirect. Annotating rationales may incur additional cost, which may vary across annotators, instances, annotation tasks, user interface design, etc. We compare the two annotation strategies of providing instance's label only (LO) and instance's label together with rationales (LR), for different additional costs for annotating rationales. For a sentiment classification task and an aviation incident cause identification task, we show that rationales provide better performance for a given annotation cost, when annotating them incurs a small extra cost.

Annotation cost may vary across instances, annotators and annotation strategies. Annotation cost is often not known *a priori*, but it can be estimated. An estimate of the annotation cost can be used to selectively query the annotator, in order to directly minimize the total annotation cost for the desired performance. We propose a supervised regression model that uses the characteristics of an instance, annotator and annotation strategy for estimating the annotation cost in a multi-annotator environment with indirect feature feedback through rationales. For data collected from multiple annotators for a sentiment classification task, we show that an annotation cost estimate from the proposed approach outperforms simpler estimates based on any one of these characteristics.

Each instance (with or without rationales) may provide different incremental value to the learning algorithm. Annotation cost may also vary across instances and annotation strategies. We propose a cost-sensitive active learning approach, where in each iteration an instance is actively selected for a given strategy. We show that this strategy sensitive active instance selection approach for seeking rationales performs better than seeking rationales for instances selected randomly or actively, independent of the strategy. When the cost for annotating rationales is high, rationales may not be beneficial for all instances, and we may want to selectively ask for rationales. We further extend the proposed approach to jointly select the best instance and strategy in each iteration. For a sentiment classification task and an aviation incident cause identification task, we show that the proposed approach outperforms cost-sensitive and cost-insensitive instance selection for a fixed strategy, and cost-insensitive joint selection of instance and strategy, considering different additional costs for annotating rationales. While the best fixed strategy among LO and LR varies with the additional cost for annotating rationales, we show that the proposed approach performs as well as or better than the best fixed strategy, for different additional costs for annotating rationales.

The benefit from feature feedback (direct or indirect) may vary across learning problems. For two sentiment classification tasks with different instance granularity and a webpage category classification task, and several variations in the feature space, instance selection criteria, and signal to noise ratio, we show how the maximum benefit from feature feedback varies with these characteristics of a learning problem. We also show that measures for quantifying the complexity of a learning problem have a significant correlation with the amount of benefit from feature feedback.

To my family.

Acknowledgements

First and foremost, I would like to thank my adviser Dr. Eric Nyberg for giving me the opportunity to pursue research in this area. He introduced me to the topic of this thesis and guided me through this research. He always gave me the freedom to explore my own ideas. I am thankful to him for putting his confidence in me and encouraging me in difficult times. I learned several academic skills from him - scientific writing, research methods, teaching skills, presentation skills, etc. I am grateful to him for his help and patience through this learning process.

I would also like to thank my thesis committee members Dr. Carolyn P. Rosé, Dr. Pinar Donmez and Dr. Jaime Carbonell. I have worked with Carolyn since the initial years of my degree and co-authored several papers with her. I learned a lot from her about statistical analysis, user experiments, and scientific writing. I am thankful to her for her help with the research for this thesis and beyond. I am very grateful to Pinar for numerous long-distance discussions over emails and phone for shaping up the part of my thesis on active learning. I am also thankful to Jaime for his feedback on this work at different stages of the thesis.

I am also thankful to colleagues at CMU for helpful discussions for this research: Mahesh Joshi, Abhay Harpale, Burr Settles, Vamshi Ambati, Elijah Mayfield, Gaurav Veda. Thanks to Mahesh and Elijah for collaborating on projects and co-authoring research papers with me.

I would also like to thank the researchers whose work this thesis follows up on, for discussions and providing the data/tools. Special thanks to Omar Zaidan, Xifeng Yan, Sudheendra Vijayanarasimhan, Omid Madani, Hema Raghavan, Muhammad Arshad Abedin, Lijie Ren and Lei Zhao.

Beyond this thesis, I also got an opportunity to work on several research projects at CMU. One of major projects I worked on was the Machine Reading project. I would like to thank Chris Welty, Siddharth Patwardhan and James Fan from IBM research, for a great learning experience as part of this project. I would also like to thank Hideki Shima, Teruko Mitamura, Eric Riebling, Andy Schlaikjer, Matthew Bilotti, Manas Pathak, at CMU for collaboration on several research projects. I am thankful to my long time officemate Hideki Shima, for sharing the office with me and for several technical discussions. Thanks to my other office mates: Nico Schlaefer and Rui Liu.

I would like to thank my friends and colleagues from the time as a Masters student at CMU: Sachin Agarwal, Sameer Badaskar, Suresh Ganapathy, Abhaya Agarwal, Dipanjan Das, Abhay Harpale, Mahesh Joshi, Anagha Kulkarni, Frank Lin, Hideki Shima, Ni Lao, Mengqui Wang, Wei Chen, for discussions and collaboration on several class projects. I am also thankful to other friends at CMU and outside: Barkha Harpale, Sonia Singhal, Anshul Nigham, Gaurav Veda, Satanjeev Banerjee, Namrata Malik, Mehrbod Sharifi, Carol Sisson, Kriti Puniyani, Meghana Kshirsagar, Bhavana Dalvi, Yi-Chia Wang, Narges Razavian, for their friendship and support. Special thanks to my long time roommate and close friend Usha Kuppuswamy. Thanks to my other roommates: Archi Agarwal, Vini Shakti, Geeta Shroff, Ashwati Krishnan.

I am also thankful to LTI staff members - Stacey Young, Radha Rao, Mary Jo Bensasi, Linda Hanger, and Corinne Meloni, etc., for their help with administrative tasks and for answering queries about requirements, travel, etc.

This thesis would not have been completed without the love and support of my family. I would like to begin by thanking my parents: Ravinder and Saroj Arora. I am grateful to them for their unconditional love, emotional support and encouragement through these years in school. I am also very thankful to my brother, Rahul Arora, for his love and support through this journey. Even though younger to me by several years, he has always been my closest friend. I would also like to thank my sister and brother in-law, Priyanka and Sanjay Luthra, for providing a home away from home in Canada, and for giving me my niece, Samaira, whose photo in my phone made me smile in the toughest of times. I am also grateful to my parents in-law, Desh and Renu Batra, and brother in-law, Rahul Batra, for their love and emotional support through these times.

Finally, I would like to thank my dear husband, Gautam Batra, for his unconditional love and emotional support through this journey. Without his support, I would not have been able to complete this thesis. I am thankful to him for his encouragement in difficult times and sacrifices for my busy schedule. I am also thankful to him for moving to Pittsburgh. I share this accomplishment with him.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

An *annotation* is a property of an instance, for example, whether a given text expresses positive or negative sentiment, an image contains an object of interest or not, etc. A *target annotation* is an annotation that we wish to learn. In this thesis, we focus on supervised annotation learning, where a domain expert labels a set of instances with the target annotations, and a model is learned from these examples (instance and annotation pair). The learned model is used to predict annotations for unlabeled instances. There are several costs associated with supervised annotation learning. Turney (2000) provides a taxonomy of various costs in supervised learning, some of which we study in this thesis. Figure 1.1 shows the main steps involved in supervised annotation learning. The first step is to collect and prepare the unlabeled data. The associated cost for this step may not be trivial. For example, when annotating clinical records, we may need to remove the personal information contained in them, and the associated cost may be substantial. The second step is to define the features for representing the instances, which may require substantial effort from the domain experts. The next step is to obtain the target annotations for the training instances, which may require substantial annotation effort from the experts. Lastly, we learn a model from the annotated data. Apart from the developer's effort in designing the learner, this also often requires substantial computation cost.

In this thesis, our goal is to reduce the total cost required to achieve the desired performance in supervised annotation learning. Specifically, we focus on reducing the cost of feature engineering, and the total annotation cost. To reduce the cost of feature engineering, we propose an automatic approach to extract features from given properties of an instance. In order to reduce the total annotation cost, we propose an alternate annotation strategy, where in addition to providing an instance's label, the user identifies parts of the instance that are *rationales*, i.e. key indicators, for its class label. Annotation cost may vary across instances. Additional cost for rationales may also vary across instances, annotators, annotation tasks, user interface designs, etc. Each instance (with or without rationales) may provide different incremental value to the learning algorithm. We propose a cost-sensitive active learning approach, where in each iteration, an instance and annotation strategy (instance's label with or without rationales) are jointly selected as those expected to provide the maximum improvement in performance for the given annotation cost.

Figure 1.2 presents the proposed interactive annotation learning framework. In this framework, features are automatically extracted from prior linguistic annotations (Steps 2 and 3). In each iteration, instance and strategy are selected actively as those expected to bring the most benefit to the model for the expected cost (Steps 4 to 6). This process is repeated until we exhaust our budget or the desired performance is achieved.

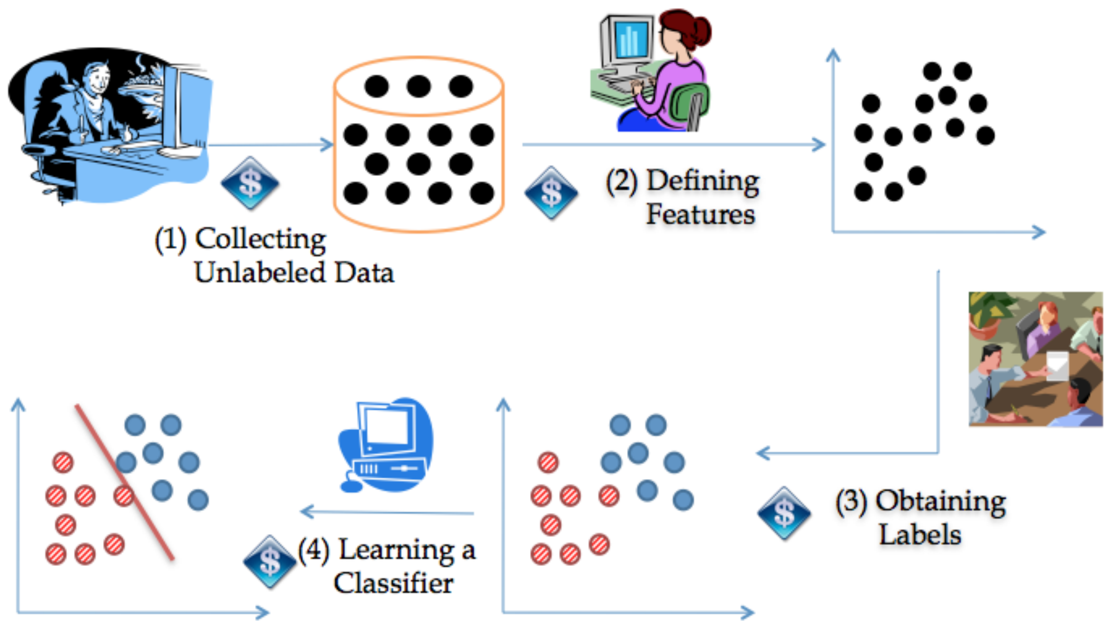


Figure 1.1: Main steps in supervised annotation learning.

Next, we introduce the proposed approach for automatic feature engineering, additional feedback through rationales, annotation cost estimation and cost-sensitive joint selection of instance and annotation strategy.

1.1 Generalized Automatic Feature Extraction

In supervised learning, an instance is represented by a feature vector. For a text instance, often words are used as features. This representation is called the *bag-of-words* representation, primarily because it ignores the word ordering. Bag-of-words representation is commonly used for text instances, since tokenization is most fundamental to Natural Language Processing (NLP). For most languages, segmenting text into words is straightforward. Even for languages where tokenization is harder (e.g. Chinese or Japanese), bag-of-words is still the simplest instance representation. It has also been found to provide good performance for many learning tasks. However, word features are limited in the information they can provide, and it is natural to believe that the relative ordering of words in a sentence is important for many learning tasks. N-grams preserve the ordering between words, and hence can provide more informative features. For example, bigram features, such as "bad movie", "good acting", etc., and trigram features such as "movie was good", etc. However, longer n-grams are sparse, and shorter n-grams cannot capture long distance dependencies, such as between "movie" and "bad" in the sentence, "the movie we saw yesterday was bad". Other annotations such as dependency parse trees can be used to capture such long distance dependencies between words. For example, in the sentence "the movie we saw yesterday was bad", 'movie' and 'bad' are in a 'nsubj' dependency relation. We refer to these annotations as *prior annotations* to distinguish them from the new annotations that we wish to learn, and since we assume that they are available *a priori* . We refer to the features derived from these prior linguistic annotations as *structured* features, since they capture the linguistic structure of text, and they have a structure of their own, described by nodes and edges (relations between the nodes). Another example of a

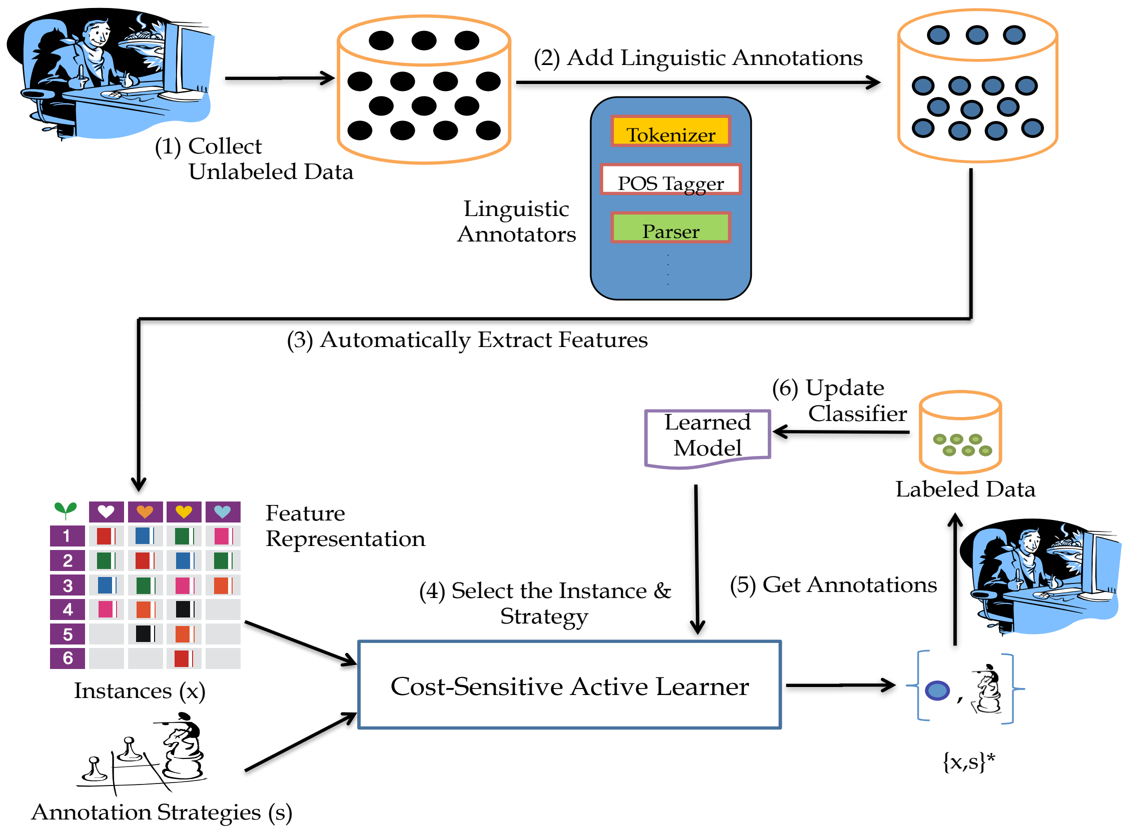


Figure 1.2: Interactive Annotation Learning (IAL) framework.

structured feature is the path between the verb and its candidate argument in a syntactic parse tree of a sentence, used for learning semantic roles (Pradhan et al., 2004). Similarly, in predicting whether a given relation exists between two entities, features derived from syntactic and semantic annotations have been found to be useful (GuoDong et al., 2005). Arora et al. (2009a) show that deep syntactic scope features constructed from transitive closure of dependency relations give a significant improvement over a bag-of-words model, when identifying types of claims in product reviews. Gamon (2004) found that deep linguistic features, derived from phrase structure trees and parts of speech annotations, give a significant improvement in performance over n-gram features, for the task of predicting satisfaction ratings in customer feedback data. Wilson et al. (2004) use syntactic clues derived from dependency parse trees as features for predicting the intensity of opinion phrases. Joshi and Rosé (2009) show that a combination of dependency relations and parts of speech annotations boosts the performance of a bag-of-words model for a sentiment classification task. Thus, features that combine several linguistic annotations have been found to improve performance of a bag-of-words model.

Often structured features, such as the ones described above, are defined by the domain experts based on their prior knowledge and data inspection, or through extensive experimentation. However, experts are often expensive to hire, and data inspection and experimentation are time consuming. This process does not generalize across domains or problems, and has to be repeated in an ad hoc way for each learning problem. We propose a general approach for automatically deriving structured features from prior annotations with minimal human supervision[1]. We propose an annotation graph representation for prior annotations and derive structured features automatically as frequent subgraphs from the annotation graphs. For a sentiment classification task and a protein-protein interaction extraction task, we show that these automatically extracted structured features improve performance of a bag-of-words model.

1.2 Indirect Feature Feedback through Rationales

Training a classifier involves learning a function of the features to approximate the target variable for the annotation we wish to learn. To learn a good approximation, several labeled instances are needed. Annotating an instance may require substantial user effort, called the *annotation cost*. In order to reduce the total annotation cost, instead of labeling several instances, the annotator could provide information about the features directly. While the annotator may not be able to provide the exact weights for the features, they may be able to identify the relevant features and indicate their class association. Feedback on features, in addition to annotating the instances, has been shown to achieve the same performance with fewer examples (Godbole et al., 2004; Raghavan and Allan, 2007; Druck et al., 2009; Melville and Sindhwani, 2009). However, most of the work so far on feature feedback has focussed on direct feedback on simple features, such as words. It is not clear how direct feature feedback can be extended to structured features that are often hard to visualize. Direct feedback is often solicited without the context of an instance. For some features, it may be difficult to determine their relevance and class association without any context. Instead, we propose an alternate annotation strategy, where in addition to providing an instance's label, the annotator indicates parts of the instance that are *rationales* for the indicated label. Rationales provide indirect feedback on features, since features that overlap with rationales should be important for the classification task, and this indication is only indirect. Zaidan et al. (2007) define *rationales* as spans of text in a document that are key indicators of its class label. In this work, we segment an instance into sub-instances, and the annotator identifies the sub-instances that are rationales. For example, a document may be segmented into sentences and the annotator identifies the sentences that are rationales for the document's label. To annotate a span of text as a rationale, the annotator must perform the cognitive task of identifying the appropriate boundaries for the rationale, which may require more time than voting on a pre-segmented span of text. Also, highlighting a span should require more user interface time than voting on a text segment. Additionally, exact spans for the rationales may vary across users, while we can expect to see more agreement between the users when they vote on sub-instances. Rationales, as sub-instances in an instance, are not limited to a text instance. An image can be segmented into regions (Shi and Malik, 2000; Estrada et al., 2004) and a video can be segmented into scenes (Grundmann et al., 2010), and the annotator identifies the segments of an image or a video that are rationales for its label. While direct feature feedback has been modeled as a separate task from labeling instances in most prior work (Godbole et al., 2004; Raghavan and Allan, 2007; Druck et al., 2009; Melville and Sindhwani, 2009), rationales are solicited together with the instance's label. The annotator perhaps already makes this distinction to determine the instance's label. We ask them to provide their reason for assigning the indicated label to an instance.

In this thesis, we consider two annotation strategies where the user provides: 1) instance's $\underline{l}$abel (target annotation) $\underline{o}$nly (*LO*), and 2) instance's $\underline{l}$abel, together with $\underline{r}$ationales in support of the instance's label (*LR*). Rationales provide additional information per instance. However, it may take an annotator some extra time to annotate rationales in addition to providing an instance's

[1]The expert still needs to decide what prior annotations to use.

label. Since the annotators are often paid by the hour, we measure the annotation cost in terms of the annotation time. In prior work (Zaidan et al., 2007; Arora and Nyberg, 2009; Abedin et al., 2011; Donahue and Grauman, 2011), the additional cost for annotating rationales is not accounted for, and the two annotation strategies are compared in terms of the performance for a given number of instances. The additional cost for rationales may vary across users, instances, annotation tasks, user interface design, etc. We compare the two annotation strategies of LO and LR for different additional costs for annotating rationales. For a sentiment classification task and an aviation incident cause identification task, we show that rationales can provide better performance for a given annotation cost, when annotating them incurs a small extra cost.

The amount of benefit from feature feedback may vary across learning problems. With enough labeled data, we may not benefit from feature feedback. The benefit from feature feedback may also vary with the features used to represent the instances. If the feature space is large, we may need several labeled instances to identify the relevant features, while relatively fewer labeled features may help us quickly find these relevant features and achieve the desired performance. Apart from the feature space size, it also matters what types of features are used. When hand-crafted features from a domain expert are used (Pradhan et al., 2004), we expect to gain less from feature feedback, since most of the features will be relevant. On the other hand, when features are extracted automatically as patterns in the annotation graphs, as described above, feature feedback can help to identify the relevant features in the large feature space. The benefit from feature feedback will also depend on the strategy for selecting instances for annotation. In the case of indirect feature feedback through rationales, instances selected will also determine what features receive feedback. Hence, the instance selection strategy should affect the amount of benefit from feature feedback. In text classification, an instance often contains a large amount of text, and even a simple bag-of-words representation will generate a lot of features. Often only a part of the text is relevant for the classification task. For example, in movie reviews, often the reviewers talk about the plot and characters in addition to providing their opinion about the movie. Often this extra information is not relevant to the classification task and bloats the feature space without adding many useful features. With feature feedback, we hope to filter out some of this noise and improve the model. Thus, the amount of irrelevant information in the instance should also play an important role in determining the benefit from feature feedback. We expect to see less of such noise when an instance is more concise. For example, a movie review snippet (about a sentence length) tends to have less irrelevant text than a full movie review (several sentences). In addition to analyzing document instances with varying amount of noise, we also compare the benefit from feature feedback for problems with different instance granularity. The amount of benefit from feature feedback will also depend on how feedback is solicited from the annotator, and how it is incorporated back into the model. Independent of these factors, we estimate the maximum benefit from feature feedback, and show how it varies across learning problems. For two sentiment classification tasks with different instance granularity and a webpage category classification task, we show what characteristics of a learning problem have a significant effect on the amount of benefit from feature feedback. We find a significant correlation between measures proposed in the literature for quantifying the complexity of the learning problems (Raghavan et al., 2007) and the maximum benefit from feature feedback. We also suggest other measures for categorizing learning problems to understand the observed differences in the amount of benefit from feature feedback across learning problems.

1.3 Annotation Cost Estimation

Annotation cost is the amount of time spent in providing the desired supervision for learning the target concept. The annotation cost is often measured in terms of the number of instances, assuming a uniform cost per instance. However, this assumption does not always hold, and annotation times may vary across instances (Donmez and Carbonell, 2008b; Ringger et al., 2008; Settles et al., 2008a; Arora et al., 2009b; Baldridge and Palmer, 2009). Longer sentences/documents require more time to read and annotate than shorter ones. Ambiguous instances are more difficult to annotate than easier ones. Thus, instances may have different annotation costs. Annotation cost also varies across annotators, and multiple annotators are often employed in annotation tasks. Non-native speakers of a language may require more time than native speakers to understand and annotate a text instance in that language. Also, the annotators may have different expertise at the annotation task and familiarity with reading text online. Annotation cost will also depend on the annotation strategy, i.e., the tasks performed in annotating an example. If the annotator provides feedback on features in addition to the instance labels, more work is done per instance and the annotation cost would be more. Annotation cost may be known *a priori* for some tasks, for example, medical tests for diagnosis have a predetermined cost. However, for many annotation tasks, the annotation cost is not known *a priori*, as it includes the time required to read and understand the text, and the time to make the annotation decisions, which may vary with the instance, annotation strategy, and the annotator. If the annotation cost can be reliably estimated, the estimate can be used in active selection to directly minimize the annotation cost for the desired performance. An annotation cost estimate can also be used for estimating the annotation budget and for comparing several selective sampling strategies before selecting one to use for a real annotation task. In this thesis, we propose a supervised regression model for estimating the annotation cost for an instance, strategy and annotator in a multi-annotator environment with indirect feature feedback through rationales. For data collected from multiple annotators for a sentiment classification task, we show that the proposed approach that uses the characteristics of an instance, strategy and annotator, outperforms simpler estimates based on a single characteristic.

1.4 Cost-Sensitive Active Selection of Instance and Strategy

Traditionally, in supervised annotation learning, unlabeled instances are labeled in the given order, or are randomly selected for annotation, until the desired performance is reached. This approach to supervised learning is often referred to as *passive learning*. However, instances may provide different incremental value to the learning algorithm. In order to minimize the total annotation cost for the desired performance, *active learning* selectively samples the instances to label in each iteration, as those expected to bring the most benefit to the model. The annotator first labels a small subset of the unlabeled data, and a supervised model is learned on this initial set. The learned model is applied to the pool of unlabeled instances. From this pool, instances expected to improve the model's performance the most, are presented to the annotator. This procedure continues until all instances have been labeled, the desired performance is achieved, or we exhaust our annotation budget.

In this thesis, we consider two annotation strategies of instance's label only (LO) and instance's label together with rationales (LR). We may not benefit equally from rationales for all instances. Each instance (with or without rationales) may provide different incremental value to the learning algorithm. The annotation cost may also vary across instances and annotation strategies. We propose a cost-sensitive active learning approach, where in each iteration, an instance is actively selected for the given strategy. For a sentiment classification task and an aviation incident cause identification task, we show that this strategy sensitive active instance selection outperforms random or strategy independent active instance selection. When rationales are very expensive to annotate, they may not be beneficial for all instances for the additional cost. We further extend the proposed approach to selectively ask for rationales by jointly selecting an instance and strategy in each iteration. While the best fixed strategy among *LO* and *LR* varies with the additional cost for annotating rationales, we show that the proposed approach performs as well as or better than the best fixed strategy at different additional costs for annotating rationales.

1.5 Thesis Contributions

We now summarize the main contributions of this thesis. In this thesis, our goal is to reduce the total cost for the desired performance in supervised annotation learning. Specifically we focus on reducing the cost of feature engineering and the total annotation cost. The main contributions of the thesis towards this goal, are the following.

- **Generalized Automatic Feature Extraction:** We propose an annotation graph representation for instances based on words and other linguistic annotations. Structured features are extracted automatically from the annotation graphs for instances using a subgraph mining algorithm. The proposed approach provides a generic framework for finding structured features automatically. For a sentiment classification task and a protein-protein interaction extraction task, we show that these automatically extracted structured features provide a significant improvement in performance over the bag-of-words features.

- **Active Annotation with Indirect Feature Feedback:** In order to reduce the total annotation cost for the desired performance, we propose an alternate annotation strategy, where in addition to the instance's label, the annotator provides rationales in support of the instance's label. We segment an instance into sub-instances, and the annotator identifies the sub-instances that are rationales. The additional cost for rationales may vary across the annotators, instances, annotation tasks, user interface design, etc. For a sentiment classification task and an aviation incident cause identification task, we show that rationales provide a significant improvement in performance for a given annotation cost, when the additional cost for annotating them is small. Each instance (with or without rationales) may provide different incremental value to the learning algorithm. We propose a cost-sensitive active learning framework, where in each iteration, an instance and strategy are jointly selected to minimize the total annotation cost for the desired performance. We show that the proposed approach outperforms cost-sensitive or cost-insensitive instance selection for a fixed strategy, and cost-insensitive joint selection of instance and strategy, for different additional costs for annotating rationales.

- **Annotation Cost Estimation:** We propose a supervised regression model for estimating the annotation cost for a given instance, strategy and annotator, in a multi-annotator environment with indirect feedback on features through rationales. We use the characteristics of the instance, annotation strategy and annotator to estimate the annotation cost. For data collected from multiple annotators for a sentiment classification task, we show that this estimate is more accurate than simpler estimates based on a single characteristic.

- **Assessment of the Benefit from Feature Feedback:** For two sentiment classification tasks with different instance granularity and a webpage category classification task, we analyze how the maximum benefit from feature feedback varies across learning problems, and what characteristics of a learning problem have a significant effect on the amount of benefit from feature feedback. We also show that measures for quantifying the complexity of the learning problems (Raghavan et al., 2007) have a significant correlation with the maximum benefit from feature feedback. We also suggest other measures for categorizing learning problems and study their relationship with the amount of benefit from feature feedback.

1.6 Thesis Outline

In this chapter, we presented an introduction to the main contributions of this research. The remaining thesis is organized as follows:

- *Chapter 2: Generalized Automatic Feature Extraction* describes the proposed annotation graph representation and the subgraph feature mining approach for automatically extracting structured features from annotation graphs.
- *Chapter 3: Active Annotation with Indirect Feature Feedback* presents a comparison of the two annotation strategies of *LO* and *LR* at different additional costs for annotating rationales, and for several instance selection strategies. It also presents the proposed approach for cost-sensitive joint selection of instance and strategy in each iteration, and its comparison with instance selection for a fixed strategy and other baselines.
- *Chapter 4: Annotation Cost Estimation* describes the supervised model for estimating the annotation cost in a multi-annotator environment with indirect feature feedback through rationales.
- *Chapter 5: Assessment of Benefit from Feature Feedback* presents an analysis of how maximum benefit from feature feedback varies across learning problems, and what characteristics of a learning problem have a significant effect on the amount of benefit from feature feedback.
- Finally, *Chapter 6: Conclusions and Future Work* presents the conclusions and directions for the future work.

Chapter 2

Generalized Automatic Feature Extraction

An instance in supervised learning is represented by a feature vector. For example, a text instance can be simply represented as a bag of words. However, this representation ignores the linguistic relations between words, which should benefit most learning tasks. Features that capture the linguistic structure in text by combining several linguistic annotations, such as parts of speech, dependency relations, etc., have been shown to improve performance beyond bag-of-words features for several learning tasks, such as semantic role labeling (Gildea and Jurafsky, 2000; Pradhan et al., 2004), relation extraction (GuoDong et al., 2005), sentiment classification (Wilson et al., 2004; Arora et al., 2009a; Gamon, 2004; Joshi and Rosé, 2009), etc.

Natural Language Processing (NLP) field has advanced to a stage that there are several linguistic annotators available with acceptable accuracy. For example, tokenizers, parts of speech taggers, dependency parsers, phrase structure tree parsers, semantic role labelers, named entity taggers, etc. Stanford NLP group provides several of these annotators[1]. CMU provides a link grammar parser[2] and a frame-semantic parser[3]. UIUC also provides a few NLP tools[4]. ASSERT[5] is a semantic role labeler freely available. OpenCalais[6] is a web service that provides several semantic annotations, such as named entities, events, facts, etc. There are also other tools available, such as WordNet[7], a large lexical database of nouns, verbs, adjectives and adverbs, grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept.

Often the text is processed through several linguistic annotators, and structured features that capture the linguistic patterns, are often handcrafted by the domain experts after careful examination of the data. For example, as a preprocessing step in *Reconcile* (Stoyanov et al., 2009), a research platform for coreference resolution[8], all documents are passed through a series of linguistic processors, such as tokenizers, parts of speech taggers, syntactic parsers, etc. Features are constructed from these prior annotations as properties that characterize a pair of noun phrases. For example, a feature might denote whether two noun phrases agree in number, or whether two

[1]`http://nlp.stanford.edu/software/`

[2]`http://www.link.cs.cmu.edu/link/index.html`

[3]`http://www.ark.cs.cmu.edu/SEMAFOR/`

[4]`http://cogcomp.cs.illinois.edu/page/software`

[5]`http://cemantix.org/assert.html`

[6]`http://www.opencalais.com/about`

[7]`http://wordnet.princeton.edu/`

[8]Coreference resolution is the task of identifying noun phrases in text that refer to the same entity.

noun phrases share a word in common. The features used in Reconcile are fixed and defined by the domain experts. However, the cost of such feature engineering is often ignored when accounting for the cost of developing new annotators.

Hand-crafted features often do not generalize well across datasets and domains. Pradhan et al. (2008), in their follow up experiments with Semantic Role Labeling, found that some features from their list of handcrafted features (Pradhan et al., 2004) were specifically suited for one argument class and tended to hurt performance for other arguments. They also found that different features are useful for argument identification and argument classification[9]. Semantic features are dominant in the argument classification task, and syntactic features are dominant in the argument identification task. Furthermore, they found that feature usefulness varies with the genre. Features designed for WSJ corpus gave similar performance on Brown corpus for the argument identification task. However, for the argument classification task, the features designed for WSJ corpus gave considerably lower performance for Brown corpus. Predicates and head words (or, words in general) were found to be more useful features for a homogeneous corpus like WSJ, as opposed to a heterogeneous collection like Brown corpus. Thus, hand-crafted features may not generalize across domains/genres, and the expensive process of feature engineering has to be repeated for each new task and domain.

The goal in this thesis is to reduce the cost of feature engineering by automatically extracting features from prior linguistic annotations in text. We propose a generic *annotation graph* representation (Arora and Nyberg, 2009), a formalism for representing several linguistic annotations together on text. Instances are represented as annotation graphs, from which frequent subgraph patterns are extracted and used as features for learning new annotations. We use an efficient frequent subgraph mining algorithm (gSpan) (Yan and Han, 2002a) to extract frequent subgraphs from the annotation graphs. The annotation graph and subgraph mining algorithm provide us a quick way to test several alternative linguistic representations of text. For a sentiment classification task and a protein-protein interaction extraction task, we show that structured features automatically extracted using the proposed approach provide a significant improvement in performance over a bag-of-words model.

The work presented in this chapter is an exposition of the work presented in (Arora et al., 2010), with some additional ideas, experiments and results. Figure 2.1 presents the details of the automatic feature extraction in interactive annotation learning framework proposed in this thesis. The first step is to decide what prior annotations are suitable for the learning task, and how nodes and edges in the graph should be created from these annotations. The second step is to automatically extract features as frequent subgraphs in these annotation graphs. The third step is to create a feature representation for the instances, based on the selected subset of subgraph features.

For the remaining chapter, we first present a formal definition of the annotation graph representation, how it is constructed for an instance in different learning problems, and a few motivating examples for subgraph features. We then present the frequent subgraph mining approach used to extract subgraph features, followed by feature selection techniques we experiment with. After this, we present our experiments and results, followed by the related work, conclusions and suggestions for the future work.

[9]Argument identification is the task of identifying what constituents in a parse tree are arguments of a given predicate. Argument classification is the task of classifying a given argument of a predicate into one of the argument type classes.

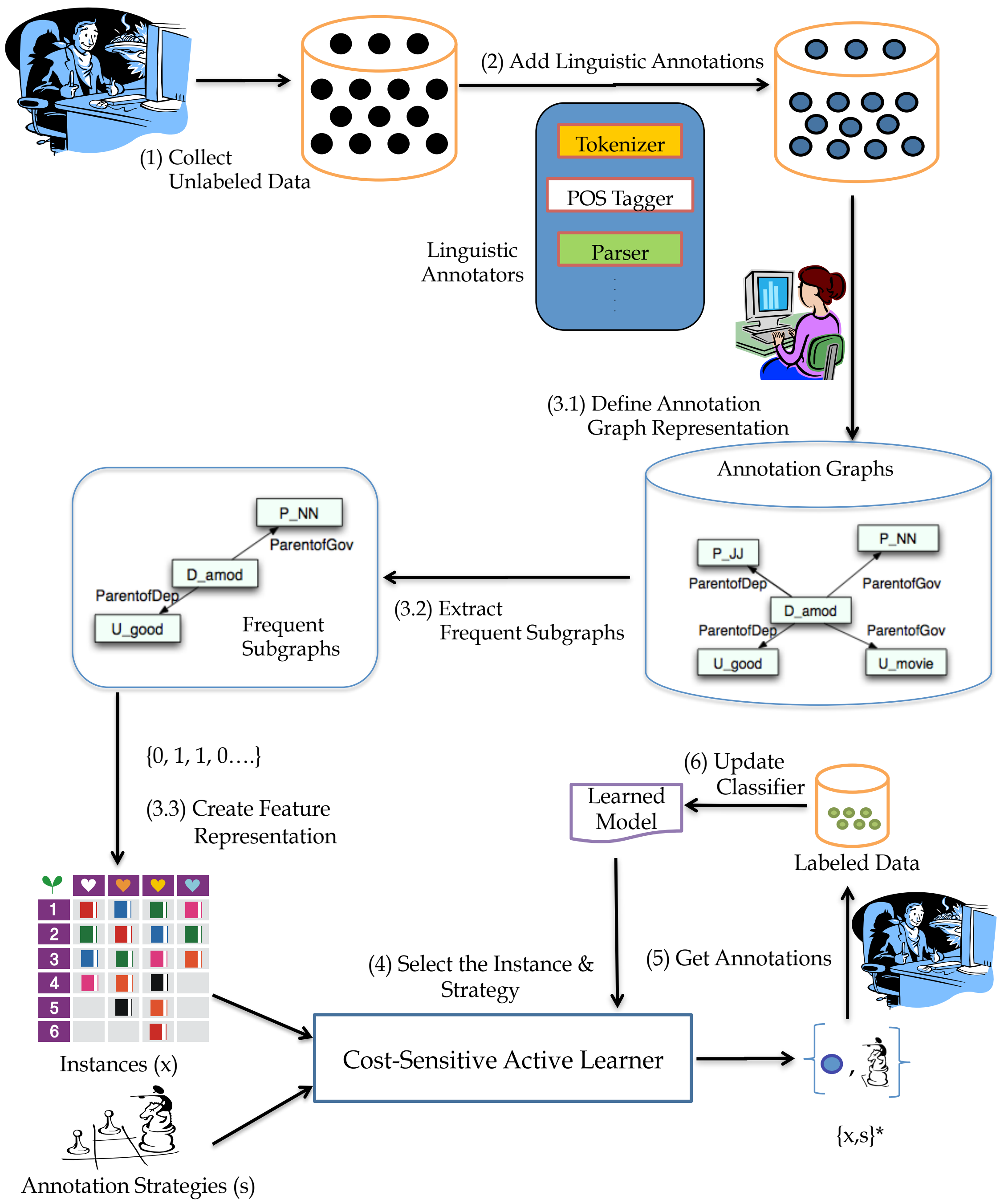

(1) Collect Unlabeled Data
(2) Add Linguistic Annotations
Tokenizer
POS Tagger
Parser
Linguistic Annotators
(3.1) Define Annotation Graph Representation
Annotation Graphs
P_JJ
P_NN
ParentofDep
ParentofGov
D_amod
U_good
U_movie
(3.2) Extract Frequent Subgraphs
Frequent Subgraphs
{0, 1, 1, 0....}
(3.3) Create Feature Representation
Instances (x)
Annotation Strategies (s)
(4) Select the Instance & Strategy
Cost-Sensitive Active Learner
Learned Model
(5) Get Annotations
(6) Update Classifier
Labeled Data
{x,s}*

Figure 2.1: Automatic feature extraction in interactive annotation learning framework.

2.1 Annotation Graph Representation

We define the annotation graph as a quadruple: $G = (N, E, \Sigma, \lambda)$, where N is the set of nodes, E is the set of edges s.t. $E \subset N \times N$, and $\Sigma = \Sigma_N \cup \Sigma_E$ is the set of labels for nodes and edges, and $\lambda : N \cup E \rightarrow \Sigma$ is the labeling function (Arora and Nyberg, 2009). Examples of node labels (Σ_N) are *tokens (unigrams)* and annotations such as *parts of speech*, *polarity*, etc. Examples of edge labels

(Σ_E) are *leftOf*, *dependency type*, etc. The *leftOf* relation is defined between two adjacent nodes, for example, between two adjacent words in a sentence. The *dependency type* relation is defined between a head word and its modifier, for example, an 'amod' relation between 'movie' (head) and 'good' (modifier) in the phrase "good movie".

Annotations may be represented in an annotation graph in several ways. For example, a dependency triple annotation 'good_amod_movie', may be represented as an edge with label *d_amod* between the nodes for the head or governor word 'movie' and its modifier or dependent word 'good', or as a node *d_amod* with edges *ParentOfGov* and *ParentOfDep* to the nodes for the head and modifier words, respectively. An example of an annotation graph is shown in Figure 2.2. It describes a movie review comment, *'interesting, but not compelling'*. The words 'interesting' and 'compelling' both have positive prior polarity, however, the phrase expresses a negative sentiment towards the movie. Heuristics for special handling of *negation* have been proposed in the literature. For example, Pang et al. (2002) append every word following a negation, until a punctuation, with a 'NOT'. Applying a similar technique to our example gives us two sentiment bearing features, one positive (*'interesting'*) and one negative (*'NOT-compelling'*), making the instance ambiguous.

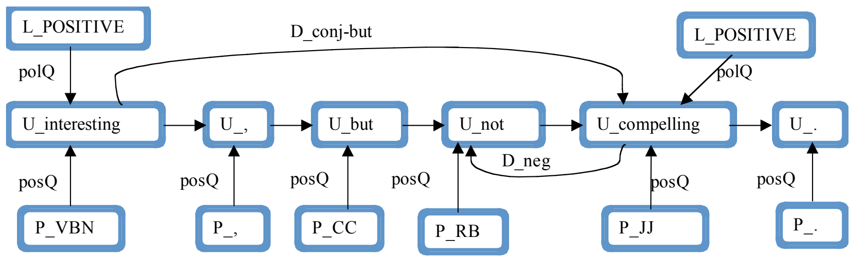
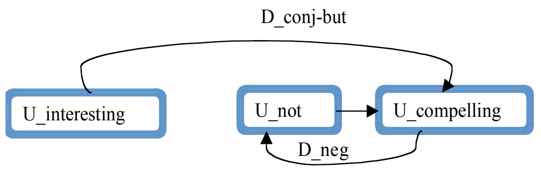


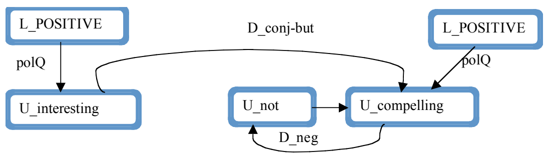
Figure 2.2: Annotation graph for sentence *'interesting, but not compelling.'* . Prefixes: 'U' for unigrams (tokens), 'L' for polarity, 'D' for dependency relation and 'P' for parts of speech (POS). Edges with no label encode the 'leftOf' relation between words. 'posQ' is short for 'POSQualifies' relation between a word and its POS.

Figure 2.3 shows three discriminating sub-graph features for this example phrase, derived from the annotation graph in Figure 2.2. These subgraph features capture the negative sentiment in the phrase. The first feature in Figure 2.3a captures the pattern using dependency relations between words. A different review comment may use the same linguistic construction but with different words, for example, *'good, but not excellent'* This is the same linguistic pattern but with different words, which the model may not have seen before, and hence may not classify this instance correctly. This suggests that the feature in Figure 2.3a is too specific.

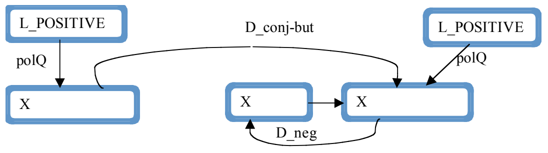
In order to mine general features that capture the linguistic structure in text, we may add prior polarity annotations to the annotation graph, using a polarity lexicon (Wilson et al., 2005). Figure 2.3b shows the subgraph in Figure 2.3a with polarity annotations. If we want to generalize the pattern in Figure 2.3a to any positive words, we may use the feature subgraph in Figure 2.3c with a wild card (X) on words that are polar. This feature subgraph captures the negative sentiment in both phrases *'interesting, but not compelling'* and *'good, but not excellent'*. Similar generalization using wild cards on words may be applied with other annotations, such as parts of speech. By choosing where to put the wild card, we can get features similar to, but more powerful than, the dependency back-off features in (Joshi and Rosé, 2009).



(a)



(b)



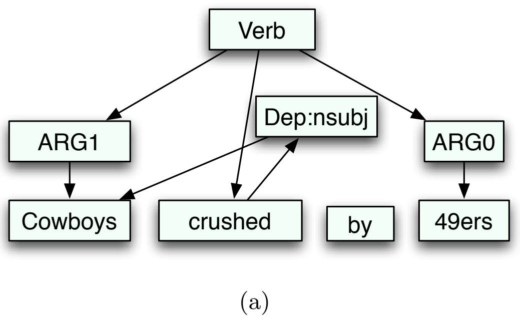
(c)

Figure 2.3: Subgraph features from the annotation graph in Figure 2.2.

Dolphins win wild one over Chiefs
Ravens defeat Giants
Dolphins outgun patriots
Ravens defeat Giants
Giants defeat Vikings
Colts dump Jaguars
Colts improve perfect NFL start to 11-0 by stomping Steelers
Colts stretch perfect NFL start as doom tolls for ”Big Ben”
Steelers edge Ravens
Steelers edge Chargers
Broncos ride roughshod over Chiefs
Steelers edge Jets and Falcons rout Rams
Steelers edge Jets
Eagles rout Dallas
Titans dump Miami
Jets snap Titans’ six-game NFL win streak
Bledsoe leads Patriots past Steelers
Packers end 49ers’ NFL reign
Giants outlast Cardinals in cellar fight
Cowboys crushed by 49ers
San Diego ousts Miami
49ers rip Bears

Table 2.1: Example of NFL news headlines. The task here is to identify who won the game.

As another motivating example for annotation graph representation and automatically extracting structured features from the annotation graphs, consider the task of identifying the winning team in a game from NFL sports news headlines. Table 2.1 shows a few examples of headings from a sports news corpus[10]. The classification task here is to classify for each team mentioned in the heading, whether that team won the game. A simple bag-of-words approach will fail for this task on half of the data, as both the teams will have the same bag-of-words features. Features that capture the immediate context, such as the words to the left or right of the team name will be helpful in identifying the winner for many instances. To capture the longer context, dependency relations can be used. For example, the winner is often the subject of the main verb as can be seen from examples in Table 2.1. However, such features will not be able to differentiate between "Cowboys crushed by 49ers" and "Cowboys crushed 49ers". 'Cowboys' is the winner in the second instance, but not in the first one. However, in both instance, 'Cowboys' has the same context, and is also the subject ('nsubj' dependency relation) of the verb 'crushed'. The difference between the two instances is that first one uses a passive voice, and the second one uses an active voice. Semantic roles (Gildea and Jurafsky, 2000) can distinguish between active and passive voice, and can correctly identify the agent (or the argument 'ARG0') in both cases. Figure 2.4 shows the semantic roles (and dependency relations) for the two sentences in our example. As can be seen, the team that is the 'ARG0' of the main verb, is the winner. If we observe carefully, for all instances in Table 2.1, the winner is the 'ARG0' or agent of the main verb of the sentence. In other words, when reporting a winner of a game, they are often referred to as the agent or the causer of something. As we can see from the examples in Table 2.1, many different verbs are used to report the winner of the game. Many of these verbs are similar in their meaning, and can be generalized to a common class of verbs, say the 'winner-verbs' in this case. The winner could then be described as the 'ARG0' of a 'winner-verb'.



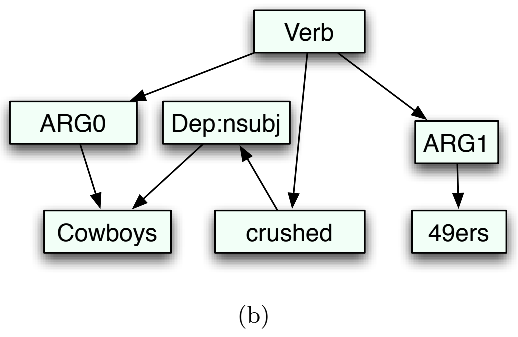


Figure 2.4: Dependency relations and Semantic Role Annotations for NFL winner examples in Table 2.1.

In this simple example, we went through three levels of prior annotations, in increasing order of complexity, in order to determine the right set of features to use. When building a classifier, much more work is required. With an automatic approach for feature extraction from linguistic annotation graphs, our goal is to reduce this effort for feature engineering in building new annotators.

2.1.1 Instance Representation

An instance in the proposed approach is represented by an annotation graph. For a sentence classification task, the representation is straightforward. A sentence is a common unit of input to a linguistic annotator, such as parser. Each sentence is run through a linguistic annotator, and the

[10]Simplified headings from a NFL News Corpus from DARPA's Machine Reading program.

output annotations are used to create an annotation graph, by mapping the annotations to nodes and edges in the graph. If the classification unit is a document, like in a document categorization task, then either a single annotation graph can be constructed for the whole document, or a document can be considered as a collection of graphs. However, an instance might also be a word or a phrase in a sentence, like in semantic role labeling. An instance might also be a pair of phrases/words, like in relation detection or coreference resolution. In such cases, a simple approach is to use the annotation graph for the full sentence. Another approach is to define the annotation graph from only the local context of the phrase(s) to be classified. There are many heuristics for defining the annotation graph from the local context. One approach is to only use annotations that include the word, phrase or pair of phrases to be classified. An alternative approach when classifying a pair of phrases is to only consider the shortest path that connects the two phrases in the annotation graph. For example, consider the protein-protein interaction classification task (Erkan et al., 2007), where the goal is to identify mentions of protein pair interactions in the biomedical literature. A given sentence may contain several protein mentions, and the task is to detect for each pair of protein mentions, whether the sentence suggests that they interact. In this case, if we consider the complete annotation graph for the sentence as is, then for each pair of proteins we will have the same annotation graph, same features and hence the same classification. A simple approach to differentiate between multiple instances in a sentence, is to create a copy of the sentence for each instance, and mark the protein pair to be classified in each, by giving them a special name or adding an annotation on them. For example, in (Erkan et al., 2007), a sentence with n proteins is replicated for each pair of proteins, generating $\binom{n}{2}$ instances per sentence. For example, the sentence,"The results demonstrated that KaiC interacts rhythmically with KaiA, KaiB, and SasA", has 3 proteins and 6 ($\binom{4}{2}$) instances, 3 of which are the following: 1) The results demonstrated that PROTX1 interacts rhythmically with PROTX2, PROTX0, and PROTX0; 2) The results demonstrated that PROTX1 interacts rhythmically with PROTX0, PROTX2, and PROTX0; and 3) The results demonstrated that PROTX0 interacts rhythmically with PROTX0, PROTX1, and PROTX2. Here, PROTX1 and PROTX2 represent the protein pair under consideration, and PROTX0 is used for all other proteins. These labels (e.g. PROTX1) are used to reduce the data sparseness due to specific protein names, as in the data shared by Erkan et al. (2007). With these labels for proteins, the annotation graph will be different for each instance. Figure 2.5 shows an annotation graph with the protein pair to be classified marked with the labels PROTX1 and PROTX2. The annotation graph here was constructed from the dependency relation annotations from the Stanford Parser (Klein and Manning, 2003). A dependency relation such as 'interaction_prep-of_PROTX2', has a governor node ('interaction'), a relation node ('prep-of') and a dependent node ('PROTX2'). In the annotation graph, we add a directed edge (*GovToRel*) from the governor node to the dependency relation node, and a directed edge (*RelToDep*) from the relation node to the dependent node.

The entire sentence and its annotation graph may not be required for classifying a pair of proteins. It has also been widely acknowledged in the literature that the local context of a protein pair is likely to carry the important information regarding their relationship. For example, Bunescu and Mooney (2005a) consider only the sequence of words before, between and after the protein pair in a sentence, with a restricted total number of words in the context. Erkan et al. (2007), Bunescu and Mooney (2005b) and Fayruzov et al. (2009) only consider the shortest path between the protein pair in a dependency parse tree. However in some cases, the information about interaction of two proteins might be outside the shortest path. Figure 2.6 shows two instances in a sentence where the shortest path between the two proteins under consideration, is the same, while one is a positive instance and the other is negative. Airola et al. (2008) thus consider all paths between protein

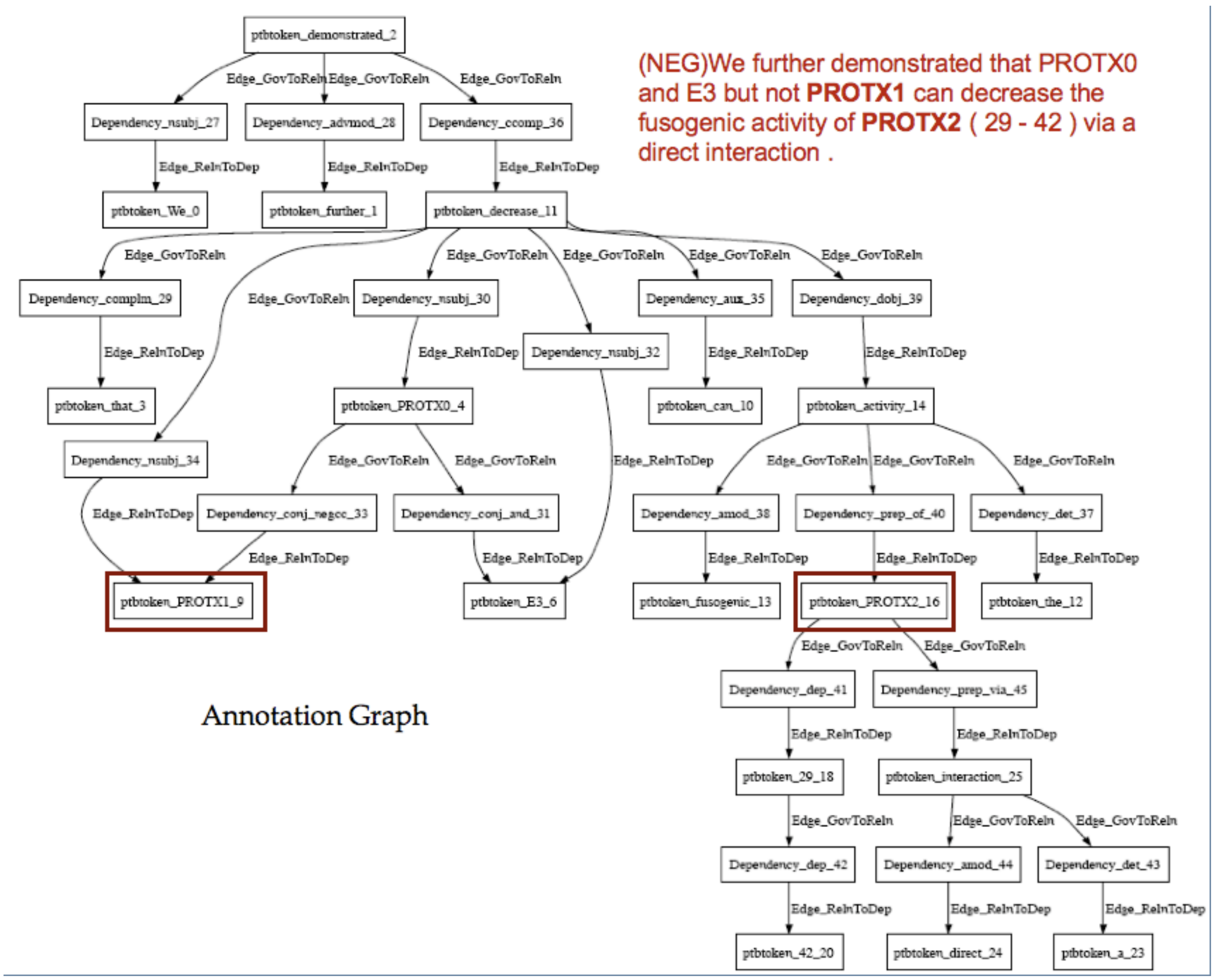


Figure 2.5: Annotation graph for the sentence "We further demonstrated that PROTX0 and E3 but not PROTX1 can decrease the fusogenic activity of PROTX2 (29 - 42) via a direct interaction .". The task is to judge if the sentence suggests interaction between PROTX1 and PROTX2. For each dependency triple annotation, a directed edge (*GovToRel*) is added from the governor node to the dependency relation node, and another directed edge (*RelToDep*) is added from the relation node to the dependent node.

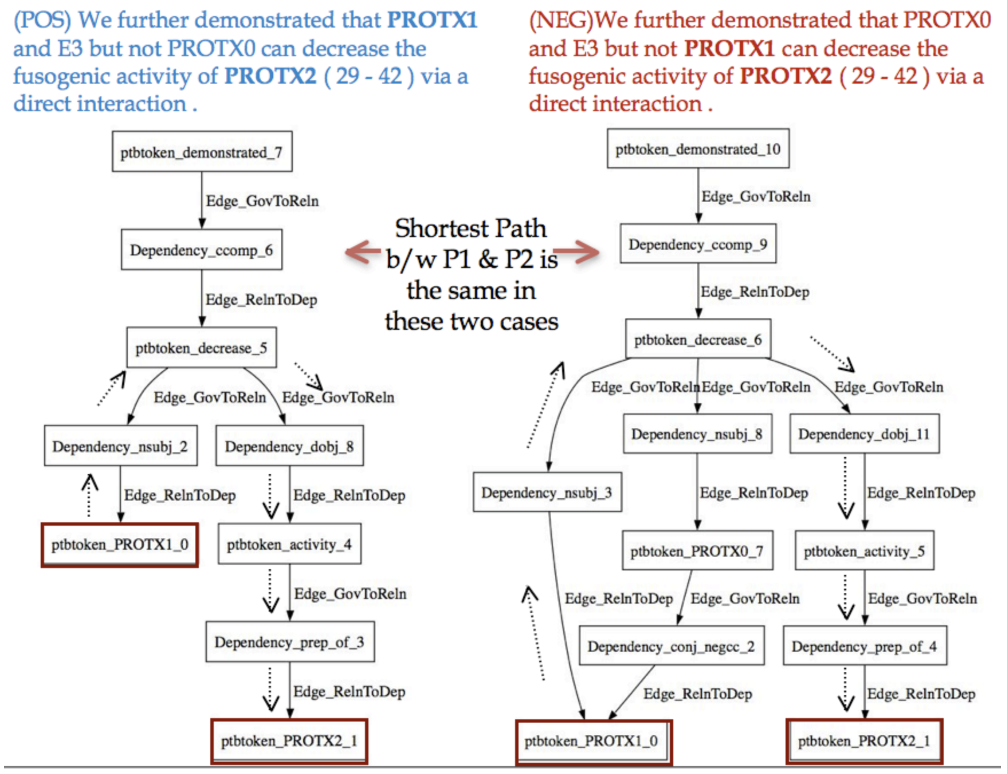


Figure 2.6: Pruned annotation graphs for sentences “We further demonstrated that PROTX1 and E3 but not PROTX0 can decrease the fusogenic activity of PROTX2 (29 - 42) via a direct interaction .” and “We further demonstrated that PROTX0 and E3 but not PROTX1 can decrease the fusogenic activity of PROTX2 (29 - 42) via a direct interaction .”. The shortest path from PROTX1 to PROTX2 is marked with dotted arrows.

pairs in a dependency graph, giving more weight to the shortest path.

In most of the work discussed above, words/tokens or dependency parse trees/graphs are used as prior annotations. With a few annotations, considering sequence and/or paths in a tree or a graph might be sufficient. However, with the annotation graph representation, where we may have several annotations, important information may not be captured by paths alone, and we may need to consider additional branches from the path. Also, all the information on a path might not be important, and we may want to consider only parts of the path, which can be represented as a disconnected graph. Thus, we suggest a pruned annotation graph representation for each protein pair in a sentence, where only nodes and edges thought to be relevant for classifying the protein pair are used. We now describe an approach to construct such a pruned annotation graph by selecting a subset of the nodes and edges from the annotation graph based on the local context of the protein pair. Assuming directed edges in the annotation graph, a node and the corresponding edge is added to the pruned graph only if it has a path to reach one of the nodes in the protein pair to be classified (referred to as the target nodes). The hypothesis is that a node has an influence on a target node, only if it has a way to reach it in the graph. Algorithm 1 describes this procedure for constructing a pruned annotation graph, given a pair of target nodes. In the protein-protein interaction task, the target nodes are the protein pair to be classified, marked as PROTX1 and

PROTX2, as mentioned before.

Algorithm 1 Algorithm for deriving pruned annotation graph. Edges are directed, from the parent to the child node.

```
Mark the target nodes and add all their edges to an inspection list
while there are edges in the inspection list, and there are new nodes marked do
  for each edge in the inspection list do
    if child is marked then
      if parent is marked then
        Add the edge to the pruned graph, and remove it from the inspection list
      else
        Mark the parent
        Add the parent node and the edge to the pruned graph, and remove it from the inspec-
        tion list
        Add all other edges of the parent node to the inspection list
      end if
    else if child is a terminal child, i.e. it has no children, and hence it would never be marked
    then
      Remove the edge from the inspection list
    end if
  end for
end while
```

Figure 2.7 shows pruned annotation graphs for the three proteins pairs in the sentence, “The study supports the interaction of PROTX2 with PROTX0 and PROTX1.” Figure 2.6 shows the pruned annotation graph for a positive and a negative instance in the sentence “We further demonstrated that PROTX1 and E3 but not PROTX0 can decrease the fusogenic activity of PROTX2 (29 - 42) via a direct interaction .”. Note that the pruned annotation graph captures more information than the shortest path, which is same for the two instances in this sentence. The complete annotation graph for this sentence is shown in Figure 2.5.

Thus, an instance may be represented by the complete annotation graph or a pruned annotation graph. There can be many heuristics for creating the pruned annotation graph. We presented one such algorithm above. Note that the shortest path can be considered as a pruned annotation graph, with the heuristic of selecting only the nodes and edges on the shortest path between the two nodes in a graph. After we have constructed the annotation graphs (pruned or complete) for the instances, we derive features as subgraphs from these annotation graphs. In the next subsection, we describe the subgraph mining and matching algorithms we use in this work.

2.2 Subgraph Mining and Subgraph Matching Algorithms

In the previous section, we demonstrated that the subgraphs from the annotation graphs can be used to capture the linguistic patterns useful for learning the target annotations. Subgraph patterns that are frequent will generalize better to new instances. Hence, we use a frequent subgraph mining algorithm to find frequent subgraph patterns in annotation graphs for the training instances, which we use as features in a supervised learning algorithm. At test time, we need to know what subgraph features are present in a test instance. This requires searching for these subgraphs in the annotation graphs for the test instances. We call this the *subgraph matching* procedure. We now describe the algorithms and packages we use for frequent subgraph mining and subgraph matching.

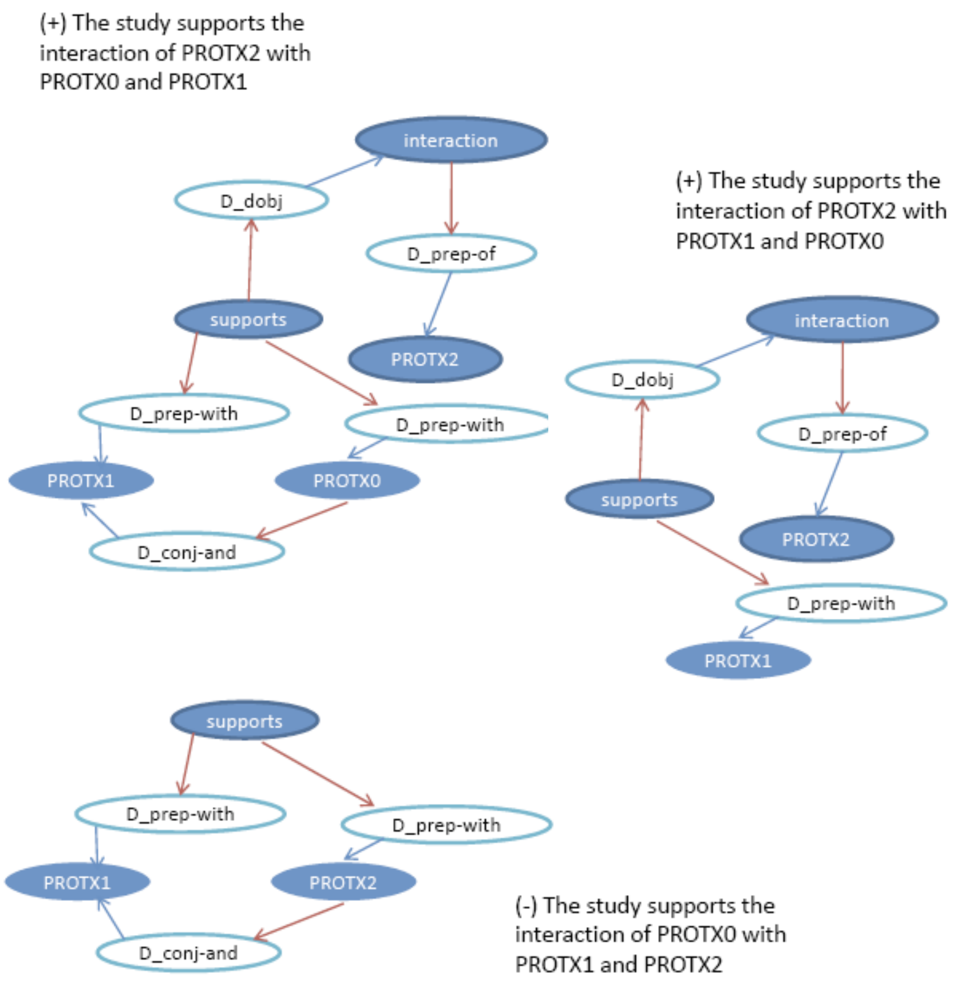

(+) The study supports the interaction of PROTX2 with PROTX0 and PROTX1
interaction
D_dobj
D_prep-of
supports
PROTX2
D_prep-with
D_prep-with
PROTX1
PROTX0
D_conj-and
(+) The study supports the interaction of PROTX2 with PROTX1 and PROTX0
interaction
D_dobj
D_prep-of
supports
PROTX2
D_prep-with
PROTX1
supports
D_prep-with
D_prep-with
PROTX1
PROTX2
D_conj-and
(-) The study supports the interaction of PROTX0 with PROTX1 and PROTX2

Figure 2.7: Pruned annotation graph for three proteins pairs in a sentence. The red arrow denotes a *GovToRel* relation and the blue arrow denotes a *RelToDep* relation. (+) is used to indicate a positive instance i.e. a sentence that suggests interaction between PROTX1 and PROTX2 protein mentions.

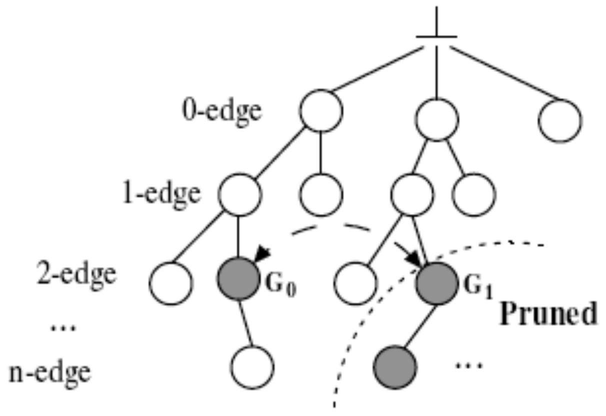


Figure 2.8: Subgraph Search Space: DFS Code Tree from (Yan and Han, 2002a).

2.2.1 Frequent Subgraph Mining

The goal in frequent subgraph mining is to find frequent subgraphs in a collection of graphs. A graph G' is a subgraph of another graph G if there exists a subgraph isomorphism[11] from G' to G, denoted by $G' \sqsubseteq G$. Earlier approaches in frequent subgraph mining (Inokuchi et al., 2000; Kuramochi and Karypis, 2002) used a two-step approach of first generating the candidate subgraphs, and then testing their frequency in a graph database. The second step involves a subgraph isomorphism test, which is NP-complete. Although efficient algorithms have been developed for isomorphism testing, with lots of candidate subgraphs to test it can still be very expensive for real applications.

gSpan (Yan and Han, 2002a) uses a pattern growth based approach to frequent subgraph mining. For each discovered subgraph G', new edges are added recursively until all frequent supergraphs of G' have been discovered. Figure 2.8, taken from Yan and Han (2002a), shows the search space for gSpan algorithm which finds frequent subgraphs one by one, from small to large. Each node in this search space represents a n-edge subgraph that grows from an (n-1)-edge subgraph. Each node is assigned a lexicographic label. Smaller the lexicographic label for a subgraph, earlier it is discovered in the search space. Note that two nodes in the search space may represent the same subgraph. gSpan determines duplicate subgraphs in the search space based on a unique canonical label for a graph, which we now describe.

gSpan uses depth first search (DFS) for frequent subgraph mining. A graph can have several different *DFS trees*, depending on where you start the search. Depth first discovery of nodes in a graph forms a linear order. Nodes are labeled with subscripts based on this order. Yan and Han (2002a) define a linear order for edges in a graph based on the linear order of the nodes. This linear oder is used to define an edge sequence, called the *DFS code*, to represent a given DFS tree for a graph. Yan and Han (2002a) also define a lexicographic order for DFS codes (for DFS trees of a graph), which is used to define the *Minimum DFS code* for a graph. The minimum DFS codes for two graphs are equivalent, if the graphs are isomorphic. Thus, mining frequent connected subgraphs is equivalent to mining their minimum DFS codes.

A *DFS Code Tree* (Figure 2.8) is defined to model the relationship among subgraphs, where a node in the tree represents the DFS code for a subgraph. Given a DFS code $\alpha = \{a_1, a_2, ..., a_m\}$[12],

[11] `http://en.wikipedia.org/wiki/Subgraph_isomorphism_problem`

[12] a_i and b are edges used to define the DFS code. An edge is defined in terms its label, and labels and subscripts

any valid DFS code of the form $\beta = \{a_1, a_2, ..., a_m, b\}$ is called its child in the DFS code tree. For β to be a valid DFS code, b must be an edge that grows from the vertices on the rightmost path in the graph with code α. With a depth first search of the DFS code tree, all minimum DFS codes for the frequent subgraphs can be discovered, that is all frequent subgrahs can be discovered.

To avoid unnecessary computation of duplicate subgraphs and their descendants, gSpan performs pre-pruning using the minimum DFS code check. Any subgraph whose DFS code is not the same as the minimum DFS code for that subgraph, is pruned. From the lexicographic ordering followed in the subgraph search, we know that a subgraph with DFS code other than the minimum code has already been discovered.

If measured in terms of the subgraph and/or graph isomorphism tests, the runtime of gSpan is bound by $O(kFS + rF)$ (Yan and Han, 2002b), where k is the maximum number of subgraph isomorphisms existing between a frequent subgraph and a graph in the dataset, F is the number of frequent subgraphs, S is the dataset size, and r is the maximum number of duplicate DFS codes of a frequent subgraph that grows from other minimum codes. kFS is the bound for the number of isomorphism tests that should be done in order to find frequent supergraphs from discovered frequent subgraphs, and rF is the bound for the maximum number of minimum DFS code checks. k is usually small for sparse graphs with diverse labels. In the worst case, with two complete graphs with no labels, k (the possible subgraph isomorphisms between them) can be P_m^n (P represents the Permutation), where m and n are the number of vertices in the two graphs ($m \leq n$). For any subgraph G, the number of duplicates that grow from other minimum DFS code, is bound by the product of the number of vertices and edges (Theorem 6 in (Yan and Han, 2002b)). Thus, r is bounded by the maximum product of the number of edges and vertices that a frequent subgraph has. Because of the pre-pruning based on the minimum DFS code in gSpan, r is usually much smaller than that.

With several implementations available[13], gSpan has been commonly used for mining frequent subgraph patterns from graphs (Kudo et al., 2004; Deshpande et al., 2005). In this work, we use gSpan to mine frequent subgraphs from the annotation graphs. At the end of Section 2.4, we present a performance analysis of gSpan algorithm for mining frequent subgraphs from annotation graphs of varying complexity, and varying minimum frequency for subgraphs, called the *minimum support threshold*. Next we discuss the subgraph matching algorithm we use for finding a given frequent subgraph in the annotation graph for a test instance.

2.2.2 Subgraph Matching

Given a set of frequent subgraphs from the training data, we need to find their occurrences in the annotation graph of a test instance. This requires a subgraph isomorphism test between a frequent subgraph and the test annotation graph. We used the code developed by Lei Zhao[14] for subgraph matching, which uses the VFLIB library[15]. VFLIB provides three algorithms for subgraph matching - UL (Ullmann, 1976), VF (Cordella et al., 1999) and VF2 (Cordella et al., 2001). Ullman's algorithm (Ullmann, 1976), although quite old, is still the most commonly used algorithm for graph matching. It is based on a backtracking procedure with an effective look ahead to reduce the search space. Cordella et al. (1999) proposed a faster graph match algorithm called VF algorithm, with the worst case complexity better than Ullman's algorithm. VF algorithm uses a set of feasibility rules based on syntax and semantics of the graph to foresee if a state in

(defined based on their order of discovery in depth first search) of its vertices.

[13]`http://www.cs.ucsb.edu/~xyan/software/gSpan.htm`, `http://www.kyb.mpg.de/bs/people/nowozin/gboost/`

[14]a student of Dr. Xifeng Yan - the author of gSpan

[15]http://amalfi.dis.unina.it/graph/db/vflib-2.0/doc/vflib.html

(D_advmod)_Edge_ParentOfDep _(U_too)
U_too
U_bad
U_movie
(D_amod)_Edge_ParentOfDep _(U_bad)

Table 2.2: Top unigram and subgraph features for Movie Snippets dataset based on χ^2 score.

the search space has no coherent successors after a certain number of steps (k-lookahead). As reported in (Cordella et al., 1999), the worst case complexity of the Ullman algorithm for matching two graphs with N nodes is $O(N!N^3)$, compared to $O(N!N)$ for the VF algorithm. The $VF2$ algorithm proposed by Cordella et al. (2001) is an enhancement of the VF algorithm that reduces the memory requirement by reorganizing the exploration of the search space. The worst case memory requirement of $VF2$ algorithm is $O(N)$, a substantial improvement over $O(N^2)$ for VF algorithm. The memory requirement of Ullman algorithm is $O(N^3)$.

We found the computation time for VF and $VF2$ algorithm to be quite similar for our annotation graphs. Thus, we selected one of them (VF) for subgraph matching. We didn't face any memory issues with VF algorithm for the annotation graphs for our datasets. At the end of Section 2.4, we present a performance analysis of the subgraph matching algorithm for subgraphs with different minimum support threshold, and annotation graphs of varying complexity.

2.3 Feature Selection

A challenge to overcome when using an automatic feature extraction approach is the large feature space size. Among the large set of subgraph features discovered through frequent subgraph mining, many are not predictive, or are very weak predictors, and only a few carry novel information that improves classification performance. Hence, directly adding many complex features to the model may not improve performance, or may even worsen performance in some cases, as the feature space's signal is drowned out by the noise.

Feature selection based on class association scores, such as information gain, χ^2 score, etc., are quite popular in the NLP community, and have been shown to work well for several NLP tasks (Yang and Pedersen, 1997). However, in our feature space, several top ranked features (ranked based on the χ^2 score) may be highly correlated. For example, the top 5 features based on χ^2 score in one of our experiments, are shown in Table 2.2; it is immediately obvious that the features are highly redundant.

Riloff et al. (2006) propose a *feature subsumption* approach to selectively add complex/structured features to the model. They define a hierarchy for features based on the information they represent. A complex feature is only added to the model if its discriminative power, measured in terms of its information gain, is a delta above the discriminative power of all its simpler forms.

Another approach to feature selection is a correlation based filter, where structured features highly correlated in their occurrence with their simpler forms are not added to the model.

Genetic Programming (Koza, 1992) has also been used for selecting most beneficial feature combinations. A genetic programming based approach evaluates interactions between features and evolves complex features from them. The advantage of a genetic programing based approach over other feature selection methods described above is that it allows us to evaluate a feature using multiple criteria. In Genetic Programming (GP), potential solutions are represented as trees consisting of *functions* (non-leaf nodes in the tree, which perform an action given their child nodes as input) and *terminals* (leaf nodes in the tree, often variables or constants in an equation). A tree (an individual) can be interpreted as a *program* to be executed, and the output of that program can be measured for *fitness* (a measurement of the program's quality). High-fitness individuals are selected for reproduction into a new generation of candidate individuals through a breeding process, where parts of each parent are combined to form a new individual.

Mayfield and Rosé (2010) apply this design to a language processing task at the stage of feature construction - given many weakly predictive features, they are combined to produce a more predictive feature. They show that genetic programming based feature construction is effective when using unigram features as input for constructing complex features, for a sentiment classification task. For the functions, boolean statements, such as AND and XOR, are used, while terminals are selected randomly from the feature set. Each leaf's value, when applied to an instance, is equal to 1 if that feature is present in the instance and 0 otherwise. For the fitness function, they use precision and recall of an individual feature in predicting the class. A penalty is added to avoid trees from becoming overly complex, and another penalty is added based on correlation with the unigram features, from which the feature combinations are generated. In the next section, we present our experiments and results, using subgraph features from annotation graphs, and some of the feature selection methods discussed above.

2.4 Experiments and Results

We evaluate our approach on two tasks: a sentiment classification task, where the goal is to classify a movie review sentence as expressing positive or negative sentiment towards the movie, and a protein-protein interaction extraction task, where the goal is to classify a sentence as suggesting that a protein pair mentioned in it interacts.

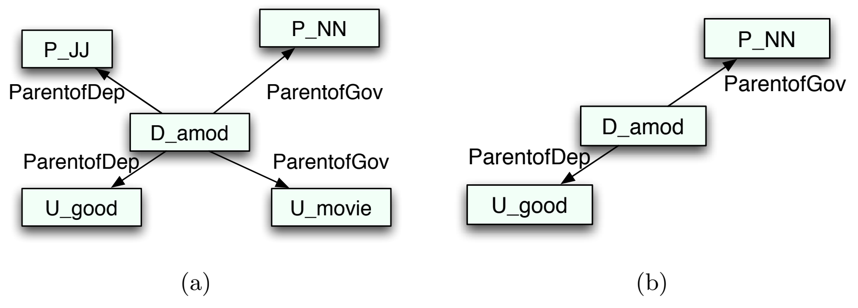
2.4.1 Data and Experimental Setup

For the sentiment classification task, we used one of the movie review datasets shared by Pang and Lee (2005)[16]. This dataset consists of 10662 snippets/sentences from the Rotten Tomatoes website[17], with an equal number of positive and negative sentences (5331 each). This dataset is different from the more common movie review dataset by Pang and Lee (2004), which consists of document length movie reviews. The snippets dataset was created and used by Pang and Lee (2005) to train a classifier for identifying sentences with positive sentiment in a full length review. The reason for not using the full movie review dataset, and instead using the movie snippets data was to simplify the annotation graph representation for our first experiment, as snippets are much shorter (about 1 sentence) than the movie review documents (on average 32 sentences). We use the first 8000 (4000 positive, 4000 negative) sentences as training data, and evaluate on the remaining 2662 (1331 positive, 1331 negative) sentences. We added parts of speech and dependency triple annotations to this data using the Stanford parser (Klein and Manning, 2003).

Annotation Graph: For the annotation graph representation, we used *Unigrams (U)*, *Parts of Speech (P)* and *Dependency Relation Type (D)* as labels for the nodes, and *ParentOfGov* and *ParentOfDep* as labels for the edges. Following the work in (Matsumoto et al., 2005), we filter the unigrams that occur only once in the training data. We also use stop word lists for unigrams, parts of speech (POS), and dependency relations. For unigrams, we used the stop-word list from

[16]`http://www.cs.cornell.edu/people/pabo/movie-review-data/rt-polaritydata.tar.gz`

[17]`http://www.rottentomatoes.com/`



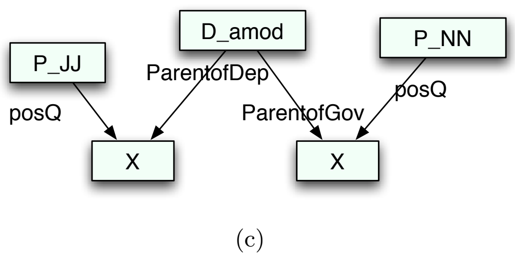


Figure 2.9: Annotation graph and a feature subgraph for dependency triple annotation "amod_good_camera". (c) shows an alternative representation with wild cards.

(Manning et al., 2008) (with one modification: removed 'will', added 'this'[18]). For POS, we used the stop word list from (Matsumoto et al., 2005) (with one modification: removed IN). For dependency relations, we used a small stop-word list, consisting of the following relations: {det, predet, preconj, prt, aux, auxpas, cc, punct, complm, mark, rel, ref, expl}, most of these are relations associated with stop words or punctuations.

For a dependency triple such as "amod_good_movie" and parts of speech for words 'good' and 'movie', five nodes are added to the annotation graph, as shown in Figure 2.9a. *ParentOfGov* and *ParentOfDep* edges are added from the dependency relation node *D_amod* to the unigram nodes *U_good* and *U_movie*. These edges are also added for the parts of speech nodes that correspond to the two unigrams in the dependency relation, as shown in Figure 2.9a. This allows the algorithm to find general patterns, such as dependency relation between two parts of speech nodes, two unigram nodes or a combination of the two. For example, a subgraph in Figure 2.9b captures a general pattern where the word *good* modifies a noun. This feature exists in "amod_good_movie", "amod_good_camera" and other similar dependency triples. These features are similar to the dependency back-off features proposed in (Joshi and Rosé, 2009). The extra edges are an alternative to putting wild cards on the words, suggested in an example in Section 2.1 (Figure 2.3). Putting a wild card on every word in the annotation graph for our example (Figure 2.9c), will only give features based on dependency relations between parts of speech annotations, and not between words. Thus, the wild card based approach is more restrictive than adding more edges. However, with many edges, the complexity of the annotation graph increases, which in turn increases the computation time for subgraph mining algorithm.

[18]The stop-word list from (Manning et al., 2008) contains the word 'that' but not 'this', so we added it. Also, the word 'will' may be associated with sentiment (e.g. I will go and see the movie again), hence we removed it from the stop-word list.

	V	E
Average #	61	51
Max #	178	156
Min #	2	0
STDEV #	28	25
Labels #	18748	2

Table 2.3: Statistics of the annotation graphs (average, maximum, minimum and standard deviation for the number of vertices (V) and edges (E), and the total number of labels for the vertices and edges) for Movie Review sentences (10662 sentences).

Table 2.3 shows some statistics about the complexity of our annotation graphs. Comparing our graphs to other biological datasets, where gSpan based algorithms have been used (Thoma et al., 2009), we have a larger label set for our nodes. This suggests that our graphs are sparse, and hence we may need to use a lower minimum frequency threshold. This also suggests that the gSpan algorithm should run faster on our annotation graphs, based on the complexity analysis of gSpan presented in Section 2.2.

Classifier: For our experiments we use Support Vector Machines (SVMs) with a linear kernel. We use the SVM-light[19] implementation with default settings.

Parameters: The *gSpan* algorithm requires setting the minimum support threshold ($minSup$) for the subgraph patterns to extract. Support for a subgraph is the number of graphs in a dataset that contain that subgraph. We experimented with several values for minimum support. We report the results for $minSup = 3$ and $minSup = 5$.

Feature Configurations: The dataset we used for our experiments was created and used by Pang and Lee (2005), to train a sentence level sentiment classifier for identifying positive sentences in movie review documents. They did not report any performance results for the sentence classifier. To the best of our knowledge, there is no other supervised machine learning result published on this dataset. We compare the following feature configurations in our experiments:

1. *Unigram-only:* In sentiment classification, unigram features (bag-of-words) are a strong baseline (Pang et al., 2002; Pang and Lee, 2004). As for constructing the annotation graphs, we use unigrams that occur in at least two instances in the training data, and filter out stop words using a small stop word list[20].

   For our training data, after filtering infrequent unigrams and stop words, we get 8424 features. Adding subgraph features increases the total number of features to $575,200$ (for $minSup = 5$), a factor of 68 increase in size. Feature selection can be used to reduce this size by selecting the most discriminative features. Following configurations combine unigram and subgraph features, and uses one of the feature selection methods discussed in Section 2.3:

2. *Genetic Programming (GP) based feature combinations:* We extend the genetic programming approach in (Mayfield and Rosé, 2010) to our subgraph features. For our functions we use the same boolean statements (AND and XOR) as in their work, while our terminals are selected

[19]`http://svmlight.joachims.org/`

[20]`http://nlp.stanford.edu/IR-book/html/htmledition/dropping-common-terms-stop-words-1.html` (with one modification: removed 'will', added 'this')

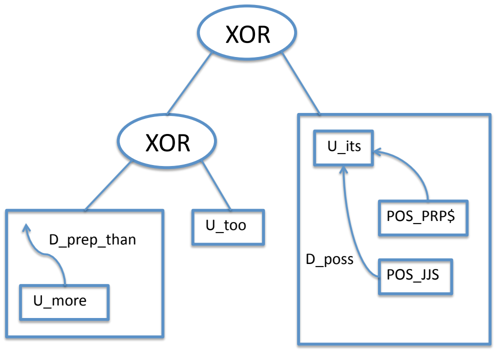


Figure 2.10: A tree constructed using subgraph features and GP (Simplified for illustrative purposes).

randomly from the set of unigram and subgraph features. For feature construction, we divide our training data in half, and train our GP features on one half of this data. This is to avoid overfitting the final SVM model to the training data. In a single GP run, we produce one feature for each class. For the sentiment classification task, a feature is evolved to be predictive of the positive instances, and another feature is evolved to be predictive of the negative instances. We repeat this procedure a total of 15 times (using different seeds for random selection of features), producing a total of 30 new features. We use the same fitness function for feature evolution as (Mayfield and Rosé, 2010):

$$\text{Fitness} = F_\beta + PP + CC \tag{2.1}$$

where F is the F-score defined based on precision and recall of the feature for the positive and negative class. β is the relative weight on recall over precision. PP is the parsimony pressure penalty to avoid overly large and complex trees. CC is a correlation[21] penalty which penalizes correlation between the feature being constructed, and the subgraphs and unigrams it is constructed from. The correlation is measured between the vectors for feature occurrences in the training set. The correlation penalty (CC) for a constructed feature is the maximum correlation of that feature with any of the features it is constructed from, minus a correlation threshold.

The tree in Figure 2.10 is a simplified example of our evolved features. It combines three features, a unigram feature ‘too’ (centre node) and two subgraph features: 1) the subgraph in the leftmost node occurs in collocations containing *“more than”* (e.g., *“nothing more than”* or *“little more than”*), 2) the subgraph in the rightmost node occurs in negative phrases such as *“opportunism at its most glaring”* (JJS is a superlative adjective and PRP$ is a possessive pronoun). A single feature combining these weak indicators can be more predictive than any one of them.

For genetic programming based feature construction, we use the ECJ toolkit[22]. We use the

[21]Measured in terms of the Pearson’s product-moment (`http://en.wikipedia.org/wiki/Pearson_product-moment_correlation_coefficient`).

[22]`http://cs.gmu.edu/~eclab/projects/ecj/`

same parameter settings as described in (Mayfield and Rosé, 2010)[23], which were tuned on a different dataset[24] than the one used in this work, although it is from the same movie review domain. The parsimony pressure penalty (PP) is set to 0.005 per node, and threshold for correlation penalty is set to 0.5. That is, the correlation penalty (CC) for a feature with maximum correlation of 0.8 with features it is constructed from, is 0.3. Mayfield and Rosé (2010) set the β ratio for F-measure to $\frac{1}{6}$, which gives 6 times more weight to precision. Since we only create 30 new GP feature combinations from all the subgraph and unigram features, we believe that a stricter constraint must be placed to find a precise subset of feature combinations. We tried a few other values of the β parameter, and $\beta = \frac{1}{15}$ gave us the best performance.

3. *χ^2 feature selection:* χ^2 feature selection (Yang and Pedersen, 1997) is commonly used in the literature. We compare two methods of feature selection with χ^2, one that selects features with χ^2 score significant at 0.05 level, and second that selects the top N features to match the size of our feature space with GP based feature selection.

4. *Feature Subsumption (FS):* Following the idea in (Riloff et al., 2006), a complex feature C is discarded if $IG(S) \geq IG(C) - \delta$, where IG is the Information Gain, and S is a simple feature that *representationally subsumes* C, i.e. the text spans that match S are a superset of the text spans that match C. In our work, complex features are subgraph features and simple features are unigram features contained in them. For example, $(D\_amod)\_Edge\_ParentOfDep\_(U\_bad)$ is a complex feature for which U_bad is a simple feature. We tried the same values for $\delta \in \{0.002, 0.001, 0.0005\}$, as Riloff et al. (2006). Since all values gave us the same number of features, we only report one result for feature subsumption.

5. *Correlation (Corr) based feature selection:* As mentioned earlier, some of the subgraph features are highly correlated with unigram features and do not provide new information. A correlation based filter for subgraph features can be used to discard a complex feature C, if its absolute correlation (measured between the feature occurrence vectors) with its simpler feature S (unigram feature) is more than a certain threshold. We use the same threshold (0.5) as used in the GP criterion, but as a hard filter instead of a penalty.

We also evaluate the proposed approach for a Protein-Protein Interaction (PPI) extraction task, where the goal is to judge whether a given sentence suggests that a pair of proteins mentioned interact. A sentence may mention more than two proteins. As described in Section 2.1.1, a sentence is replicated for each protein pair mentioned. That is, there are multiple instances per sentence, one for each pair of proteins. As mentioned before, to reduce sparsity, PROTX1 and PROTX2 are used to represent the protein pair under consideration, and PROTX0 is used to represent all other proteins mentioned in the sentence. For our experiments we use the AIMED dataset (Erkan et al., 2007). This dataset consists of 951 positive, and 3075 negative instances of protein-protein interaction. We report results for ten fold cross validation. Since we may have more than one instance per sentence, instances from the same sentence are put in the same fold. That is, we group the instances from the same sentence together, and create folds over these groups. We create the folds such that there is a similar ratio of positive to negative instances in each fold. We construct the annotation graphs using unigram and dependency triple annotations, using the same annotation packages as the Movie Review data. However, for this dataset, we did not filter any stop

[23]We thank Elijah Mayfield, for his help with running the GP experiments.
[24]Full movie review data (Pang and Lee, 2004)

SNo.	**Settings**	**minSup**	**#Features**	**Acc.**	**Δ**
1	Uni	-	8424	75.66	-
2	Uni + Sub	5	575,200	66.08	-9.58
3	Uni + Sub	3	7,019,792	64.58	-11.08
4	Uni + Sub, χ^2 sig.	5	172,338	68.82	-6.84
5	Uni + Sub, χ^2 sig.	3	2,828,488	68.52	-7.14
6	Uni + Sub, χ^2 size	5	8454	68.44	-7.22
7	Uni + Sub, (FS)	5	25,895	71.04	-4.62
8	Uni + Sub, (Corr)	5	482,073	64.65	-11.01
9	Uni + GP (U) ‡	-	8454	75.96	0.36
10	Uni + GP (U) †	-	8454	76.11	0.45
11	Uni + GP (U+S) ‡	5	8454	76	0.34
12	Uni + GP (U+S) †	5	8454	76.6	0.94
13	Uni + GP (U+S) ‡	3	8454	76.07	0.41
14	Uni + GP (U+S) †	3	8454	**76.82**	1.16

Table 2.4: Experimental results for feature spaces with unigrams and subgraph features. Feature selection with 1) fixed significance level (χ^2 sig.), 2) fixed feature space size (χ^2 size), 3) Feature Subsumption (FS) and 4) Correlation based feature filtering (Corr)). GP features for unigrams only {GP(U)}, or both unigrams and subgraph features {GP(U+S)}. Both the settings from Mayfield and Rosé (2010) (‡) and more weight on precision ($\beta = \frac{1}{15}$) (†) are reported. *#Features* is the number of features in the training data. *minSup* is the minimum support threshold for frequent subgraph mining. *Acc* is the accuracy, and Δ is the difference in accuracy from unigram-only baseline. Best performing feature configuration is highlighted in bold which is marginally significantly better than unigram based on 90% confidence interval calculated using one-way anova.

words or annotations. We normalize the instance feature vectors to unit length. Doing the same for the sentiment classification task, did not give us any improvements. We use SVMs with linear kernel for learning, like for the sentiment classification task. We set the C parameter in SVMs (the cost for miss satisfying the margin constraints) to a commonly used value of 1, and $minSup = 5$. We found that compared to the movie review dataset, relatively many more subgraphs are found for this dataset for $minSup = 5$, and many of these subgraphs are large. Very large subgraphs may not generalize across many instances, and thus we limit the maximum subgraph size (L) to 7 nodes, when mining for frequent subgraphs. As mentioned in Section 2.1.1, for PPI task, there are many ways to construct the annotation graph for instances in a sentence. We experiment with complete annotation graphs and pruned annotation graphs constructed from the local context of the protein pair in a sentence, as described in Algorithm 1. For PPI task, we use F-score to measure performance, since the number of positive and negative instances are not equal in the test set, and F-score is commonly used for evaluation in this task (Erkan et al., 2007). For both datasets, we use binary feature occurrence vectors for instance representation.

Movie Review Snippet	**Class (POS/NEG)**
the importance of being earnest , so thick with wit it plays like a reading from bartlett's familiar quotations	POS
the events of the film are just so weird that I honestly never knew what the hell was coming next	POS
in the poor remake of such a well loved classic, parker exposes the limitations of his skill and the basic flaws in his vision	NEG
initial strangeness inexorably gives way to rote sentimentality and mystical tenderness becomes narrative expedience	NEG
starts off witty and sophisticated and you want to love it – but filmmaker yvan attal quickly writes himself into a corner	NEG

Table 2.5: Some difficult examples from movie review domain where a model fails at positive/negative sentiment classification.

2.4.2 Results and Discussion

In Table 2.4, we present our results[25] for the sentiment classification. In sentiment classification, bag-of-words has been a very strong baseline (Pang et al., 2002; Pang and Lee, 2004), and linguistic features have been found to provide little benefit over the unigrams. We believe one reason for this is sparsity. Movie reviews are written by reviewers with different backgrounds, who may have different writing styles. Also, movie reviews are very informal, reviewers use incomplete sentences, sarcastic language, etc. Table 2.5 presents some of the hard examples for an algorithm to classify correctly. To learn accurate weights for a feature, it must be repeated in the data at least a few times, and be associated with one class more than the others. Lexical patterns (based on word/tokens relations) are sparse. With linguistic annotations on text, we hope to discover more general patterns that capture the linguistic structure in text, and are more frequent. Note that the goal of this research is not to show that the subgraph features outperform unigram features on a given sentiment classification task. Instead the goal is to demonstrate that linguistic patterns relevant for a classification task, that were undiscovered earlier or were handcrafted, can now be automatically discovered as subgraphs from the annotation graphs, without the additional human effort, and they improve performance of a bag-of-words model.

Automatically extracting subgraph features generates a large number of features especially when the minimum support threshold (minSup) is low. As can be seen from Table 2.4, there are about $500,000$ features for $minSup = 5$ (row 2), and about 7 million features for $minSup = 3$ (row 3). Our first strategy was to use all these features in an SVM model, which is known to be robust to a large feature space. As can be seen, subgraph features when added to the unigrams without any feature selection decrease performance substantially (second and third rows in Table 2.4). This is possibly because of the increased feature space size with many redundant features among subgraphs.

Our next approach was to use a popular feature selection strategy based on class association scores, such as χ^2, to select the most relevant features. We tried two versions of χ^2 based feature

[25]These results are a revision of the results published in (Arora et al., 2010). A problem was discovered in the frequent subgraph mining package (gSpan). It gives inaccurate results for directed graphs. We recreated our annotation graphs as undirected graphs and reran the experiments. However, the best performance is only 0.11 lower than what was reported in the paper (76.93). Note that that the node and edge labels we use here encode the same information whether we represent them as directed or undirected graphs.

selection, as described in Section 2.4.1, one based on significance and other based on fixed feature space size. The performance improves (compare rows 2 and 3 with rows 4, 5 and 6), but it is still lower than unigram-only features. We also tried other feature selection methods discussed in Section 2.4.1. Feature subsumption improves the performance substantially, but the performance is still lower than the unigram-only approach (compare row 7 with rows 1 and 2). Even with a correlation-based filter (row 8), the performance of unigram+subgraph features is much lower than the unigram-only features[26].

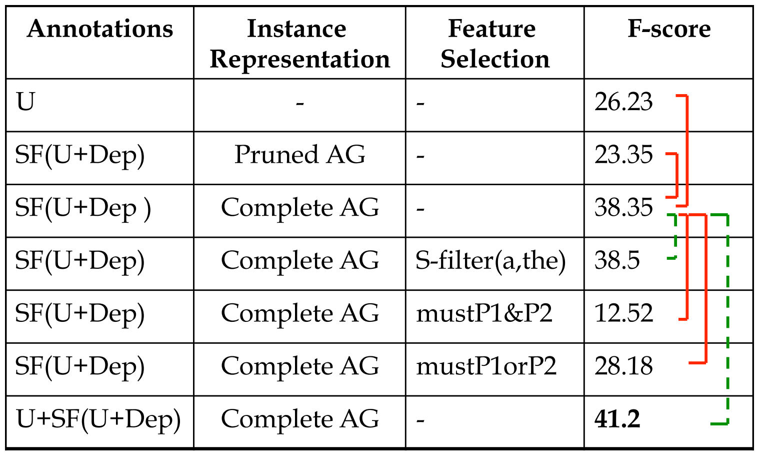
With GP-based feature construction, we create a few (30) most useful combination of features from the set of all unigram and subgraph features. When these GP feature combinations are added to the unigram features, we get a gain in performance over unigrams for both $minSup = 5$ (row 11) and $minSup = 3$ (row 13). The performance improves further when we give more weight to precision than in prior work (Mayfield and Rosé, 2010) (compare † and ‡). For $minSup = 3$, with more weight on precision (row 14), we observe a marginally significant gain ($p < 0.1$) in performance over unigrams alone[27]. GP-based feature construction can also be applied to unigram features only (Rows 9 and 10), like in (Mayfield and Rosé, 2010). However, as can be seen, GP-based feature construction with unigram and subgraph features (Rows 11-14) outperforms GP-based feature construction with only unigram features (Rows 9-10). The improvement over unigram-only features is marginally significant only when subgraph features are used in addition to the unigram features, for constructing more complex feature combinations using GP (compare rows 9-10 and 11-14 to row 1, in Table 2.4).

A problem that we see with χ^2 feature selection, as also discussed in Section 2.3, is that several top-ranked features may be highly correlated. With GP-based feature construction, we can consider this relationship between features, and construct new features as a combination of selected unigram and subgraph features. With the correlation penalty in the evolution process, we are able to build combinations of features that provide new information to the model beyond unigram features. So far, we have only varied the β parameter in GP configuration. Further improvement may be achieved by varying the other parameters and constructing more features using GP.

Figure 2.11 shows the results for the PPI task. As can be seen, using the complete annotation graphs for extracting subgraph features performs significantly better than using pruned annotation graphs. Note that the computation time for frequent subgraph mining and subgraph matching algorithms would be more when using complete annotation graphs instead of the pruned annotation graphs. However, creating pruned annotation graphs from complete annotation graphs would also consume some time. As can also be seen from Figure 2.11, subgraph features by themselves (without unigram features) significantly outperform unigram-only features, with a substantial gain of more than 10 points in F-score. We tried three simple feature selection techniques: 1) discard the subgraph features with stop words 'a' and 'the' (*S-filter(a,the)*), 2) only keep subgraph features that contain both the proteins under consideration (*mustP1&P2*), and 3) only keep subgraph features that contain at least one of the proteins under consideration (*mustP1orP2*). Since a sentence may contain more than one pair of proteins, the last two techniques for feature selection aim at selecting a subset of features relevant for the protein pair under consideration. We find

[26]We only have results for $\chi^2 - size$, feature subsumption and correlation based feature selection for $minSup = 5$. We weren't able to run these feature selection methods for $minSup = 3$ with the hardware we had, as it required scoring and ranking 7 million features. Given that they didn't make a lot of difference in performance for $minSup = 5$ (performance is still much lower than unigram-only features), we don't expect the result to be very different for $minSup = 3$.

[27]Assuming a normal distribution and computing a one-sided confidence interval, as in Section 5.2 of (Witten and Frank, 2005)



Annotations	Instance Representation	Feature Selection	F-score
U	-	-	26.23
SF(U+Dep)	Pruned AG	-	23.35
SF(U+Dep)	Complete AG	-	38.35
SF(U+Dep)	Complete AG	S-filter(a,the)	38.5
SF(U+Dep)	Complete AG	mustP1&P2	12.52
SF(U+Dep)	Complete AG	mustP1orP2	28.18
U+SF(U+Dep)	Complete AG	-	**41.2**

Figure 2.11: Comparison of structured features with unigram features for Protein-Protein Interaction task. Solid lines show pairs of significant differences ($p < 0.05$), determined using two-tailed paired t-test. The dashed lines show the differences that are not statistically significant.

that using *S-filter(a,the)* does not make a significant difference in performance. Restricting the features to only those with one or both the protein names, is significantly worse than not using any feature selection technique. This suggests that clues for interaction between proteins may not be necessarily mentioned close to where the protein names are mentioned. Our initial experiments with χ^2 based feature selection and GP based feature construction suggested that there is little benefit in using these feature selection methods for this task. The performance was either worse or not significantly different from using all subgraph features from the complete annotation graphs. However, here we used the GP configuration similar to the Movie Review dataset. There are other variations one could try to the GP fitness function. Adding unigram features to subgraph features improves performance by 2.85 points (rows 3 and 7), although this difference is not significant.

Thus, the improvement in performance with subgraph features over unigram-only features is substantially more for PPI task than sentiment classification task. As mentioned before, movie reviews are written by reviewers from different backgrounds, about different kinds of movie, the language is informal (there are often spelling mistakes, and short forms are often used), and there are a variety of ways to express sentiment. Thus, we may need a very large dataset to discover useful patterns. In PPI task, the goal is to extract mentions of protein-protein interactions in scientific articles, where the language is formal, and as we also found in our experiments, frequent patterns are larger and more frequent for this task. Thus, we see more gain from structured features for PPI task, than sentiment classification task. Figure 2.12 shows an example of a frequent subgraph feature for PPI task. Linguistic patterns like this are expected to be beneficial for identifying protein-protein interaction mentions in text.

We cannot directly compare our work with Erkan et al. (2007), as their folds were not created such that all instances from a sentence are in the same fold. They randomly shuffled the instances before creating the folds, so the distribution of positive and negative instances in each fold should be roughly the same. We explicitly create the folds such that the distribution of positive and negative instances is similar in each fold. Erkan et al. (2007) use the shortest path in the dependency tree, to represent the instance for a protein pair in a sentence. They propose two kernels, based on the cosine similarity (SVM-cos) and edit distance (SVM-edit). They report an F-score of 55.61 for

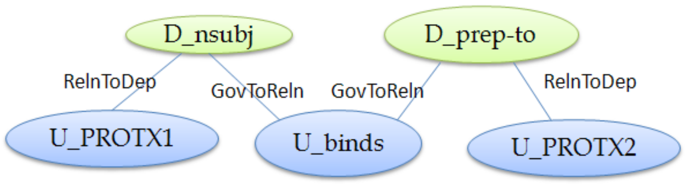


Figure 2.12: An example of a frequent subgraph feature for Protein-Protein Interaction (PPI) extraction task.

SVM-edit, and an F-score of 58.09 for SVM-cos kernels. A kernel based on cosine similarity, as described in their work, measures the number of nodes and edges in common between paths for two instances. In our approach, we use a linear kernel with subgraph features, which measures the number of frequent subgraph features (with minimum support of 5 ($minSup = 5$) and maximum size of 7 nodes ($L = 7$)) in common between the full annotation graphs. A next step would be to experiment with the shortest path as the pruned annotation graph, and to try other values of $minSup$ and L. Note that with the shortest path as the pruned annotation graph, and $minSup = 1$ and $L = 1$, our approach is similar to theirs. Cosine similarity, as described in their work, normalizes the kernel for paths of different length. For this dataset, we normalize the instance feature vectors to unit length[28], before computing the kernel, which has a similar effect. Erkan et al. (2007) do not report the value they use for the C parameter in SVMs (the cost for miss satisfying the margin constraints). We fix the C parameter to a commonly used default value of 1. One could also experiment with other values for the C parameter.

In addition to our work, Rink et al. (2010) use a similar approach for learning causal relations between event mentions in text. They define the annotation graph with syntactic and semantic annotations, such as parts of speech, syntactic parse tree, dependency parse tree, word sense, semantic roles and manually annotated temporal links. The features are extracted automatically as frequent subgraphs in the annotation graphs, using the gSpan algorithm, like in our work. They show that the automatically extracted subgraph features outperform the state of the art method for identifying causal relations between event mentions in the given dataset. We describe their work in more detail in Section 2.5, where we discuss the related work.

Thus, our experiments with automatically extracted structured features for sentiment classification and protein-protein interaction extraction, together with Rink et al. (2010)'s work on identifying causal relations between event mentions, show the potential of the proposed approach for reducing the cost of feature engineering. Next, we present an analysis of the computational performance of the frequent subgraph mining and matching algorithm, as we increase the complexity of the annotation graphs.

2.4.3 Performance Analysis for Subgraph Features

In the experiments and results presented above, we demonstrate that structured features can be used to boost the performance of a unigram-only model for a sentiment classification task and a protein-protein interaction extraction task. Apart from the well known SVM learning algorithm, the proposed approach involves a frequent subgraph mining algorithm for finding frequent subgraphs in the training data, a subgraph matching algorithm for finding the frequent subgraphs in the test annotation graphs, and a Genetic Programming approach to construct feature combinations

[28]Doing the same for the sentiment classification task, did not give us any improvements.

from unigram+subgraph features. For the movie review dataset, we present an analysis of the performance and computation time (run time) for these algorithms as we increase the complexity of the annotation graphs and reduce the minimum frequency threshold for finding the subgraphs. The runtimes are reported on a server (running LINUX OS) with 4 cores (Intel CPUs with 2.66GHz speed each) and a total of 16GB of RAM.

minSup	**# SG Features**	**Acc.**
10	98,137	76.22
5	566,776	76.6
3	7,011,368	76.82

Table 2.6: Experimental results with unigram features and GP based features (constructed from unigrams and subgraph features) as we decrease the minimum support threshold for subgraph features. The parameter setting for GP and annotations used are same as for the results presented in Table 2.4.

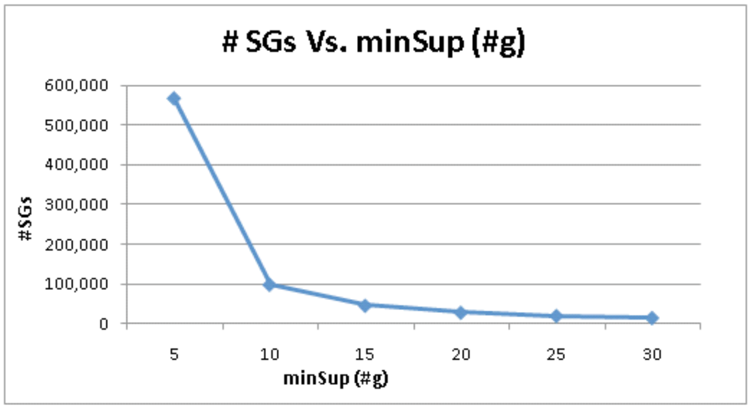


Figure 2.13: Number of subgraphs and minimum support threshold.

In the proposed approach, the features are derived as frequent subgraphs from the annotation graphs. As we lower the minimum support threshold (i.e. minimum frequency) for finding the frequent subgraph features, the number of subgraphs increase and GP algorithm is able to find good feature combinations that improve performance, as can be seen in Table 2.6. This supports our belief about the sparseness issue of features in the movie review domain. As can be seen from Figure 2.13, the number of subgraphs grow approximately exponentially as we reduce the minimum support threshold. Also, as we lower the minimum support threshold, the computation time of the subgraph mining algorithm (Figure 2.14) and the subgraph matching algorithm (Figure 2.15) grows approximately exponentially as we reduce the minimum support threshold. Thus, the improvement in performance comes with more computation cost.

We also wanted to study the effect of the number of annotations on the accuracy of the model and the computation cost. In order to do this, we compare two types of annotation graphs, one with tokens and dependency relations, and other with additional parts of speech (POS) annotations and extra edges, as discussed in Section 2.4.1 and Figure 2.9. As can be seen from Table 2.7, the

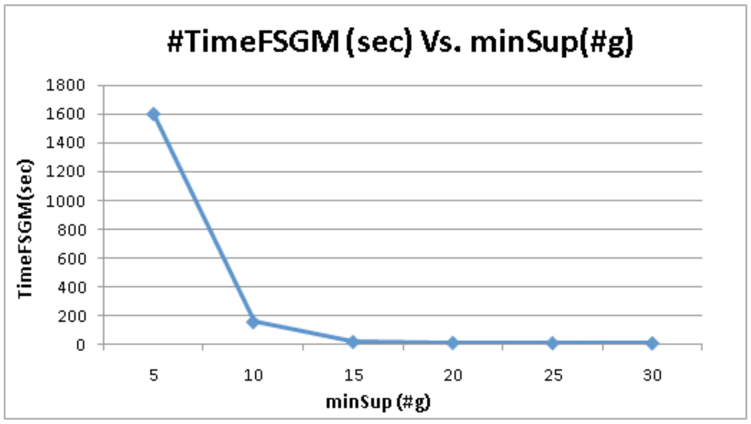


Figure 2.14: Time in seconds for frequent subgraph mining algorithm (gSpan) vs. minimum support threshold.

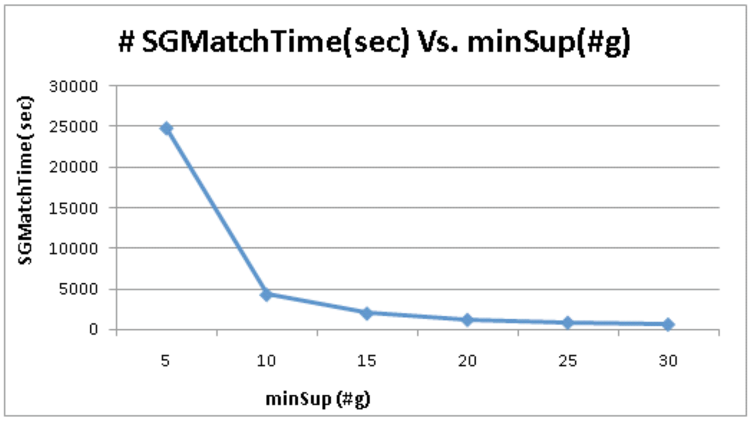


Figure 2.15: Time in seconds for subgraph matching algorithm vs. minimum support threshold.

model's accuracy improves as we add the POS annotations, but with additional annotations the graphs become more complex (average number of vertices and edges roughly doubles) and the computation times increase substantially. The number of frequent subgraphs (for $minSup = 5$) increases by approximately 50 times, and the computation time increases by approximately 500 times.

The computation time for training an SVM model, also increases as the number of features increase. Although the computation time for SVM has been found to increase only linearly with the average number of non-zero features per instance[29], we found it to be very slow with 7 million features for $minSup = 3$. When using GP based feature combinations for unigram and subgraph features, we only add 30 features to the unigram feature set, and hence the total feature set for training SVMs does not increase by much (8454).

With an increase in the total number of subgraph features, the search space for GP algorithm increases, and hence its computation time. Thus, we also analyze the computational time of

[29]`http://nlp.stanford.edu/IR-book/html/htmledition/soft-margin-classification-1.html`

Annotations	**maxV(avgV)**	**maxE(avgE)**	**#SG**	**Accuracy**	**FSGMTm**	**SGMTm**
uni+dep	112(38)	78(25)	11,974	76.07	3	300
uni+dep+pos	178(61)	156(51)	566,776	76.6	1601	24778

Table 2.7: Comparison of computation time and accuracy as annotation graph complexity increases with annotations. Accuracy reported is for the best parameter setting in Table 2.4 ($\beta = \frac{1}{6}$) and $minSup = 5$. The computation time is reported in seconds. FSGMTm: Time for frequent subgraph mining algorithm, SGMTm: Subgraph matching time. Annotations: uni - unigrams/tokens, dep - dependency, pos - parts of speech annotations.

minSup	**Annotations**	**# Features**	**Accuracy**	**GP-Time**
5	uni+dep	20,398	76.07	1569
5	uni+dep+pos	575,200	76.6	19984
10	uni+dep+pos	106,561	76.22	6890

Table 2.8: Comparison of computation time of the GP algorithm and accuracy as annotation graph complexity increases with annotations and minimum frequency threshold decreases. Accuracy reported is for the best parameter setting in Table 2.4 ($\beta = \frac{1}{6}$). The computation time is reported in seconds.

GP, as the annotation graph complexity increases and the minimum support threshold decreases. Table 2.8 compares the computation time for GP when we lower the minimum support threshold from 10 to 5, and when we add POS annotations to unigram and dependency annotations. As expected, the performance improves as we add more annotations and decrease the minimum support threshold. However, the computation time for GP also increases substantially with an increase in the complexity of the annotation graphs, and decrease in the minimum support threshold.

To conclude, from the performance analysis presented here, we see that as we lower the minimum frequency threshold, a lot of subgraph features are generated and performance improves, but it also increases the computation cost. The computation bottleneck is in finding these features in the training dataset, matching the discovered patterns with the test set, GP based feature construction on the large feature space and learning with SVMs on a large feature set (when not using GP based feature construction). A future direction is to find ways to reduce the computation cost for learning with annotation graphs. To reduce the search space for GP, we can use one of the feature selection methods discussed in Section 2.3 to reduce the total number of features, and use only the selected subset of features in GP. Feature selection can also be incorporated into the frequent subgraph mining algorithm (gSpan) to find the discriminative frequent subgraphs directly (Thoma et al., 2009; Tsuda, 2007). Another future direction is to use these extensions of gSpan to find frequent and discriminative subgraph features.

2.5 Related Work

Supervised annotation learning methods (for illustration in Relation Extraction[30]) can be broadly divided into two categories (Bach and Badaskar, 2007): 1) Feature based methods, for example,

[30]Relation extraction is often modeled as a binary classification task of determining whether two given entities participate in a given relation.

(Kambhatla, 2004; GuoDong et al., 2005) and 2) Kernel based methods, for example, (Bunescu and Mooney, 2005a; Culotta and Sorensen, 2004; Nguyen et al., 2009). In feature based methods, features are pre-defined, based on the prior knowledge, through manual inspection of the data or extensive experimentation. In kernel based methods, exact features are not defined, only a similarity score is computed between a pair of instances. Feature based methods are computationally simple but require the manual effort of defining the features. Kernel based methods do not require defining the features explicitly, and compute the similarity between instances implicitly over a much larger feature space. Feature based methods provide us patterns (as the top ranked features) commonly used to express the target annotation in text, and hence give us more insight into the problem. Kernel based methods are abstract and do not tell us much about the patterns associated with a classification task. Our approach is a feature based approach, however our features are automatically derived, thus we get the benefit of the expressive power of the feature based methods, without the need for extensive manual feature engineering. While the feature based approach is generic, kernel based approach can only be used with specific learning algorithms, such as SVMs.

For a kernel based learning algorithm, such as SVMs, the label of a new instance is determined based on its similarity to the labeled training instances, calculated using a kernel function. An instance may be represented as a vector of features, where features may be words in the text, or structured features defined using prior annotations. A simple kernel function is the linear kernel, that computes the similarity as the dot product of the instance feature vectors. For binary occurrence features, this kernel function returns the number of features in common between the instances. In our approach, an annotation graph is constructed for each instance, and frequent subgraphs in a set of annotation graphs, are used to represent an instance. A linear kernel computes the similarity between two instances as the number of common subgraphs between their annotation graphs. Instead of considering all subgraphs, we only use subgraphs that are frequent in a set of graphs, similar to a bag-of-words representation, where often only frequent words are used, since they are expected to generalize to new unseen instances.

An alternative kernel based approach may compute the similarity directly over the annotation graphs for the instances. Many kernels have been studied for computing the similarity between structured representation for instances. A convolution kernel (Haussler, 1999) defines the similarity between two structures as the sum of similarity between their substructures. String kernels (Lodhi et al., 2000) compute the similarity between two strings as the number of common subsequences between the two strings. Similarly, tree kernels compute the similarity between two trees, such as shallow parse trees or dependency trees, as the number of common subtrees (Vishwanathan and Smola, 2002), subset trees (Collins and Duffy, 2002) or partial trees (Moschitti, 2006). Nguyen et al. (2009) combine several convolution kernels defined for constituent trees, dependency trees, and sequential structures, for a relation extraction task. Airola et al. (2008) propose an all-path kernel for graphs constructed from the dependency graphs and word sequences, for a protein-protein interaction extraction task. The all-path kernel computes the similarity between two instances as the number of common weighted paths between a protein pair to be classified in the graphs. However, not all interesting patterns can be captured as paths in the graphs. Substructures, such as sub-trees or sub-graphs, are often used (Culotta and Sorensen, 2004; Zelenko et al., 2003). More powerful graph kernels, based on walks (Kashima et al., 2003), limited size subgraphs (Shervashidze et al., 2009), subtree patterns (Shervashidze and Borgwardt, 2009) have also been proposed in the literature.

A limitation of the kernel based methods discussed above is that the kernel function is computed for two instances in isolation, that is without evaluating these substructures over other training instances available. In our feature based approach, we too count the common sub-structures between structures for instances, when computing the similarity using a linear kernel. However, we only consider the frequent (and discriminative) sub-structures, that are expected to generalize to new instances and are also relevant for the classification task. Kernel based methods on the other hand have an advantage of being computationally less expensive, since at a time only two structures are compared, whereas feature based methods require analyzing the graphs for all training instances together, and hence are computationally more expensive.

In this work, we use an efficient subgraph mining algorithm (gSpan (Yan and Han, 2002a)) to find frequent subgraphs in the annotation graphs, and use these as features in our model. gSpan and its several extensions (Thoma et al., 2009; Tsuda, 2007) have been used in the literature for defining features for classifying graph structures. Most of these applications have focussed on classifying chemical compound structures (Thoma et al., 2009; Yan and Han, 2002a; Tsuda, 2007), protein structures (Thoma et al., 2009) or XML document structures (Tsuda, 2007). In this work, we use subgraph features derived from the annotation graphs for learning new annotations, a substantially different task and a novel approach to annotation learning.

The most closely related work in the literature is by Matsumoto et al. (2005), where they use frequent sub-sequence and sub-tree mining algorithms (Asai et al., 2002; Pei et al., 2004) to derive structured features, such as word sub-sequences and dependency sub-trees automatically. They show that these features outperform bag-of-words features for a sentiment classification task, on a commonly-used movie review dataset (Pang et al., 2002). However, their features are limited to sequences or tree annotations. Often, features that combine several annotations capture interesting characteristics of the text, and have been shown to improve performance (Gamon, 2004; Wilson et al., 2004; Joshi and Rosé, 2009). Combining several annotations together would often result in a graph representation.

In this work, we used gSpan to derive frequent subgraphs from the annotation graphs, which we use as features in a supervised learning algorithm. This approach generates a lot of features, many of which may be redundant, and may not discriminate between the classes of interest. We evaluated several feature selection methods that find a subset of discriminative features from the set of all frequent subgraphs. Instead of an after step, feature selection can also be integrated into the frequent subgraph mining algorithm. Thoma et al. (2009) and Tsuda (2007) propose extensions to gSpan, where they mine frequent *discriminative* subgraphs from a set of graphs, and use those as features in a supervised learning algorithm. Tsuda (2007) modify the gSpan search algorithm to introduce a search tree pruning condition based on LARS feature selection criteria (Efron et al., 2004), for efficient discovery of optimal patterns. Thoma et al. (2009) propose a greedy feature selection approach that integrates a sub-modular quality criterion into the gSpan algorithm for efficient discovery of frequent discriminative subgraphs. Embedding feature selection in the gSpan algorithm reduces the search space considerably. The advantage of applying feature selection after mining frequent subgraphs is that it can be applied to any set of features, not necessarily subgraph features. If we want to add the subgraph features to an existing set of features, to evaluate the gain in performance from the subgraph features, we can apply feature selection after joining all feature sets together, like we did in this work.

In parallel to our work (Arora et al., 2010), Rink et al. (2010) used a similar approach for learning causal relations between event mentions in text. Like in our work, they use a graph representation for sentences to encode the lexical, syntactic, and semantic information; and use frequent subgraphs among graphs for training instances (extracted using gSpan) as features in a binary SVM classifier. They also used an extension to gSpan (Thoma et al., 2009) to find discriminative subgraphs. They show that with automatically extracted subgraph features their approach outperforms the prior work on predicting causal relations between event mentions in text.

Annotation graphs have also been proposed to represent the information need in information retrieval for question answering (Bilotti et al., 2010). Linguistic annotations such as Named Entities, Semantic Roles, etc., are used to represent the information need in the question as an annotation graph. Annotation graphs are also constructed for the candidate passages. An instance (question and passage pair) is represented using subgraph features, such as semantic role patterns, that occur in the annotation graphs for the question and the passage. A rank learning algorithm (Committee Perceptron) is used to learn a passage ranking model. Subgraph features with a maximum size of three relations are automatically extracted. Additionally, they use longer path features between keywords in the annotation graph. A set of rules based on prior knowledge about the linguistic annotations are defined to avoid redundant or sparsely predictive features. An alternative approach based on our work would use the subgraph mining algorithm (gSpan) to find all common subgraphs between the annotation graphs for a question and passage. In order to find features that generalize well, we could also restrict our feature set to frequent subgraphs in the annotation graphs for questions and passages in the dataset.

Most of the work discussed above represents an instance as a vector of subgraph features, extracted using gSpan or one of its extensions, and learns a model using a supervised learning algorithm, such as SVMs. Kudo et al. (2004) propose an alternative approach, where they define a decision stump for each subgraph extracted using gSpan, and use a Boosting algorithm (Schapire and Singer, 2000) with subgraph based decision stumps as weak learners. They also discuss the analogy between their approach and SVMs with convolution kernels. They test their boosting based approach with subgraph features on a cell-phone review classification task and a chemical compound classification task. For the review classification task, they show that for a boosting based approach subgraph features outperform bag-of-words features. They also show that their boosting based approach with subgraph features is significantly better than a bag-of-words kernel and not significantly different from a tree-kernel (Collins and Duffy, 2001; Kashima and Koyanagi, 2002) in SVMs.

Dependency relations capture long-distance relationships between words. Lexical patterns are often sparse, that is, they match only a few instances. In order to find general patterns with dependency relations, in our experiments, we added extra edges between a dependency relation and parts of speech for the head and modifier words (Figure 2.9a). We expect the patterns with parts of speech to match more instances. An alternative method for finding general patterns is to search for patterns with wild cards on one or more nodes. A node with a wild card can match any node in the annotation graph for an instance. Ren and Zhao (2010) provide an extension to gSpan to find frequent subgraph patterns with wild cards (up to a specified number of maximum wild cards in a subgraph). This approach would allow us to find general patterns such as an adjective modifying a noun, as illustrated in Figure 2.9c. However, for a subgraph in an annotation graph to match a pattern with wild cards, all wild card nodes should match with some node in the subgraph. For example, "I truly loved the movie" and "I absolutely loved the movie" match the following pattern with wildcard: "I X loved the movie", where 'X' is a wild card. However, "I for once truly loved the movie" would not match the above pattern. Instead, it would match the following pattern with wild cards "I X X X loved the movie". A regular expression like pattern "I [X]+ loved the movie" would match all three sentences above, where "[X]+" implies 0 or more contiguous wild cards. In a recent work, Gianfortoni et al. (2011) propose an approach to find general sequence patterns that allow for gaps of arbitrary length within a pattern. They call these the 'stretchy patterns'. These patterns can contain a mix of words or categories for words (tags for words such as parts of speech). They show that stretchy patterns provide a significant improvement in performance over unigram and n-gram features, for a gender based text classification task. Similar idea can be applied to subgraph features with wild cards. As a post processing step, all patterns with contiguous wild cards can be collapsed into a single pattern with a regular expression of '[X]+'. The features proposed in (Gianfortoni et al., 2011) are sequence based features, while in our work we mine subgraph features of varying complexity.

Thus, features derived from prior linguistic annotations are crucial for several learning problems. In this chapter, we presented a feature-based approach for automatically extracting features from annotation graphs for instances, defined in terms of the prior linguistic annotations.

2.6 Summary and Future Work

Features that capture the linguistic structure in text have been shown to be useful for several learning tasks (Gildea and Jurafsky, 2000; Pradhan et al., 2004; Wilson et al., 2004; Arora et al., 2009a; Gamon, 2004; Joshi and Rosé, 2009; GuoDong et al., 2005). However, often these features are hand-crafted by the domain experts. In this chapter, we proposed an automatic approach for extracting structured features from available linguistic annotations. With a feature construction technique that uses genetic programming to combine these complex features without the redundancy, our approach outperforms a strong unigram-only baseline for a sentiment classification task. The improvement in performance is small, but given that in sentiment domain, specifically reviews, structured features have shown little improvement if any (Joshi and Rosé, 2009; Arora et al., 2009a), this result is notable. For the protein-protein interaction extraction task, the proposed approach gives a substantial and significant improvement over unigram features, without any feature selection. In the literature, Rink et al. (2010) show that an approach similar to ours outperforms the state of the art work in identifying causal relations between events. So far we have only used dependency parse and parts of speech tags as prior annotations in the annotation graph representation. A future direction is to use other annotators such as semantic role labelers (e.g. ASSERT[31]), named entity taggers (e.g. Stanford NE tagger[32]), and Wordnet[33], etc., to derive richer features.

Automatically extracting features from annotation graphs gives us a very large feature space. Feature selection may play an important role in identifying the important features from this large pool of features. In the work so far, we experimented with several feature selection techniques based on genetic programming, chi-square statistics, correlation, and feature subsumption. Genetic programming gave us the best performance for a sentiment classification task. There is additional refinement that can be performed on the genetic programming fitness function, to improve the quality of our features. Our feature extraction approach is not tied to any specific feature selection technique or to the SVM learning algorithm. Other methods to try are the $L1/L2$ regularization based learning algorithms that embed feature selection in the learning algorithm itself. For example, Joshi et al. (2010) successfully applied the $LARS - EN$ based learning algorithm (Zou and Hastie, 2005) to the sentiment domain, where they used some features derived from prior annotations. In the work presented here, feature selection is decoupled from the subgraph mining algorithm. As mentioned before, there has been some recent work that mines discriminative frequent subgraphs from a dataset of labeled graphs (Tsuda, 2007; Thoma et al., 2009). A future direction is to use these enhancements to gSpan algorithm for finding frequent discriminative subgraphs from the

[31]http://cemantix.org/assert.html
[32]`http://nlp.stanford.edu/software/CRF-NER.shtml`
[33]`http://wordnet.princeton.edu/`

annotation graphs, and use these subgraphs as features.

The performance of structured features derived from prior linguistic annotations (automatically or with the help of a domain expert) is dependent on the accuracy of these linguistic annotators. While many high quality linguistic annotators such as parts of speech, syntactic parsers, etc. are now available, as we move up the annotation hierarchy, the performance of available annotators decreases. For example, ASSERT (Pradhan et al., 2004), a semantic role labeler, trained on WSJ corpus with syntactic parses from the Charniak Parser (Charniak and Johnson, 2005) has an F-score of 73.4 for the joint task of argument identification and classification on a WSJ test set (Row A in Table 15 in (Pradhan et al., 2008))[34]. However, we often would not have gold annotations in the target domain to retrain a semantic role labeler. Pradhan et al. (2008) found that when ASSERT trained on WSJ corpus is applied to the Brown corpus, performance for the joint task of argument identification and classification drops to 59.1 (Row B in Table 15 in (Pradhan et al., 2008)). A direction for future work is to analyze the effect of the quality of prior annotations, from which the structured features are derived, on the performance of the final task.

For frequent subgraph mining algorithm, there are two parameters: 1) minimum support threshold ($minSup$), where support for a subgraph is the number of annotation graphs that contain the subgraph, and 2) maximum subgraph size ($maxSGsize$), based on the number of nodes in a subgraph. In this work, we determined the values for these parameters heuristically. For a new learning task, the optimal values for these parameters should be determined using a grid search based on performance on a development set. For each member in the Cartesian product of possible values for the two parameters, we would train a classifier on the training data and test its performance on the development set. We would then select the combination of values for the two parameters that gives us the best performance on the development set, to build our final model. Instead of evaluating all combinations of values in the grid, we could use a hill-climbing procedure (as also followed in (Zaidan et al., 2007)). For example, we could start with an initial guess for each parameter, say {3,7} for $minSup$ and $maxSGsize$, respectively. We can compare performance on development set for $minSup \in \{2, 3, 4\}$, i.e., we look in either direction of the current value, keeping $maxSGsize$ fixed at 7. If $minSup = 4$ gives better performance than other values, we follow that direction and evaluate performance for $minSup = 5$. We continue this process until we find a value for $minSup$ which is better than both its left and right neighbors. We then fix that value for the $minSup$ parameter, and repeat the process for the $maxSGsize$ parameter. In Section 2.4.3, we showed that for a movie review classification task with GP based feature combination, a lower value for the $minSup$ parameter gives a better performance. However, this improvement in performance comes with a high computation cost. Thus, resource constraints, if any, may also influence our choice for values for these parameters.

To conclude this chapter, we presented a novel approach for extracting features automatically from prior linguistic annotations, in order to reduce the cost of feature engineering. We show that the proposed features improve performance beyond a simple bag-of-words model, for a sentiment classification task and protein-protein interaction extraction task. A similar work in the literature (Rink et al., 2010) has also demonstrated the potential of this approach for identifying causal relationship between event mentions in text.

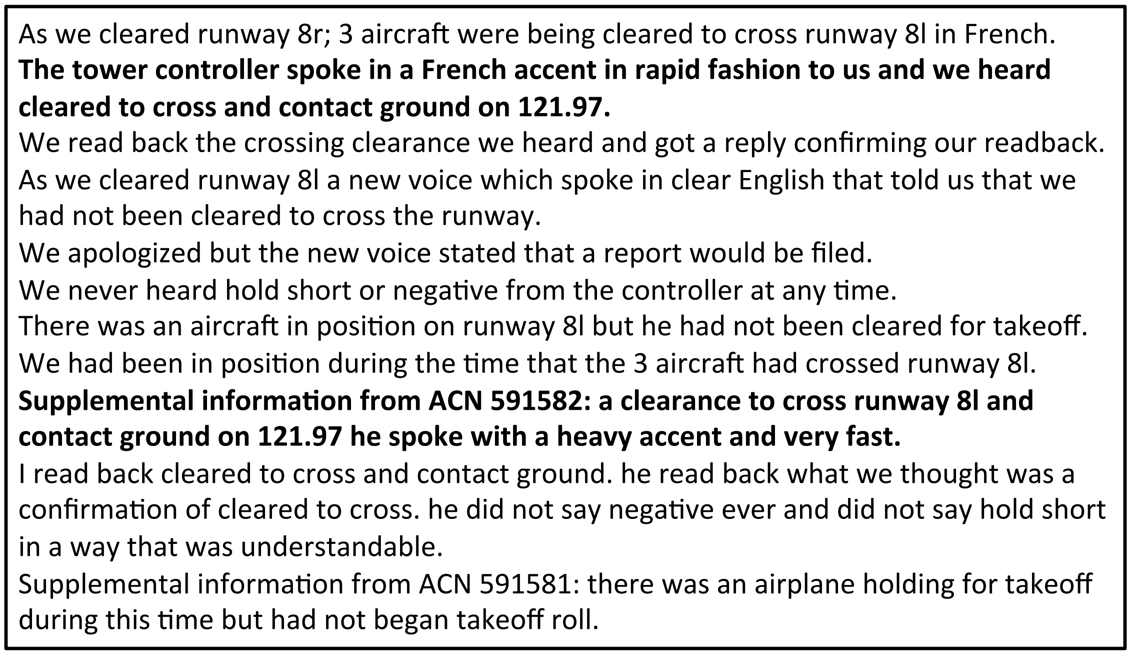
[34]If trained with features from the corrected Treebank parse trees, then the F-score is 80 with automatic parse trees for the test set (Table 4 in (Pradhan et al., 2008)).

Chapter 3

Active Annotation with Indirect Feature Feedback

In supervised learning, an instance is represented by a feature vector, and training a classifier involves learning a function of these features to approximate the target variable. To learn a good approximation, several labeled instances are needed. Labeling an instance may require substantial annotation effort, called the *annotation cost.* In order to reduce the total annotation cost for the desired performance, in addition to labeling instances, the annotator could provide information about the features directly. While the annotator may not be able to provide exact weights for the features, they could identify the relevant features and the class they support. Direct feedback on features in addition to labeling the instances has been shown to reduce the total number of labeled instances required to achieve the desired performance (Godbole et al., 2004; Raghavan and Allan, 2007; Druck et al., 2009; Melville and Sindhwani, 2009). However, such feedback is restricted to simple features, such as words. Often linguistic features, such as dependency triples, parts of speech, semantic roles, etc., are used to represent a text instance (Gamon, 2004; Pradhan et al., 2004; Joshi and Rosé, 2009). For image datasets, features such as pixel and texture values are commonly used (Settles et al., 2008b; Vijayanarasimhan and Grauman, 2011; Donahue and Grauman, 2011). The annotator may not be familiar with such features to give feedback on them. An alternative is for the annotator to provide indirect feedback on features by indicating parts of an instance that are *rationales* for the instance's label.

Different parts of an instance may be strong or weak indicators of the instance's label. For example, a movie review often consists of a plot summary, in addition to the reviewer's sentiment about the movie, which may only be a weak indicator of the reviewer's sentiment. An aviation safety report often consists of a description of the flight, beyond the cause of the aviation problem. Similarly, when classifying an image of a scene, such as a kitchen, parts of the image that show the sink, stove, etc., are strong indicators, while the part that shows the floor may only be a weak indicator. Thus, in this work, we propose an alternate annotation strategy where the annotator indicates parts of an instance that are *rationales*, i.e. key indicators, for the instance's label, for example, sentences in a document, segments in an image or a video, that are key indicators of the instance's label. In order to determine the label for an instance, the annotator perhaps already makes this distinction. We ask them to provide their reason to assign the indicated label to an instance, in form of rationales. Rationales provide indirect feedback on features, since the features that overlap with rationales should be important for the classification task and this indication is only indirect. While direct feature feedback has been modeled as a separate task from labeling instances in most prior work (Godbole et al., 2004; Raghavan and Allan, 2007; Druck et al., 2009; Melville and Sindhwani, 2009), rationales are solicited together with the instance's label. Figure 3.1 shows an example of a rationale sentence in a document.



As we cleared runway 8r; 3 aircraft were being cleared to cross runway 8l in French.
The tower controller spoke in a French accent in rapid fashion to us and we heard cleared to cross and contact ground on 121.97.
We read back the crossing clearance we heard and got a reply confirming our readback.
As we cleared runway 8l a new voice which spoke in clear English that told us that we had not been cleared to cross the runway.
We apologized but the new voice stated that a report would be filed.
We never heard hold short or negative from the controller at any time.
There was an aircraft in position on runway 8l but he had not been cleared for takeoff.
We had been in position during the time that the 3 aircraft had crossed runway 8l.
Supplemental information from ACN 591582: a clearance to cross runway 8l and contact ground on 121.97 he spoke with a heavy accent and very fast.
I read back cleared to cross and contact ground. he read back what we thought was a confirmation of cleared to cross. he did not say negative ever and did not say hold short in a way that was understandable.
Supplemental information from ACN 591581: there was an airplane holding for takeoff during this time but had not began takeoff roll.

Figure 3.1: An example of rationale sentences (in bold) in an aviation safety report (Abedin et al., 2011) with incident cause factor as 'Communication Environment'.

Zaidan et al. (2007) define *rationales* as spans of text in a document that are key indicators of its class label. In this work, we segment an instance into sub-instances, and the annotator identifies the sub-instances that are rationales. To annotate a span of text as rationale, the annotator must perform the cognitive task of identifying the appropriate boundaries for the rationale, which may require more time than voting on a pre-segmented span of text in a document. Also, highlighting a span should require more user interface time than voting on a pre-highlighted text segment. Additionally, exact spans for rationales may vary across annotators, while we can expect to see more agreement between the annotators when they vote on sub-instances. In this work, we consider two annotation strategies where the annotator provides : 1) instance's label only (LO), and 2) instance's label, together with rationales in support of the instance's label (LR). Rationales provide additional information per instance, but they may come with an additional cost. In prior work, the additional cost for annotating rationales (as spans of text) is not accounted for, and the two annotation strategies are compared in terms of the performance for a given number of instances (Zaidan et al., 2007; Arora and Nyberg, 2009). The additional cost for rationales may vary with the annotators, instances, annotation tasks, user interface design, etc. We compare the two strategies for different additional costs for annotating rationales. For a sentiment classification task and an aviation incident cause identification task, we show that rationales (as votes on sub-instances in an instance) can provide better performance for a given annotation cost, when the additional cost for annotating them is small.

Each instance (with or without rationales) may provide different incremental value to the learning algorithm. Annotation cost may also vary across instances and annotation strategies. We propose a cost-sensitive active learning approach, where in each iteration an instance is actively selected for a given strategy. We show that this strategy sensitive instance selection approach for seeking rationales performs better than seeking rationales for instances selected randomly or actively, independent of the strategy. When the cost for annotating rationales is high, rationales may not be beneficial for all instances, and we may want to selectively ask for rationales. We further extend the proposed approach to jointly select the best instance and strategy in each iteration. The best fixed strategy (*LO* or *LR*) varies with the additional cost for rationales. For different costs for rationales, we show that the proposed approach follows the best fixed strategy, and in some cases it is better than both strategies. That is, selectively asking for rationales performs better than rationales for all instances or rationales for none. We evaluate several models for incorporating cost in the active selection criteria. We show that the proposed approach outperforms cost-sensitive and cost-insensitive instance selection for a fixed strategy, and cost-insensitive joint selection of instance and strategy.

For the rest of the chapter, we first discuss different methods proposed in the literature for seeking feedback on features. Next, we discuss methods for incorporating information about rationales in learning. After this, we present our cost-sensitive active learning approach. We then present our experiments and results, followed by the related work, conclusions and suggestions for the future work. The work presented here is an exposition of the work presented in (Arora et al., 2012), with some additional experiments and analysis.

3.1 Seeking Feedback on Features

Different methods to seek feedback on features from the annotator can be broadly divided into two categories: *direct* and *indirect* feedback. Seeking direct feedback on features may involve showing the annotator a list of features, and asking them to identify the relevant features among them, and indicate the class they support. Direct supervision for simple features has been explored in the literature. Raghavan et al. (2006) seek feedback from the annotator on unigram, bigram and trigram features for document classification tasks. Druck et al. (2008) and Settles (2011) seek feedback on word features for document classification tasks. Druck et al. (2009) seek feedback on words, regular expressions, etc., for a sequence labeling task. Settles (2011) also allows the annotator to specify relevant words in a text box, whether or not they are currently in the model's vocabulary.

Among the set of all features, a decision needs to be made about what features to present to the annotator. Raghavan et al. (2006) rank features by information gain[1] on labeled data, select the top 20 features, mix them randomly with features much lower in the list, and show one feature at a time to the annotator. The annotator indicates whether a given feature is relevant or not. A feature is relevant if it helps to discriminate between the target classes. Druck et al. (2008) use topic models to select the most representative features to present to the annotator for feedback. An alternative to this passive approach to pre-select the features for feedback is to selectively sample the features in an iterative procedure (active learning), where in each iteration the features expected to benefit the current model the most are presented to the annotator. Godbole et al. (2004) select features for feedback in each iteration as those expected to reduce the current model's uncertainty on unlabeled data the most. Raghavan and Allan (2007) select features based on their co-occurance with the top ranked features from the current model. Druck et al. (2009) evaluate features in terms of the model's uncertainty about them, calculated as the sum of the marginal entropies at the positions where the feature occurs, scaled by the frequency (log count) of the candidate feature in

[1]`http://nlp.stanford.edu/IR-book/html/htmledition/mutual-information-1.html`

the corpus. Settles (2011) organizes the features by the class label, and ranks them by information gain computed using the labeled data and probabilistically-labeled unlabeled data to reflect the model's beliefs about the features. To organize features into classes, a feature is posed in a class with which it occurs at least 75% as often.

When asking for feature feedback, we may only ask the annotator to differentiate between relevant and non-relevant features as in (Raghavan et al., 2006), or the annotator may also be asked to provide class association for the features as in (Raghavan and Allan, 2007) and (Druck et al., 2008).

An important question to answer about feature feedback from the annotators is whether they are able to provide useful feedback on features. In other words, are they able to identify features that are relevant for the learning task. A large amount of labeled data (often referred to as 'oracle') can be used to determine what features are relevant by using class association scores, such as information gain. Raghavan et al. (2006) and Raghavan and Allan (2007) compare feature feedback from the annotators with that obtained from the oracle. Raghavan et al. (2006) evaluate the precision and recall of the features labeled by annotators against the top 20 features as ranked by the oracle. On average (over a few document classification tasks), the annotator scores a precision of 0.58 and a recall of 0.65. Druck et al. (2008), in their user experiments to label 100 features for a binary classification task, observed that on average the user labeled features have a precision of 0.62 and a recall of 0.96, when compared against the oracle. The user labeled features that did not match the oracle labeled features were also found to be moderately relevant when inspected manually.

Direct feature feedback methods discussed so far ask the annotators to identify globally relevant features, but certain features are difficult to vote on without the context of an instance, and may take on very different meanings in different contexts. Druck et al. (2009) use a grid interface to seek feedback on features in context. Features that appear in similar contexts are grouped together and presented in a box. This interface is expected to be useful, as features in a group are expected to receive the same label from the annotator. Features in a box are sorted by model's uncertainty and the annotator labels a single feature at a time. The annotator is shown model's current belief about the features, and occurrences of the selected features in context.

Direct feature feedback methods discussed above are limited to simple features like unigrams. Complex linguistic features are often difficult to visualize, and the annotator may not be familiar with such features to give feedback on them. In image datasets, often features such as pixel values, color and texture values, etc. are used, the annotator may not be able to comprehend such features, and give feedback on them. Also, in most of the work discussed above, feature feedback is a separate task from labeling instances. An alternative approach is to seek indirect feedback on features by asking the annotator to highlight spans of text as *rationales* (Zaidan et al., 2007) for the instance's label. This task is not separate from the original task of labeling an instance. In addition to providing the instance's label, the annotator provides rationales for the indicated label. The annotator perhaps already makes this distinction in order to determine the label for an instance. In order to leverage from the annotator's expertise about the task, it is perhaps better to have them identify the parts of an instance that are meaningful to them, rather than having them vote on features chosen by the system. Rationales provide indirect feature feedback, since features that overlap with the rationales should be important for the classification task, and this indication is only indirect. As we will see in the next chapter, information about rationales is used to indirectly influence the weights that are learnt for these features.

Figure 3.2 shows an example of a rationale for a movie review document, from the data shared by Zaidan et al. (2007). The annotation task here is to label a movie review as expressing positive or negative sentiment towards the movie. The annotator additionally provides rationales for their decision to label a review as positive or negative. Abedin et al. (2011) adopt a similar approach to seek rationales for aviation safety reports. The annotation task here is to label a report with the causes (from a list of 14 classes) of the aviation problem. Additionally, the annotator highlights spans of text as rationales for the instance's label. Figure 3.3 shows an example of a rationale for an aviation safety report, from the data shared by Abedin et al. (2011). Rationales are not limited to text instances. Donahue and Grauman (2011) adopt a similar approach for a scene classification task. In addition to labeling a image with one of the 15 scene categories, the annotator uses a polygon drawing tool to highlight the regions most indicative of their label for the image. For example, in an image of a kitchen, regions that correspond to a sink and fridge are strong indicators of the kitchen. Figure 3.4 shows an example from (Donahue and Grauman, 2011) of an image of a scene labeled with rationales.

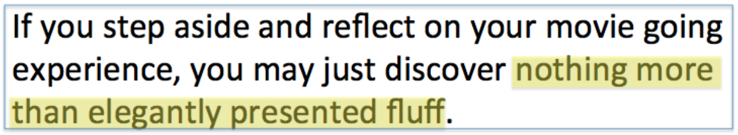

If you step aside and reflect on your movie going experience, you may just discover nothing more than elegantly presented fluff.

Figure 3.2: A sentence from a movie review, with a rationale highlighted, from the data shared by Zaidan et al. (2007), where the annotation task is to label a review as expressing positive or negative opinion towards the movie, and highlight spans of text as rationales.

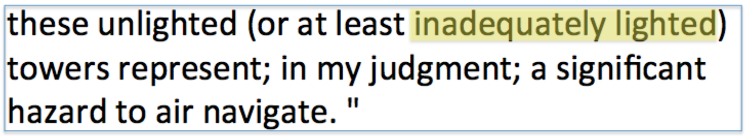

these unlighted (or at least inadequately lighted) towers represent; in my judgment; a significant hazard to air navigate. "

Figure 3.3: A sentence from Aviation Safety Reports data (Abedin et al., 2011), with a rationale highlighted. The annotation task is to label a document with causes of aviation safety problem, and highlight spans of text as rationales. In this example, one of the cause factors is *Resource Deficiency*, and the corresponding rationale string is highlighted.

Rationales, as defined by Zaidan et al. (2007) (spans of text in a document), leave the annotator with a lot of flexibility on how they may annotate rationales. This raises a concern that rationales may be specific to an annotator. In order to verify the consistency of rationales across annotators, Zaidan et al. (2007) calculate the agreement between rationales from four annotators, on a subset of the data (about 100 documents with equal distribution of the two classes). A rationale from annotator A_i is considered to have been also annotated by A_j, if at least one of A_j's rationales overlaps with it. They calculate pairwise annotator agreement between all annotators. The percentage of rationales for a given annotator that were also annotated by another annotator was found to be between $75 - 91\%$ for four annotators. This suggests that there is a fair amount of agreement between the annotators about the rationales. However, Zaidan et al. (2007) did not evaluate the extent of overlap between rationales that match across annotators. Since the entire span of text annotated as rationale is used to incorporate information about rationales, different spans for rationales may give us a different result. In this thesis, we propose a modification to

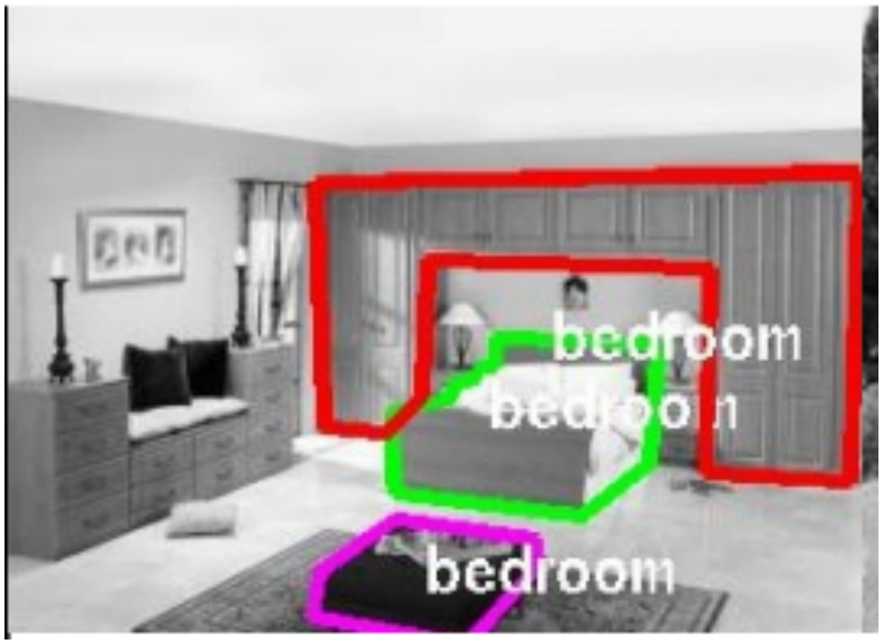


Figure 3.4: An image from Scene classification dataset (Donahue and Grauman (2011)), with few rationales marked in color. The annotation task is to label an image with one of the 15 scene categories (bedroom in this case), and use a polygon drawing tool to mark the rationales.

Zaidan et al. (2007)'s definition of rationales. We segment an instance into sub-instances, and the annotator indicates the sub-instances that are rationales. For example, a document may be segmented into sentences or phrases (using a syntactic parse tree), and the annotator indicates the sentences/phrases in a document that are rationales for the document's label. There are three main advantages of this approach :

1. Voting on a pre-highlighted text segment should require less user interface time than highlighting a span of text. Additionally, the annotator would require some extra time to decide the appropriate span for a rationale.

2. Rationales from different annotators would have the same span, unlike when they decide the span for the rationale themselves.

3. In this thesis, we propose a cost-sensitive active learning approach that jointly selects the instance and strategy in each iteration. As we present later, in order to estimate the benefit to the model from rationales, we need to calculate the expected risk over possible rationales. If rationales can be arbitrary spans of text, the set of possible rationales is very large, and it is computationally infeasible to consider all possible spans for rationales. However, when rationales are constrained to be sub-instances in an instance, the set of possible choices are relatively much smaller.

In this thesis, we only study rationales (as sub-instances in an instance) for text classification tasks. However, the ideas are general and can be applied to other non-text classification tasks. For example, an image can be segmented into regions, and a video can be segmented into frames; and the annotator indicates the regions/frames that are rationales. As mentioned before, Donahue and Grauman (2011) seek rationales as sub-regions in an image. The annotator uses a polygon drawing tool to highlight the regions most indicative of their label for the image. The exact regions for rationales annotated this way may vary across annotators. As in text classification, highlighting a region in an image would require more user interface time than voting on a pre-segmented region. Image segmentation algorithms (e.g. (Shi and Malik, 2000; Estrada et al., 2004)) can be used to segment an image into regions. The annotator then identifies regions that are rationales for the image's label. Similarly, a video can be segmented into spatio-temporal regions (e.g. (Grundmann et al., 2010)) and the annotator indicates the regions that are rationales for the video's label.

Rationales as sub-instances in an instance are less precise than spans of text. However, in a recent work, Yessenalina et al. (2010) found that extending rationale sub-strings to sentence boundaries did not have a significant effect on performance, for a movie review dataset (Zaidan et al., 2007). In this work, we evaluate whether rationales as sub-instances in an instance are useful for four text classification tasks in two datasets, and propose a cost-sensitive active learning approach to selectively ask for rationales. In the next section, we describe how information about rationales is incorporated into learning.

3.2 Incorporating Feedback on Features

We use SVMs to learn a binary classifier ($\mathcal{L} = \{+1, -1\}$), and adopt the approach in (Zaidan et al., 2007) for incorporating information about rationales in learning. Their approach has also been adopted in other work with rationales (Abedin et al., 2011; Donahue and Grauman, 2011). Let X_i be an instance with N_i sub-instances ($X_i = \{x_{i,1}, x_{i,2}, ..., x_{i,N_i}\}$), and $Y_i \in \mathcal{L}$ be its label. A sub-instance ($x_{i,j}$) may be a rationale for the instance's label ($y_{i,j} = R$) or not ($y_{i,j} = NR$). As discussed in (Zaidan et al., 2007), for each rationale ($x_{i,j}$), a *contrast instance* ($c_{i,j}$) is created, which is the original instance (X_i) with the rationale masked. In addition to the standard constraints in a linear hard-margin SVM ($w \cdot X_i \cdot Y_i \geq 1$), *contrast constraints* ($w \cdot X_i \cdot Y_i - w \cdot c_{i,j} \cdot Y_i \geq \mu$) are added to the model based on the intuition that the correct model should be less sure of its classification of the contrast instance, than its classification of the original instance, since it lacks the important information present in the rationale. In other words, the contrast instance should be closer to the decision hyperplane than the original instance. $\mu \geq 0$ controls the size of the desired margin between the original and the contrast instance. For a soft-margin linear SVM, these constraints are modified as follows :

$$\forall i, j \quad w \cdot (X_i - c_{i,j}) \cdot Y_i \geq \mu(1 - \xi_{i,j}) \tag{3.1}$$

where $\xi_{i,j}$ is the slack variable associated with the contrast constraints. A different C parameter ($C_{contrast}$) is used to control for the importance of satisfying the contrast constraints. Dividing both sides of Equation 3.1 by μ, we get :

$$\forall i, j \quad w \cdot (r_{i,j}) \cdot Y_i \geq 1 - \xi_{i,j} \tag{3.2}$$

where $r_{i,j} = \frac{X_i - c_{i,j}}{\mu}$ is a *pseudo instance*. Since Equation 3.2 takes the same form as the soft-margin SVM constraint for instance X_i ($w \cdot X_i \cdot Y_i \geq 1 - \xi_i$), we can simply add the pair ($r_{i,j}, Y_i$) to the training set, weighted by $C_{contrast}$ instead of C. To allow a biased hyperplane, each instance feature vector (original or contrast) is prepended with a 1, as in (Zaidan et al., 2007). This means, however, that we must prepend a 0 element to the feature vector for each pseudo-instance ($r_{i,j}$), since $r_{i,j} = \frac{X_i - c_{i,j}}{\mu}$.

If binary occurrence features are used, a contrast instance defined by masking a rationale in the original instance (as in (Zaidan et al., 2007)), will only differ from the original instance in features that are present in the rationale, but are not present in another rationale or anywhere else in the document. We refer to this approach as *CIMask*. This means that features that occur in a given rationale, and are also present in another rationale, or somewhere else in the document, will also be present in the contrast instance. If rationales share a lot of features, contrast instances may not be much different from the original instance. An alternative is to define the contrast instance such that only features present in the document, but not present in the corresponding rationale have value 1. That is, $c_{i,j} = X_i - x_{i,j}$, and hence $r_{i,j} = \frac{(X_i - c_{i,j})}{\mu} = \frac{x_{i,j}}{\mu}$, where $x_{i,j}$ is one of the rationale sub-instance in instance X_i. We refer to this approach as *CISub*. In this work, we compare both these approaches. Note that for numeric features, such as feature counts in text and pixel values in images, both these approaches are equivalent.

In a follow up work, Zaidan and Eisner (2008) propose an alternate approach for incorporating information about rationales in learning. In addition to the sentiment classification model for the reviews, they introduce a rationale model to explain the annotator's behavior when he/she annotates the rationales. The parameters for the classification model and the rationale model are chosen to maximize the likelihood of the observed classifications and the rationales. They categorize the rationale model as a noisy channel model, where a 'channel model' verifies that rationales faithfully signal the features that strongly support the classification, and a 'language model' ensures that rationales are well-formed. A log-linear model is used for classifying the documents, and a Conditional Random Fields (CRFs) is used for modeling the rationales. The emission features in CRFs capture the channel model, and the transition features capture the language model. However, the generative approach in (Zaidan and Eisner, 2008) did not give a substantial improvement[2] over the contrast instance approach in (Zaidan et al., 2007). Since SVMs are known to perform well for text classification, we adopt their contrast instance approach (Zaidan et al., 2007).

In addition to the contrast instances, Abedin et al. (2011) use rationales to construct *residue* instances. A residue instance contains only the features extracted from the rationale. Abedin et al. (2011) suggest that a residue instance contains less relevant information for classification than the original instance, and hence the SVM model should be less sure of its classification of a residue instance than its classification of the original instance. If there are several rationales in an instance, a residue instance would contain only part of the information indicative of the instance's label, and hence the model should be less sure of its classification of the residue instance than its classification of the original instance. Residue constraints capture this intuition. Abedin et al. (2011) found that adding residue constraints to the model improves performance on their dataset. Abedin et al. (2011) report overall performance (for all classes combined). For some of their classes, we found that there are on average less than 2 rationale sub-instances per instance (Table 3.2 in Section 3.4.1). Residue constraints may not be as helpful for these classes.

Another alternative to construct contrast instances is to construct a single contrast instance, by masking out all rationales at once. Contrast instance created this way would lack most of the relevant information and would be substantially different from the original instance. In this thesis, we only experiment with incorporating information about rationales through contrast constraints (as in (Zaidan et al., 2007)) with two variants for constructing contrast instances : $CIMask$ and $CISub$. We focus on evaluating rationales as sub-instances in an instance, at different additional costs for annotating rationales. We also propose and evaluate a cost-sensitive active learning approach for joint selection of instance and strategy.

[2]About 1-2 points improvement at few points in the learning curve.

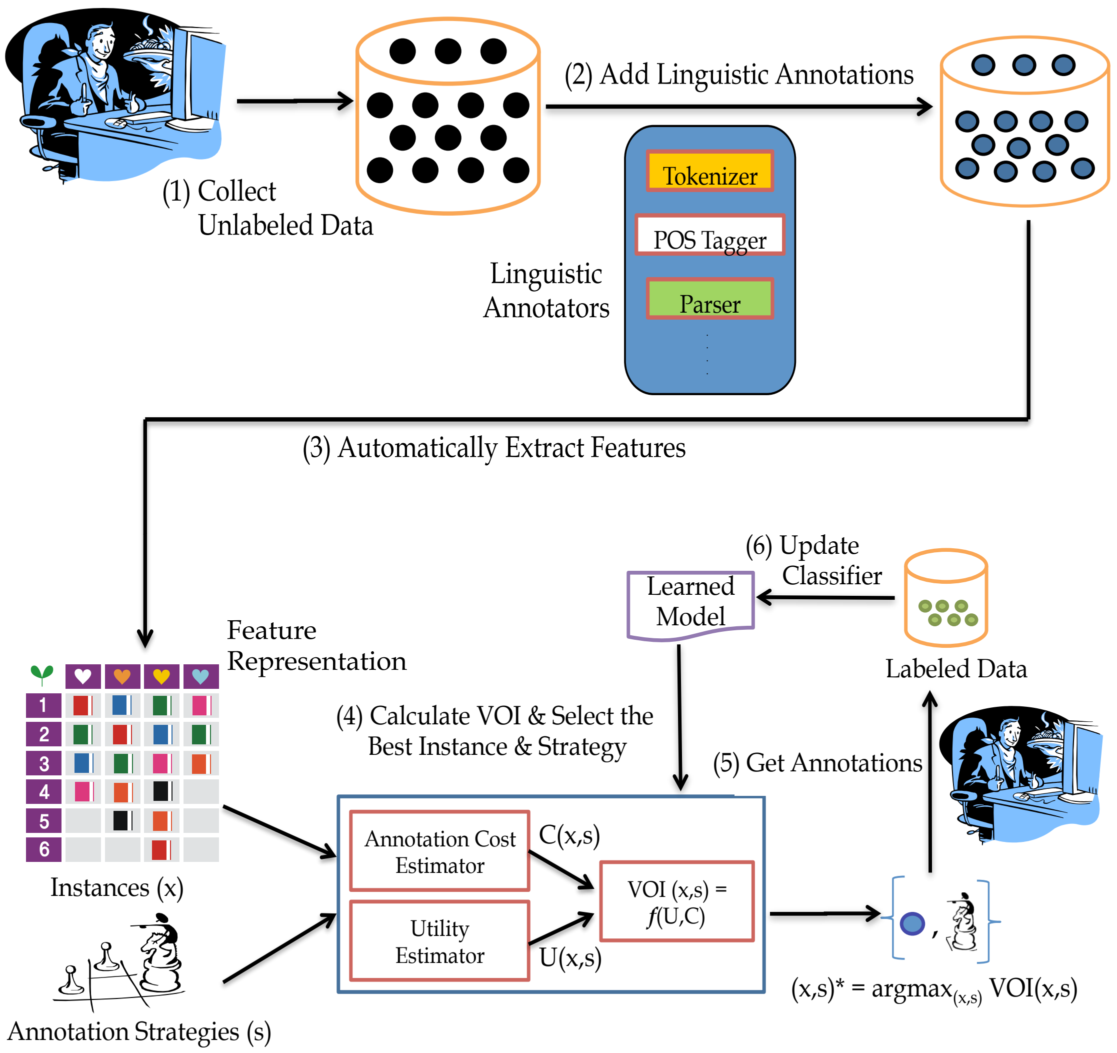

(1) Collect Unlabeled Data
(2) Add Linguistic Annotations
Linguistic Annotators
Tokenizer
POS Tagger
Parser
(3) Automatically Extract Features
Feature Representation
Instances (x)
Annotation Strategies (s)
(4) Calculate VOI & Select the Best Instance & Strategy
Annotation Cost Estimator
Utility Estimator
C(x,s)
U(x,s)
VOI (x,s) = f(U,C)
Learned Model
(6) Update Classifier
Labeled Data
(5) Get Annotations
(x,s)* = argmax(x,s) VOI(x,s)

Figure 3.5: Cost-sensitive active learning in interactive annotation learning framework.

3.3 Cost-Sensitive Active Learning

Each instance with or without rationales may provide different incremental value to the model. Annotation cost may also vary across instances and strategies. Thus, we propose a cost-sensitive active selection criteria to select an instance for the given strategy. When rationales are very expensive to annotate, we may want to selectively ask for rationales. The proposed approach can also be used to jointly select an instance and annotation strategy (LO or LR), in each iteration. Figure 3.5 presents the details of the cost-sensitive active learner in interactive annotation learning framework proposed in this thesis. Each candidate (instance and strategy pair) is assigned a score, called the Value of Information (VOI). In each iteration, the instance and strategy pair with maximum VOI is selected for next annotation. Steps 4 to 6 are repeated until the desired performance is reached or we run out of budget. We evaluate several models for calculating VOI for a candidate.

Let $\mathcal{X}_u^t$ and $\mathcal{X}_l^t$ be the pool of unlabeled and labeled instances, at time t. We define the Value of Information (VOI) for a candidate instance $X_i \in \mathcal{X}_u^t$, and a candidate strategy $s \in S = \{LO, LR\}$, as a function of its expected *utility* ($\hat{\mathcal{U}}(X_i, s)$), and expected annotation *cost* ($\hat{\mathcal{C}}(X_i, s)$):,

$$VOI(X_i \in \mathcal{X}_u^t, s \in S) = f(\hat{\mathcal{U}}(X_i, s),\ \hat{\mathcal{C}}((X_i, s)) \tag{3.3}$$

The first approach we explore for defining VOI is similar to the decision theoretic approach proposed in (Kapoor et al., 2007). This approach was used recently in a setting similar to ours, for actively selecting among different annotation types (with different costs) in multiple instance learning (Vijayanarasimhan and Grauman, 2011). The value of information is defined as the reduction in the total expected cost of the use of the classifier minus the cost of obtaining the candidate annotation. We pay a cost for misclassifying an instance every time the classifier is used. We calculate the expected cost of the use of the classifier (called the *misclassification risk*) on the remaining unlabeled data. Kapoor et al. (2007) calculate the misclassification risk on both labeled and unlabeled data. Since our goal is to select an instance that improves the model's performance on an unseen test instance, we only consider the model's risk on unlabeled data. Thus, VOI is defined as follows:

$$VOI\text{-}RTC(X_i \in \mathcal{X}_u^t, s \in S) = \hat{\mathcal{RR}}(X_i, s) -\ \hat{\mathcal{C}}(X_i, s) \tag{3.4}$$

where $\hat{\mathcal{RR}}(X_i, s)$ is the expected reduction in model's total misclassification risk after obtaining annotation for instance X_i labeled with strategy s, and $\hat{\mathcal{C}}(X_i, s)$ is the cost of annotating instance X_i with strategy s. Model's total misclassification risk at any given point is the expected cost to label the remaining unlabeled data. The VOI approach then evaluates by how much a candidate $\underline{r}$educes this $\underline{t}$otal $\underline{c}$ost (RTC). Reduction in total risk ($\hat{\mathcal{RR}}(X_i, s)$) is calculated as follows:

$$\hat{\mathcal{RR}}(X_i, s) = (\hat{\mathcal{R}}_{\mathcal{X}_l^t}(\mathcal{X}_u^t) - \hat{\mathcal{R}}_{\mathcal{X}_l^{t+1}((X_i,Y_i),s)}(\mathcal{X}_u^{t+1}(X_i)) \tag{3.5}$$

where $\hat{\mathcal{R}}_{\mathcal{X}_l^t}(\mathcal{X}_u^t)$ is the model's total misclassification risk at time t. It is the expected risk on the unlabeled data ($\mathcal{X}_u^t$), of the model trained on labeled data available at time t ($\mathcal{X}_l^t$), calculated as follows:

$$\hat{\mathcal{R}}_{\mathcal{X}_l^t}(\mathcal{X}_u^t) = \sum_{X_k \in \mathcal{X}_u^t} \sum_{h \in \mathcal{L}} r_h (1 - p(Y_k = h|X_k))\bar{p}(Y_k = h|X_k) \tag{3.6}$$

where $\bar{p}(Y_k = h|X_k)$ is the true probability that instance X_k has label $h \in \mathcal{L}$. $(1 - p(Y_k = h|X_k))$ is the probability that the model will not assign this label to the instance, and r_h is the cost of misclassifying an instance with label h. We assume an equal penalty for misclassifying all instances, and set r_h equal to the average cost of labeling an instance without rationales[3]. Since $\bar{p}(Y_k = h|X_k)$ is unknown, we use an estimate for it from the current classifier (i.e. $p(Y_k = h|X_k)$)), by fitting a sigmoid function to SVM's output. Next, we estimate the model's risk after learning on instance X_i annotated with strategy s (i.e. $\hat{\mathcal{R}}_{\mathcal{X}_l^{t+1}((X_i,Y_i),s)}(\mathcal{X}_u^{t+1}(X_i))$ in Eq. 3.5). Since we don't know the true annotation for the candidate instance and strategy, we compute an expected value of this risk. For LO strategy, this expectation is calculated as follows:

[3]For some tasks, penalty may be different for misclassifying instances of different classes, and r_h can be set accordingly.

$$E[\hat{\mathcal{R}}_{\mathcal{X}_l^{t+1}((X_i,Y_i),LO)}(\mathcal{X}_u^{t+1}(X_i))] = \sum_{b\in\mathcal{L}} p(Y_i = b|X_i)\hat{\mathcal{R}}_{\mathcal{X}_l^{t+1}((X_i,b),LO)}(\mathcal{X}_u^{t+1}(X_i)) \tag{3.7}$$

where $\mathcal{X}_u^{t+1}(X_i) = \mathcal{X}_u^t \backslash X_i$, $\mathcal{X}_l^{t+1}((X_i,b),LO)$= $\mathcal{X}_l^t \cup (X_i,b)$, and $\hat{\mathcal{R}}_{\mathcal{X}_l^{t+1}((X_i,b),LO)}(\mathcal{X}_u^{t+1}(X_i))$ is calculated as in Equation 3.6. For LR strategy, to calculate the expected risk, we would need to take an expectation over all possible rationale label sets for sub-instances in an instance. For an instance with N_i sub-instances, there are 2^{N_i} possible label sets. Computing the risk for all label sets can be computationally very expensive. Instead, we compute an approximation of the expected risk. For each class, from the data annotated so far, we train a classifier to predict what sub-instances in an instance are rationales. We use this classifier's prediction for what sub-instances in a candidate instance are rationales, and use Equation 3.7 to calculate an approximation of the expected risk. We set $\mathcal{X}_l^{t+1}((X_i,b),LR) = \mathcal{X}_l^t \cup (X_i,b) \cup((\hat{r}_{i,1},b)...(\hat{r}_{i,\hat{R}_i},b))$, where $(\hat{r}_{i,1},...,\hat{r}_{i,\hat{R}_i})$ are pseudo instances for sub-instances predicted to be rationales in instance X_i.

In the above formulation for VOI, we use the model's total risk on unlabeled data as the expected cost to label the remaining unlabeled data. The VOI approach then evaluates by how much does a candidate reduce this cost. However, this means that risk, and hence VOI, will be different for a smaller or a larger unlabeled set, given the same distribution (i.e. same mean and variance) of the reduction in risk scores in the set. The size of the unlabeled set may influence the weight on reduction in risk over cost. An alternative approach is to treat reduction in risk and cost as two independent scores for evaluating the candidates, and standardize the scores (SS) before subtracting them (suggested in (Donmez and Carbonell, 2008b)):

$$VOI\text{-}SS(X_i \in \mathcal{X}_u^t, s \in S) = \hat{\mathcal{RR}}^{\int}(X_i,s) - \hat{\mathcal{C}}^{\int}(X_i,s) \tag{3.8}$$

where $\hat{\mathcal{RR}}^{\int}$ and $\hat{\mathcal{C}}^{\int}$ are the standardized[4] scores. Standardized scores can be compared directly. This allows us to combine any two scores for ranking the candidates. For example, Tomanek and Hahn (2010) combine the ranks for the candidates based on the uncertainty of the model on the candidate instance, and its annotation cost. However, their approach ignores the actual scores. The difference between the scores for candidates with same ranks based on the two criteria, may be quite different. This difference is not taken into account when combining the ranks. Standardized scores capture the relative difference between the scores for the candidates.

Ultimately, our goal is to reduce the error of the model on an unseen test instance, i.e. the generalization error of the model. An alternative VOI formulation is to select a candidate that reduces the model's *expected* risk on an unseen instance (expected risk for short) the most. Assuming a uniform distribution for instances, that is all instances are equally likely, a simple estimate of this risk is the average risk on a large set. We estimate the model's expected risk in each iteration on the remaining unlabeled data. That is, utility, as reduction in model's expected risk, is calculated as follows:

$$\hat{\mathcal{RR}}_e(X_i,s) = \frac{\hat{\mathcal{R}}_{\mathcal{X}_l^t}(\mathcal{X}_u^t)}{N} - \frac{\hat{\mathcal{R}}_{\mathcal{X}_l^{t+1}((X_i,Y_i),s)}(\mathcal{X}_u^{t+1}(X_i)) + 0}{N} \tag{3.9}$$

where $\hat{\mathcal{RR}}_e(X_i,s)$ is the reduction in model's expected risk. $\hat{\mathcal{R}}_{\mathcal{X}_l^t}(\mathcal{X}_u^t)$ and $\hat{\mathcal{R}}_{\mathcal{X}_l^{t+1}((X_i,Y_i),s)}(\mathcal{X}_u^{t+1}(X_i))$ are calculated as described before in Eq. 3.6 and 3.7 respectively. N is the size of the unlabeled set $\mathcal{X}_u^t$. Risk on the candidate instance after training on it is assumed to be 0.

[4]Subtract the sample mean and divide by the sample standard deviation, where the sample is the candidate set.

If we subtract the cost from reduction in expected (or average) risk (as in VOI-RPC), cost will be the dominant factor, and LR strategy may never be selected. VOI computed using standardized scores for cost and reduction in risk (VOI-SS) is the same whether we use total or average risk. A third approach is to calculate VOI as reduction in risk per unit annotation cost (RPC). That is, VOI is calculated as follows:

$$VOI\text{-}RPC(X_i \in \mathcal{X}_u^t, s \in S) = \frac{\hat{\mathcal{RR}}_e(X_i, s)}{\hat{\mathcal{C}}(X_i, s)} \tag{3.10}$$

In other words, here we are evaluating which candidate reduces the expected risk of the model the most, per unit cost. If we were to plot reduction in risk vs. annotation cost in a graph, this method would correspond to selecting the candidate that is expected to result in the largest slope for the next iteration. This approach of defining VOI as utility per unit cost has been used in the active learning literature with different measures of utility (Melville et al., 2005; Donmez and Carbonell, 2008b; Haertel et al., 2008; Tomanek and Hahn, 2010). In Table 3.1, we present a hypothetical example with three candidates. Here, the three approaches described above will select a different candidate.

Cand.	$\hat{\mathcal{RR}}$	$\hat{\mathcal{C}}$	$\hat{\mathcal{RR}}^{\int}$	$\hat{\mathcal{C}}^{\int}$	VOI-RTC	VOI-SS	VOI-RPC
1	21	15	-0.0798	-0.0867	6	**0.0069**	0.467
2	10	5	-0.9577	-0.9538	5	-0.0039	**0.667**
3	35	28	1.0375	1.0405	**7**	-0.0030	0.417
Avg.	*22*	*16*					
Stdev.	*12.530*	*11.533*					

Table 3.1: Toy example for different ways to compute VOI. In bold is the score for instance selected by the given VOI criterion.

To summarize, in this work, we consider the following three methods for computing VOI: 1) Reduction in total cost of the use of the classifier (VOI-RTC), 2) Subtracting standardized scores for cost and reduction in risk (VOI-SS), 3) Reduction in the model's expected risk on an unseen instance per unit cost (VOI-RPC).

We now describe how we estimate the annotation cost for an instance X_i and strategy s ($\hat{\mathcal{C}}(X_i, s)$). Annotation cost may vary across instances (Donmez and Carbonell, 2008b; Ringger et al., 2008; Settles et al., 2008a; Arora et al., 2009b; Baldridge and Palmer, 2009). Longer documents require more time to read and annotate than shorter ones. Ambiguous instances require more time to label than simpler ones. Annotation cost is more when annotating rationales in addition to providing an instance's label. Zaidan et al. (2007), found that it takes twice as much time to annotate a movie review with rationales, in addition to providing the review's label. In most cases, annotation cost is not known *a priori*. Hence, to calculate VOI, we use an estimate for the annotation cost. We use a simple estimate based on an instance's length, which has been found to be a good indicator of the annotation cost (Ringger et al., 2008; Arora et al., 2009b). Arora et al. (2009b)[5] observed a linear correlation between document length and annotation cost, across multiple annotators. Whether providing rationales or not, we expect the annotation cost to vary with the length of an instance.

[5]This work is also presented in Chapter 4.

Thus, we use a simple cost model where annotation cost is a linear function of the length of an instance. Let $L(x)$ be the length of an instance x, and let $CF(s)$ be the cost factor for strategy s. The cost for annotating an instance x with strategy s is given by:

$$\hat{\mathcal{C}}(x, s) = CF(s) * L(x) \tag{3.11}$$

where the factor $CF(s)$ depends on the annotation strategy s. Arora et al. (2009b) also found a significant linear correlation between the number of rationales and annotation cost. Hence, we assume a linear relationship between the costs for the two annotation strategies of LO and LR. That is, $CF(\text{LR}) = CF(\text{LO}) * \alpha$. We set $CF(\text{LR}) = 1$, and hence $CF(\text{LO}) = \frac{1}{\alpha}$. The actual value of $CF(\text{LR})$ should not affect the relative difference between the candidates, and hence their ranking. This is so because risk has a factor for the cost of misclassifying an instance, set equal to the average cost of labeling an instance without rationales. Thus, both reduction in risk and cost in VOI include this factor. We measure an instance's length ($L(x)$) in terms of the number of sub-instances. For the LR strategy, the annotator identifies which sub-instances are rationales. Thus, the annotation cost for LR strategy should be proportional to the number of sub-instances in an instance. Length of a sub-instance and other properties of an instance such as ambiguity etc. may also effect the annotation time. For the current work, we use a simple model based on instance length as the number of sub-instances in an instance.

Based on the VOI selective sampling criterion described above, the algorithm for joint selective sampling of instance and strategy is described in Algorithm 2.

Algorithm 2 Selective Sampling of an Instance and Strategy $\{X_i, s\}$ with a total annotation budget of B

```
Budget spent so far: $b = 0$
**while** $b < B$ **do**
  **for all** $X_i \in \mathcal{X}_u^t, s \in S = \{\text{LO,LR}\}$ **do**
    Calculate VOI for $\{X_i, s\}$ (Eq. 3.3)
  **end for**
  $(X_i^*, s^*) = \arg\max_{X_i \in \mathcal{X}_u^t, s \in S} VOI(X_i, s)$
  **if** $s^* = LR$ **then**
    $\mathcal{X}_l^{t+1} = \mathcal{X}_l^t \cup (X_i^*, Y_i) \cup \sum_j^{R_i} (r_{i,j}, Y_i)$,
  **else**
    $\mathcal{X}_l^{t+1} = \mathcal{X}_l^t \cup (X_i^*, Y_i)$
  **end if**
  Update model with $\mathcal{X}_l^{t+1}$
  $\mathcal{X}_u^{t+1} = \mathcal{X}_u^t \setminus X_i^*$
  $b = b + \mathcal{C}(X_i^*, s^*)$, $t = t + 1$
**end while**
```

3.4 Experiments and Results

In this section, we describe the datasets we use for our experiments and the experimental setup, followed by results and discussion.

3.4.1 Data and Experimental Setup

To the best of our knowledge, there are only two text datasets available with rationales annotated as sub-strings. The first dataset (*MR*) is a collection of movie reviews annotated with rationales (Zaidan et al., 2007). The goal here is to classify a movie review as expressing positive or negative sentiment (P vs. N) towards the movie. We use unigram[6] features that occur more than 3 times in the whole set, and normalize the features for an instance and contrast instance as in (Zaidan et al., 2007). We divide the dataset (900 positive, 900 negative) into 10 folds with equal numbers of positive and negative instances. One fold forms our development set, for estimating the parameters. We evaluate our models on one of the other folds, and run active learning experiments on the remaining folds.

The second dataset is the Aviation Safety Reports (*ASR*) data annotated with rationales by Abedin et al. (2011). The task here is to identify which of the 14 cause factors were the cause of the problem described in a given report. We evaluate our approach on 3 pair-wise classification tasks for three cause factors: {Communication Environment (CE), Proficiency (PR), Resource Deficiency (RD)}[7]. Table 3.2 summarizes some statistics about the two datasets. As can be seen, aviation safety reports are shorter in length than movie reviews. There are also fewer rationales per instance for *ASR* dataset than *MR* dataset. The percentage of rationales per instance for *RD* class is comparable to the *MR* dataset. However, for *CE* and *PR* classes, the percentage of rationales is much smaller. As mentioned before, the annotation cost was found to increase linearly with the number of rationales (Chapter 4). Thus, we expect a lower rationale cost factor (α) for *ASR* dataset than *MR* dataset. Since *ASR* dataset is small, we use unigram features without any filtering and average results over three random splits of the data into 5 folds. In order to make a comparison across different learning problems in this dataset, we use the same amount of data for each class. As for the *MR* dataset, we split the data into folds such that there is an equal number of positive and negative instances in each fold. In the end, we have 21 instances per class in each fold.

	#I	**Avg. #SI/I**	**Avg. #RSI/I**	**Avg. %RSI/I**	**Avg. SI CharLen**	**Avg. RSI CharLen**	**Avg. % RatText**
CE	141	15.7	1.50	12.4	108.2	114.9	13.0
PR	268	14.0	1.40	13.5	105.8	123.8	15.7
RD	509	13.7	3.22	28.0	98.8	116.9	32.5
MR	1800	32.6	8.00	26.8	120.2	129.1	28.5

Table 3.2: Data statistics for *ASR* (CE, PR, RD), and *MR* datasets. #SI/I = number of sub-instances per instance; #RSI/I = number of rationale sub-instances per instance; %RSI/I = percentage of rationale sub-instances per instance; SI CharLen = character length of a sub-instance in an instance; RSI CharLen = character length of a rationale sub-instance in an instance; % RatText = percentage of text in an instance that is covered by rationales.

[6]Extracted using the tokenizer in Stanford Parser (Klein and Manning, 2003) (`http://nlp.stanford.edu/software/lex-parser.shtml`).

[7]We only use part of the data (training set) that is annotated with rationales. Only 6 out of 14 classes have more than 100 positive instances. So far, we have evaluated our approach for 3 of them. For each pairwise classification task, we exclude the instances that belong to both the classes.

We use MxTerminator (Reynar and Ratnaparkhi, 1997)[8] to segment a document into sentences, which constitute our sub-instances. For the two datasets, we extend the rationale sub-strings in a document to sentence boundaries, to get the rationale sub-instances. Yessenalina et al. (2010) found that this does not have a significant effect on performance, for MR dataset. As mentioned in Section 3.3, we have a second classifier that predicts whether a sub-instance is a rationale or not (R vs. NR). For each rationale sub-instance, a pseudo-instance is added to the main classifier, as described in Section 3.2. The same instance is also added to the R vs. NR classifier with the label R. Pseudo-instances are also constructed for the non-rationale sub-instances, and these are added to the R vs. NR classifier with the label NR. Since ASR dataset is small and has fewer rationales, R vs. NR classifier may not predict any rationales for an instance. Since we assume that there is at least one rationale in each instance, if no rationales are predicted for an instance, we select the most likely one as the rationale. In this rationale predictor we make a simplifying assumption that labels for sub-instances in an instance are independent of each other, given the instance's label. A more sophisticated model that relaxes this assumption may improve the performance further.

We set the parameters for the cost of dissatisfying the margin constraints in SVMs (C) and contrast constraints ($C_{contrast}$) in (Zaidan et al., 2007) to the default value of 1. We select the μ parameter based on performance on the development set, averaged over folds, for random instance selection and fixed strategy of LR.

Risk estimation (for calculating VOI) is an expensive operation, as it requires learning and unlearning for every possible label for each candidate instance and its expected rationales. We developed an efficient implementation in Java of an online incremental/decremental SVM algorithm proposed by Cauwenberghs and Poggio (2000), based on the C++ implementation in (Vijayanarasimhan and Grauman, 2011). To speed up the selection process further, we apply VOI based selection to a subset of instances (set to 50 for our experiments). We select these instances using a common active selection criteria of *uncertainty* sampling, which selects instances the model is least certain about (i.e. closest to the hyperplane in SVMs). Since the ASR dataset is relatively small, we select an instance using VOI from the complete unlabeled pool. On a server with 6 cores (2.4GHz) and a total of 24GB of RAM (program only requires maximum of 8GB RAM for MR and $1GB$ for ASR dataset), using a sequential algorithm, it takes on average 0.1 seconds to select an instance and strategy for ASR dataset. Since MR dataset is larger with more rationales, it takes on average 44 seconds to select a candidate. This can be improved further by parallelizing the algorithm.

We compare different methods in terms of performance for given annotation cost. Ideally, for evaluation, one should use the clock time for annotation cost. However, since we do not have the annotation times for every instance and strategy pair, we use the same estimate for annotation cost based on instance's length described in Section 3.3. In the literature, length of a text instance (e.g. in terms of the number of words) has been used as an estimate for annotation cost for comparing different active learning methods (Engelson and Dagan, 1996; Tomanek et al., 2007). Also, as mentioned before, a linear correlation between annotation time and length of a text instance has been observed in the literature (Ringger et al., 2008; Arora et al., 2009b).

For MR data, we run the active learning experiments up to a total annotation cost of $10,000$ units[9]. We measure performance, in terms of the accuracy, for every 500 annotation cost units spent. We call this a *cost milestone*. The results are plotted with accuracy averaged over 10 folds at a given cost milestone. We measure significance using a two-tailed paired t-test and represent

[8]`http://web.mit.edu/course/6/6.863/tools/jmx/MXTERMINATOR.html`

[9]About $400 - 450$ instances for LR strategy. For LO strategy, number of instances depend on the value of α (about 750 instances for $\alpha = 2$).

the results using a character string, one character for each milestone. I denotes an insignificant, S denotes a significant ($p < 0.05$), and M denotes a marginally significant ($p < 0.1$) difference. For example, a significance string of '`SSSSMSSSSSIISSSSSIMI`' indicates that the difference is marginally significant at 2500 and 9500, insignificant at 5500, 6000, 9000 and 10000, and significant at all other cost milestones from 500 to 10000 (at every 500 cost milestone). For calculating the risk, we estimate probabilities from raw SVM scores. We use the development set to estimate these probabilities, by fitting a sigmoid function to SVM's output (Platt, 1999). We repeat this process at each increment of 500 in total cost.

For ASR data, we show results up to a total cost of 700. That is when we run out of instances for LO strategy at $\alpha = 2$, for some classes. Since this dataset is small, we fit the sigmoid after each iteration. We report performance and significance results at every 100 cost milestone.

3.4.2 Results and Discussion

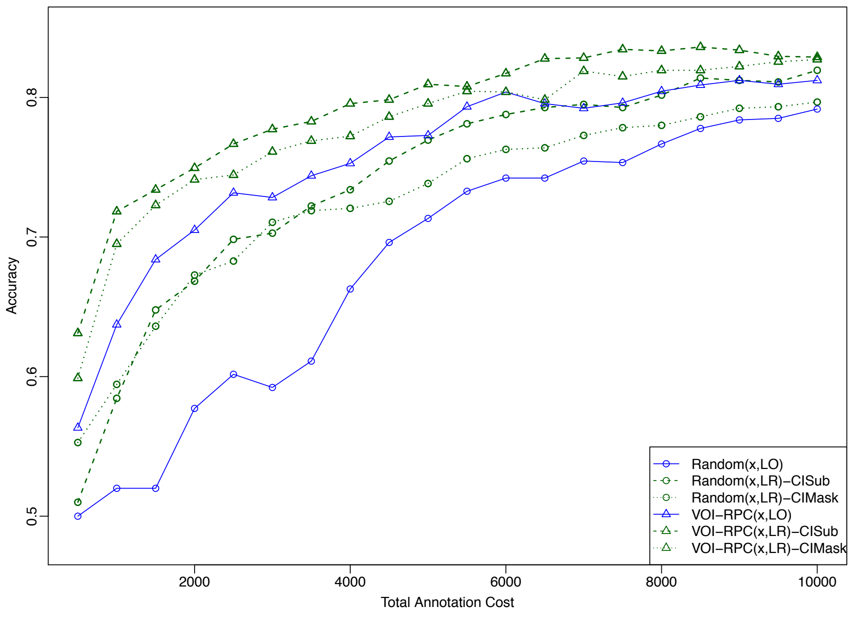
We first compare the fixed strategies of LO and LR, that is, we evaluate whether annotating rationales in addition to the instance's label is useful. Then, we compare strategy selection to the fixed strategies, for variants of VOI discussed above. Following this, we present an analysis of the effect of number of rationales on model's performance.

Comparison of Annotation Strategies: *LO* vs. *LR*

We first make this comparison for $\alpha = 1$, that is, there is no extra cost for annotating rationales. This is to understand if rationales as sub-instances in an instance are useful. For LR strategy, we also compare the two variants for constructing contrast instances (*CISub* and *CIMask*), discussed in Section 3.2. We compare the two fixed strategies of LO and LR for random, and one of the VOI based instance selection (VOI-RPC) methods. We start with an initial training set of one instance per class, and in each iteration an instance is selected randomly, or actively for the given strategy.

Figure 3.6 and Figure 3.7 summarize the results for MR dataset and ASR dataset respectively. As can be seen, for MR dataset, *CISub* outperforms *CIMask* significantly, for both random and VOI based instance selection. For ASR dataset, except for CE vs. RD task with random instance selection (in some parts of the learning curve), *CISub* outperforms *CIMask* significantly for all other cases. As mentioned in Section 3.2, if rationales share a lot of features with each other or with other parts of the document, a contrast instance constructed using $CIMask$ approach may not be much different from the original instance, and contrast constraints may not provide much additional information to the model. Thus, going forward, we use *CISub* approach for constructing contrast instances.

From Figure 3.6 we see that annotating rationales in addition to providing the instance's label improves performance significantly, for both random and VOI based instance selection. The difference between *Random(x,LR)-CISub* and *Random(x,LO)* is statistically significant at all cost milestones. The difference between *VOI-RPC(x,LR)-CISub* and *VOI-RPC(x,LO)* is significant at several cost milestones. The benefit from rationales decreases as learning progresses. This is expected, as the model matures, the extra information that rationales provide may not be as helpful as in the beginning. However, even later on in learning, rationales do not hurt the model and there is significant benefit from them. The benefit from rationales seems more for random instance selection than VOI based instance selection. However, note that VOI based instance selection even without rationales outperforms random instance selection with rationales (the difference is



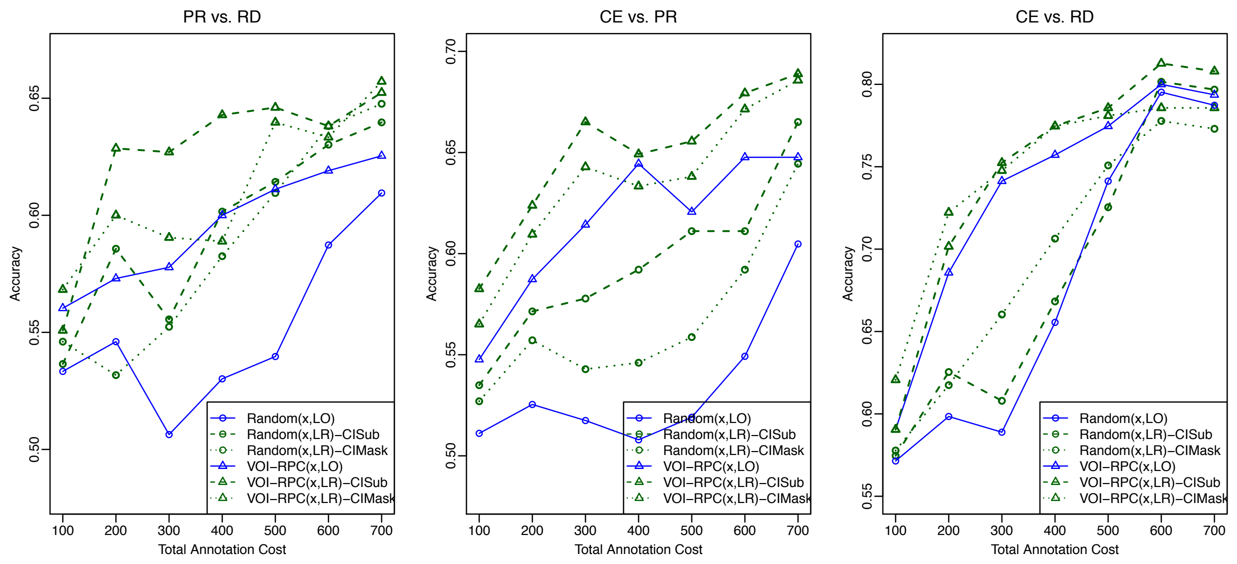
	CISub vs. CIMask	**LO vs. LR-CISub**
R	SIIIIIIISSSSSSISSSMM	SSSSSSSSSSSSSSSSSSSS
VOI	IMIIIIIIIIIISIMIIIII	SSSSMSSSSSIISSSSSIMI

	LO	**LR**
VOI vs. Random	SSSSSSSSSSSSSSSSSSSM	SSSSSSSSSSSSSSSSSSII

Figure 3.6: Comparison of the two strategies of LO (solid lines) and LR (dotted or dashed lines) for $\alpha = 1$, for MR dataset. *Triangles* for VOI and *circles* for random instance selection. Significance strings summarize results from 500 to 10000 (at every 500) cost milestone.

marginally significant at a few points in the beginning). We see that rationales provide further benefit to the model when the instances are actively selected as those expected to bring the most benefit from rationales. From Figure 3.6 we also see that for the fixed strategies of LO and LR, VOI based instance selection outperforms random instance selection. This difference is significant at almost all cost milestones.

We observe similar results for ASR dataset (Figure 3.7). The fixed strategy of LR significantly outperforms the fixed strategy of LO for all three tasks, for both random and VOI based instance selection. As can be seen, for a given annotation cost, we are able to achieve much higher accuracy (with or without rationales) for CE vs. RD task than the other two tasks. This suggests that distinguishing PR class from the other two classes is a harder problem, i.e., it requires more supervision to achieve the same performance. However, we see more benefit from rationales for CE vs. PR and PR vs. RD tasks, than CE vs. RD task. This suggests that rationales are more useful for PR class than CE and RD classes. One caveat is that Abedin et al. (2011) did not retain rationales inline and every occurrence of a rationale sub-string is considered a rationale. This however may not be true, specifically for short rationale strings. Figure 3.8 shows a table of most frequent rationales for each cause factor, taken from (Abedin et al., 2011). It is clear that some

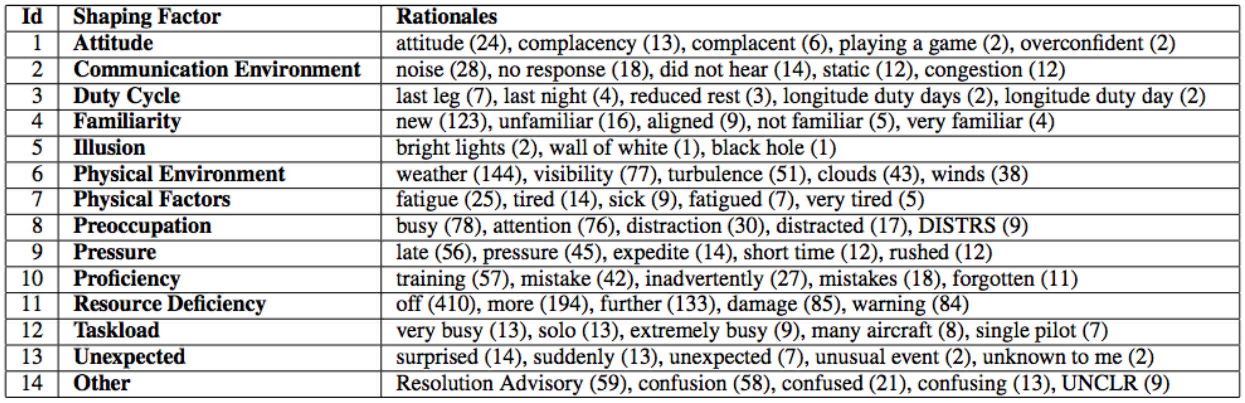


	PR vs. RD	**CE vs. PR**	**CE vs. RD**
CISub vs. CIMask			
Random	`ISIIIII`	`IISSSMI`	`IISSIMS`
VOI	`IMMSIII`	`IIIIIII`	`IIIIISI`
LO vs. LR-CISub			
Random	`IMSSSSI`	`MSSSSSS`	`SIMSIMS`
VOI	`ISSSSIM`	`SMSISMS`	`IIIIIIS`
VOI vs. Random			
LO	`IISSSII`	`MMSSSSM`	`ISSSIII`
LR	`IISMIII`	`IISSSSI`	`ISSSSII`

Figure 3.7: Comparison of the two strategies of LO (solid lines) and LR (dotted or dashed lines) for $\alpha = 1$, for three classification tasks in ASR dataset. *Triangles* for VOI and *circles* for random instance selection. Significance strings summarize results from 100 to 700 (at every 100) cost milestone.

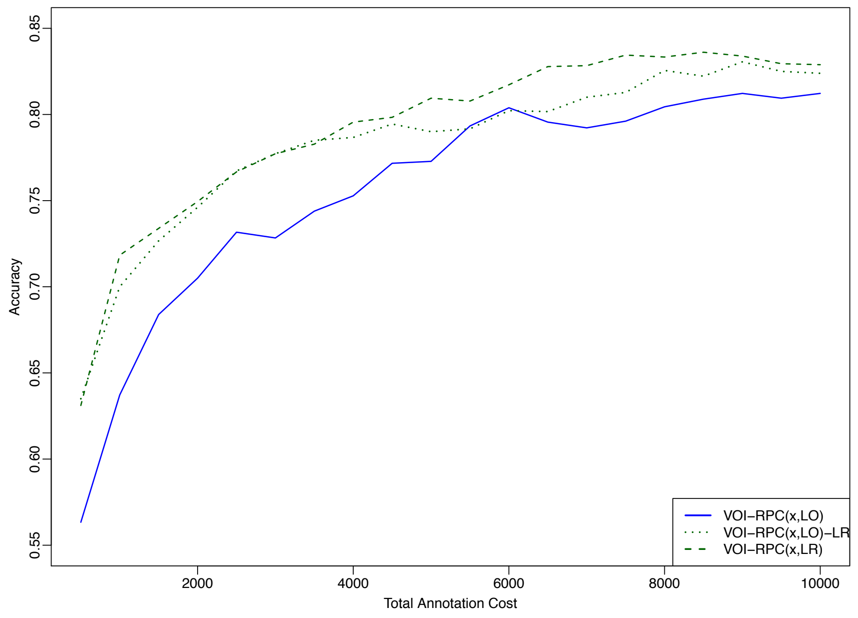
rationale strings, such as ‘off’ for *resource deficiency* (RD), may not be a rationale in every context. For example, the occurrence of ‘off’ in “just prior to outer marker we were handed off to tower.” is not a rationale. On the other hand, top rationales for *proficiency* (PR) (e.g. training, mistake, etc.) seem like they may be rationales in most of the contexts. This noise in rationale annotations, which may be more or less for different classes, may explain the differences we observe in benefit from rationales for different classification tasks. Additionally, for some classes, unigram features may be ambiguous and longer n-grams or other linguistic features may be needed. However, such features are often rare and may require a larger dataset than what we have. From Figure 3.7 we also see that VOI based instance selection outperforms random instance selection significantly for both strategies of LO and LR.

In the results above, in each iteration the most suitable instance for a fixed strategy of LO or LR is selected using VOI. We saw that with this strategy sensitive instance selection, fixed strategy of LR significantly outperforms the fixed strategy of LO. We also compare the two fixed strategies



Id	Shaping Factor	Rationales
1	**Attitude**	attitude (24), complacency (13), complacent (6), playing a game (2), overconfident (2)
2	**Communication Environment**	noise (28), no response (18), did not hear (14), static (12), congestion (12)
3	**Duty Cycle**	last leg (7), last night (4), reduced rest (3), longitude duty days (2), longitude duty day (2)
4	**Familiarity**	new (123), unfamiliar (16), aligned (9), not familiar (5), very familiar (4)
5	**Illusion**	bright lights (2), wall of white (1), black hole (1)
6	**Physical Environment**	weather (144), visibility (77), turbulence (51), clouds (43), winds (38)
7	**Physical Factors**	fatigue (25), tired (14), sick (9), fatigued (7), very tired (5)
8	**Preoccupation**	busy (78), attention (76), distraction (30), distracted (17), DISTRS (9)
9	**Pressure**	late (56), pressure (45), expedite (14), short time (12), rushed (12)
10	**Proficiency**	training (57), mistake (42), inadvertently (27), mistakes (18), forgotten (11)
11	**Resource Deficiency**	off (410), more (194), further (133), damage (85), warning (84)
12	**Taskload**	very busy (13), solo (13), extremely busy (9), many aircraft (8), single pilot (7)
13	**Unexpected**	surprised (14), suddenly (13), unexpected (7), unusual event (2), unknown to me (2)
14	**Other**	Resolution Advisory (59), confusion (58), confused (21), confusing (13), UNCLR (9)

Figure 3.8: Example of five most frequent rationale strings (with their frequency) for each cause factor as in (Abedin et al., 2011).

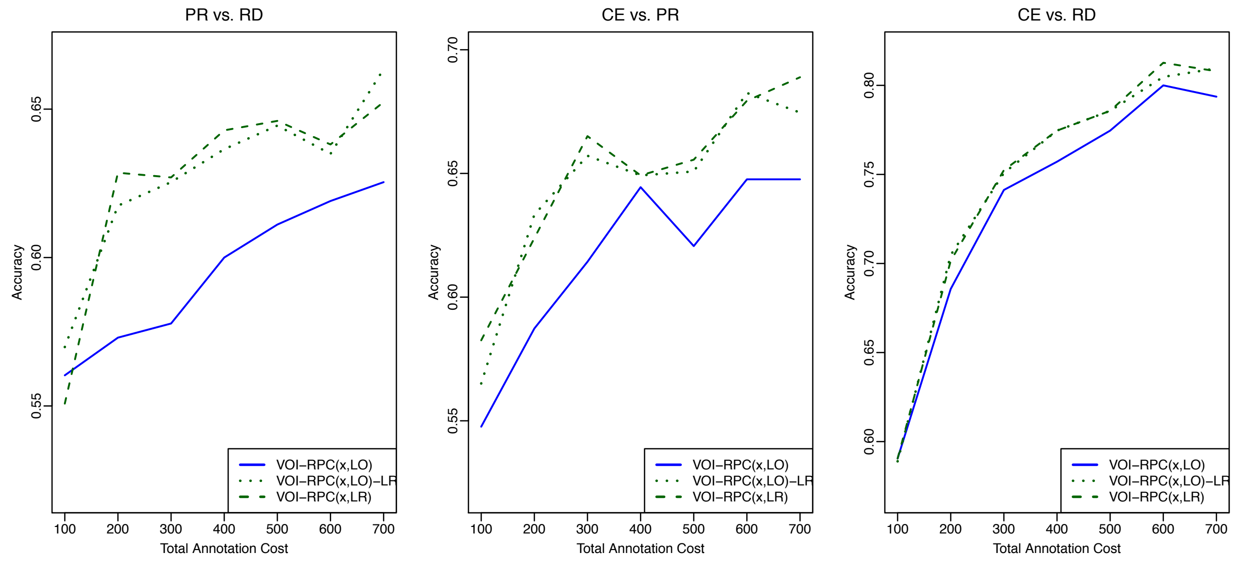


$VOI\text{-}RPC(x, LR)$ vs. $VOI\text{-}RPC(x, LO)\text{-}LR$	`IIIIIIIIIIIISIMIIIII`
$VOI\text{-}RPC(x, LR)$ vs. $VOI\text{-}RPC(x, LO)$	`SSSSMSSSSSIISSSSSIMI`
$VOI\text{-}RPC(x, LO)\text{-}LR$ vs. $VOI\text{-}RPC(x, LO)$	`SSSSSSSSSSIIIMMSIIII`

Figure 3.9: Comparison of strategy sensitive (dashed) and strategy independent (dotted) instance selection with LR, with reference of LO strategy (solid line) for $\alpha = 1$, for MR dataset.

for same instance selection criteria. An instance is selected as the one expected to bring the most benefit to the model for the instance's label. That is, in each iteration an instance is selected for the LO strategy and we compare the fixed strategies of LO and LR for this instance selection procedure (i.e. $VOI\text{-}RPC(x, LO)\text{-}LR$ vs. $VOI\text{-}RPC(x, LO)$). Since strategy sensitive instance selection for LR strategy computes an approximation of the expected risk using a rationale predictor (Section

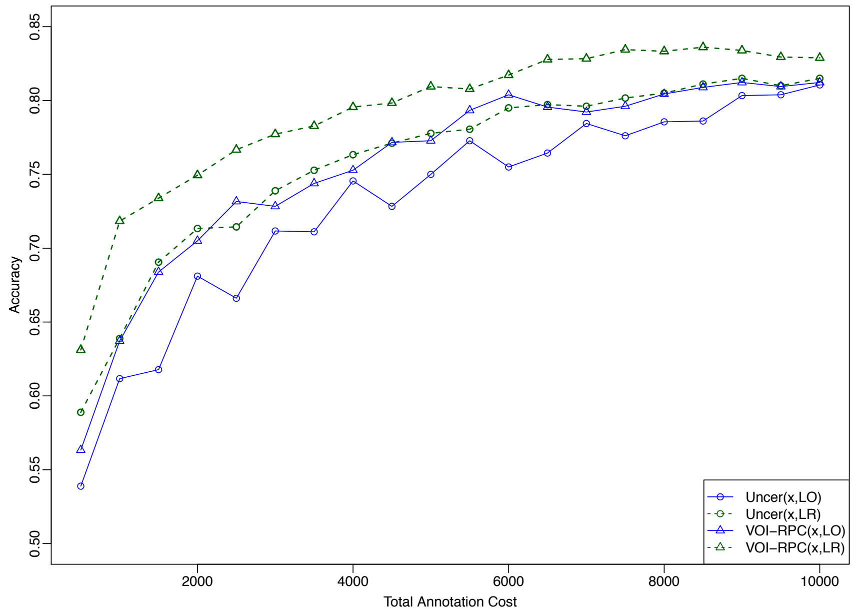
3.3), it is also important to evaluate it against a simpler approach of using instances selected for LO strategy. As can be seen from Figure 3.9, there is significant benefit from rationales, even when a simpler instance selection procedure of selecting instances for LO strategy is used. However, the benefit from rationales is more when instances are actively selected for the LR strategy. That is, strategy sensitive instance selection for a fixed strategy of LR ($VOI\text{-}RPC(x, LR)$) is better than seeking rationales for instances selected for LO strategy ($VOI\text{-}RPC(x, LO)\text{-}LR$). This difference is significant or marginally significant at a few points in the learning curve. This supports our intuition that rationales may not be equally useful for all instances, actively selecting the instances as those expected to bring the most benefit from rationales is useful.



	PR vs. RD	CE vs. PR	CE vs. RD
$VOI\text{-}RPC(x, LR)$ vs. $VOI\text{-}RPC(x, LO)\text{-}LR$	IIIIIIM	IIIIIII	IIIIIII
$VOI\text{-}RPC(x, LR)$ vs. $VOI\text{-}RPC(x, LO)$	ISSSSIM	SMSISMS	IIIIIIS
$VOI\text{-}RPC(x, LO)\text{-}LR$ vs. $VOI\text{-}RPC(x, LO)$	ISSMSIS	ISSIMMM	IIIIIIS

Figure 3.10: Comparison of strategy sensitive (dashed) and strategy independent (dotted) instance selection with LR, with reference of LO strategy (solid line) for $\alpha = 1$, for MR dataset.

Figure 3.10 presents the results for comparison of strategy sensitive and strategy insensitive instance selection for LR strategy, for ASR dataset. As can be seen, strategy-sensitive instance selection is better than using the same instance selection criteria as LO strategy, for a few points in the learning curve. However, this difference is not significant. Note that strategy-sensitive instance selection for LR strategy uses a classifier that is trained on the data annotated so far to predict rationales. Since the ASR dataset is small and has fewer rationales, the rationale classifier is a weak classifier and hence risk estimate may not be as accurate, especially in the beginning. Nonetheless, it performs better than using the same instance selection criteria as that for the LO strategy. Also, we see a significant benefit from rationales (LO vs. LR) for more milestones with strategy sensitive instance selection. Thus, going forward we use strategy sensitive instance selection criteria for LO and LR strategies. Note that for PR vs. RD task, towards the end of the learning curve, strategy sensitive instance selection is marginally worse than strategy independent instance selection, although it performs better at few points in the beginning. Since this dataset is small, towards the end of the learning curve we are left with very little unlabeled data for risk estimation and hence risk estimation may not be as accurate.

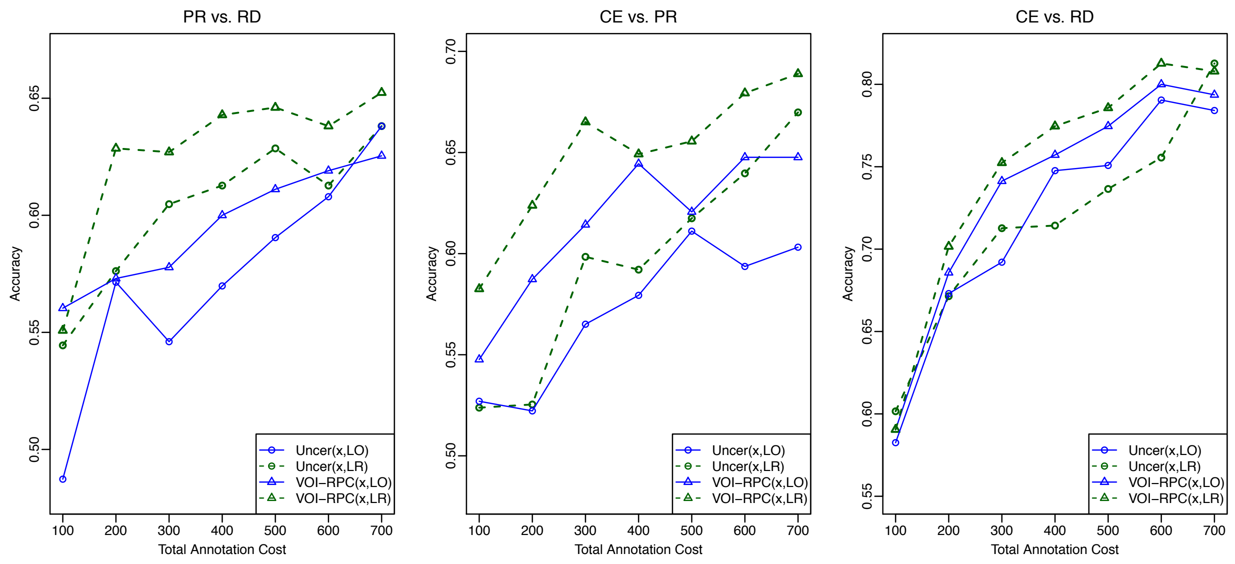


	LO	**LR**
VOI vs. Uncer.	IISIMMMISSMSSISIIIII	SSSSSSSSMMIISSSMMIII

Figure 3.11: Comparison of VOI with uncertainty based instance selection for two strategies of LO (solid lines) and LR (dashed lines) for $\alpha = 1$, for MR dataset. *Triangles* for VOI and *circles* for uncertainty based instance selection. Significance strings represent results from 500 to 10000 (at every 500) cost milestone for MR dataset.

Uncertainty sampling is a common active selection criteria used in active learning. In each iteration, an instance the model is least certain about (i.e. closest to the hyperplane in SVMs) is selected for annotation. We compare the proposed VOI approach to uncertainty based instance selection, for the two strategies (LO and LR). For MR dataset, since we only apply VOI criterion to a subset of instances selected using uncertainty, it is important to compare against it to see whether VOI provides any additional benefit. Figure 3.11 compares VOI based instance selection to uncertainty based instance selection for fixed strategies of LO and LR, for MR dataset. Figure 3.12 presents this comparison for classification tasks in ASR dataset. As can be seen, VOI based instance selection significantly outperforms uncertainty based instance selection for both fixed strategies. Note that both random and uncertainty criteria do not consider cost in instance selection, and cost does vary across instances, even for a fixed strategy. Additionally, VOI actively selects an instance for the given strategy, while uncertainty criterion selects an instance independent of the strategy.

From the discussion above, we see that rationales are helpful, i.e., they provide better performance for the same annotation cost, when there is no extra cost for annotating them. This can be true in the following scenarios:



	PR vs. RD	**CE vs. PR**	**CE vs. RD**
VOI vs. Uncertainty			
LO	`SIIIIII`	`ISMSISM`	`IISIIII`
LR	`IMIIIII`	`ISSSIII`	`IISSSSI`

Figure 3.12: Comparison of VOI with uncertainty based instance selection for two strategies of LO (solid lines) and LR (dashed lines) for $\alpha = 1$, for ASR dataset. *Triangles* for VOI and *circles* for uncertainty based instance selection. Significance strings represent results from 100 to 700 (at every 100) cost milestone.

1. Rationales are gathered implicitly, for example, via eye tracking. Recently, eye tracking has been used to understand the relevance of a document to user's query in text retrieval (Buscher et al., 2012).

2. The user interface time for annotating rationales is negligible. For example, when annotating rationales as sub-instances in an instance, rationales could be solicited from the annotator via a single click anywhere in the sentence. This should be considerably faster than highlighting spans of text, which may involve extra user actions, such as multiple clicks and dragging with mouse, etc.

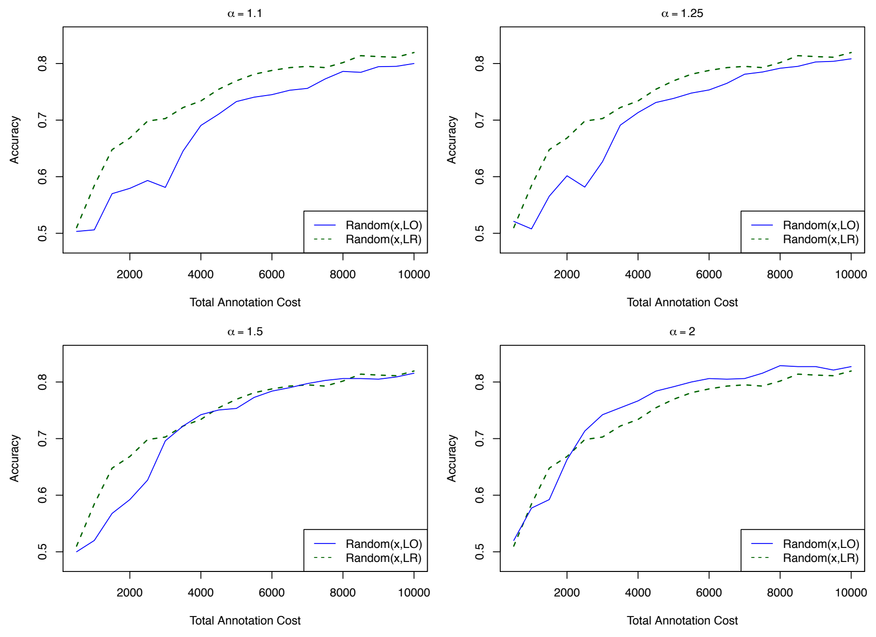
3. Annotating rationales helps the annotator in making a decision about the instance's label. In this case, the extra user interface time for annotating rationales may be compensated for by the reduction in time for providing the instance's label.

However, the cost for annotating rationales may be higher, and it may vary with the annotation task, annotator, user interface design, etc. Next, we consider different costs for annotating rationales, and evaluate when rationales are beneficial for the extra cost. In Section 3.3, we discussed our cost model, where α is the cost factor for annotating rationales. Zaidan et al. (2007) observed that annotating rationales as sub-strings, in addition to providing the instance's label, took twice as much time as providing the instance's label only (i.e. $\alpha = 2$). In our work, we seek rationales as votes on sub-instances in an instance. As mentioned before, voting on sub-instances in an instance should require less time than highlighting sub-strings of text. Thus, we expect α to be less than 2 in our case. Thus, we study the benefit from rationales for $\alpha \in \{1.1, 1.25, 1.5, 2\}$, that is {10%, 25%, 50%, 100%} extra time for annotating rationales.

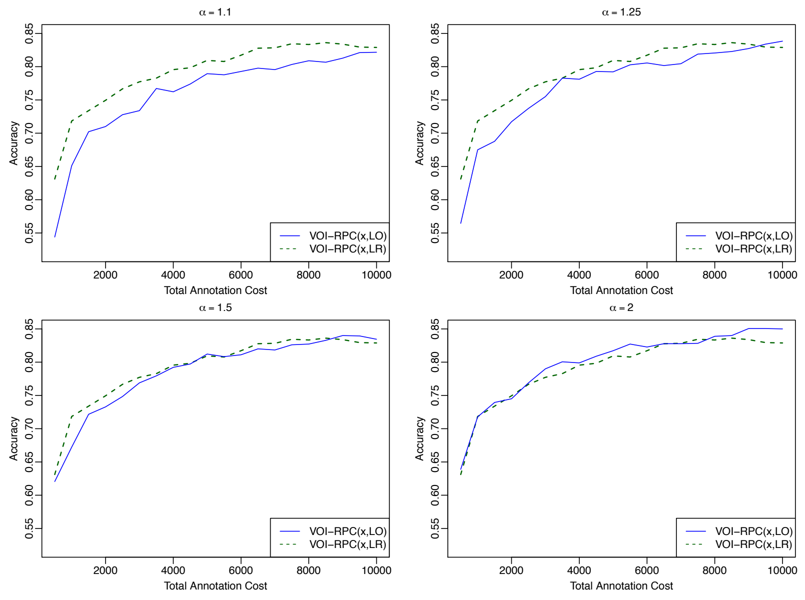
Figure 3.13a compares fixed strategies of LO and LR for random instance selection and different values of α for MR dataset. As can be seen, when α is small, that is, the cost for annotating rationales is small, we see a significant improvement in performance with rationales, for a given annotation cost. The benefit from rationales is more initially in the learning curve. As the model matures, we may not benefit as much from the additional information that rationales provide. As α increases, the gap between LR and LO strategy decreases. There is significant benefit from rationales up to $\alpha = 1.5$. At $\alpha = 2$, LR strategy is worse than LO strategy, at a few cost milestones in the second half of the learning curve. This difference is significant for a few cost milestones. Figure 3.13b shows a similar comparison as above for VOI based instance selection. We see a similar trend here. The benefit from rationales decreases as α increases. The benefit is more earlier on in the learning curve. There is significant benefit from rationales up to a rationale cost factor of $\alpha = 1.5$. For $\alpha = 2$, the two strategies are not different for the most part of the learning curve, except that LO strategy is marginally significantly better than LR strategy at a few points.

Next we compare the two fixed strategies of LO and LR for random and VOI based instance selection, for three classification tasks in ASR dataset: 1) PR vs. RD (Figure 3.14), 2) CE vs. PR (Figure 3.15), and 3) CE vs. RD (Figure 3.16). As for MR dataset, the benefit from rationales decreases as the cost for annotating them increases. For PR vs. RD and CE vs. PR tasks, there is some significant benefit from rationales earlier on in learning, even at $\alpha = 2$. For CE vs. RD task, rationales do not provide any significant improvement in performance for additional cost (i.e. for $\alpha > 1$). For higher values of α, LO strategy is significantly better than LR. As mentioned before, this may be due to noise in rationale annotations for some classes in ASR dataset. Additionally, features beyond unigrams may be needed to capture the relevant information present in rationales.

Thus, whether rationales are helpful may vary depending on the additional cost for annotating rationales and the annotation task. The benefit from rationales may also vary across instances. We saw that strategy sensitive instance selection was better than strategy insensitive instance selection for LR strategy. However, when the additional cost for annotating rationales is high, it may not be beneficial to seek rationales for all instances. The proposed cost-sensitive active selection approach can be used to jointly select an instance and strategy in each iteration. For a given rationale cost factor, the proposed approach automatically determines in each iteration which instance to query and whether to ask for rationales. Next, we compare strategy selection with fixed strategies of LO and LR. We also compare the three variants for computing VOI discussed in Section 3.3.

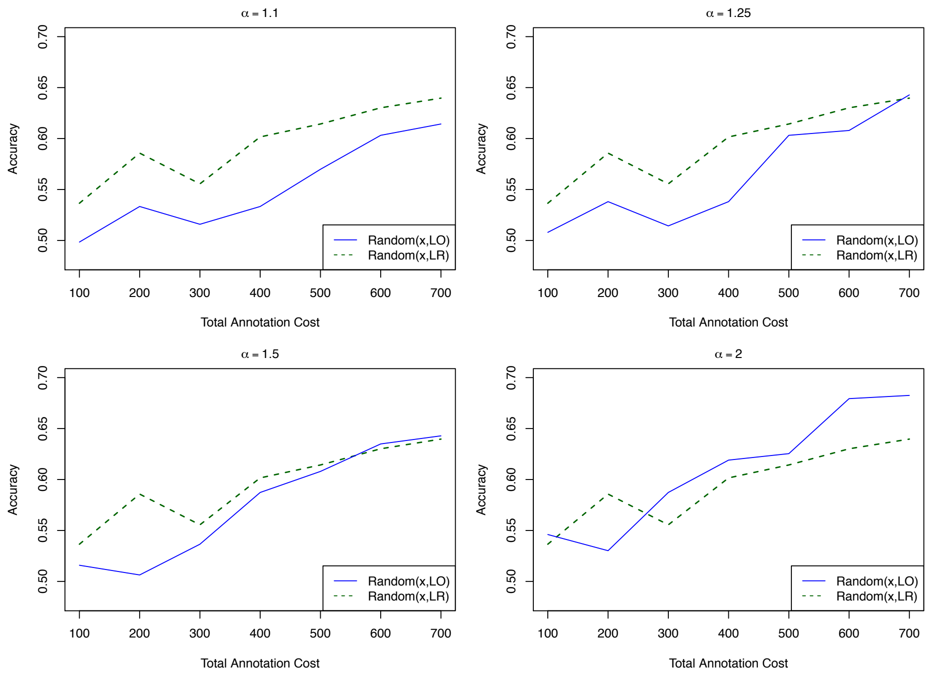


(a) Random Instance Selection. Significance results: (α=1.1): `MSSSSSSMMSSSSSMIMMIM`, (α=1.25): `ISSSSSMIISSSMIIIIIII`, (α=1.5): `SSSSSIIIIIIIIIIIIIII`, (α=2): `IIMIIMISSIIIIISSIMII`

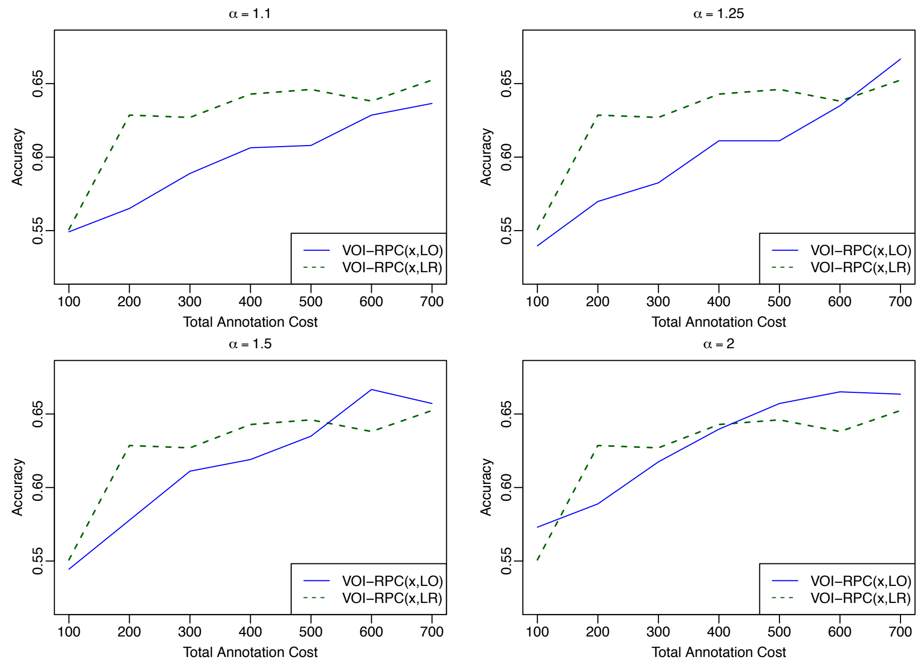


(b) VOI Instance Selection. Significance results: (α=1.1): `SSSSSSISSIIISSSSSMII`, (α=1.25): `SSSSMIIIIIIISSIIIIII`, (α=1.5): `ISIIIIIIIIIIIIIIIIII`, (α=2): `IIIIIIIMIIIIIIIIIIMI`

Figure 3.13: Comparison of LO (solid lines) and LR (dashed lines) strategies for $\alpha \in \{1.1, 1.25, 1.5, 2\}$, for MR dataset. Significance strings report results from 500 to 10000 (at every 500) cost milestone.

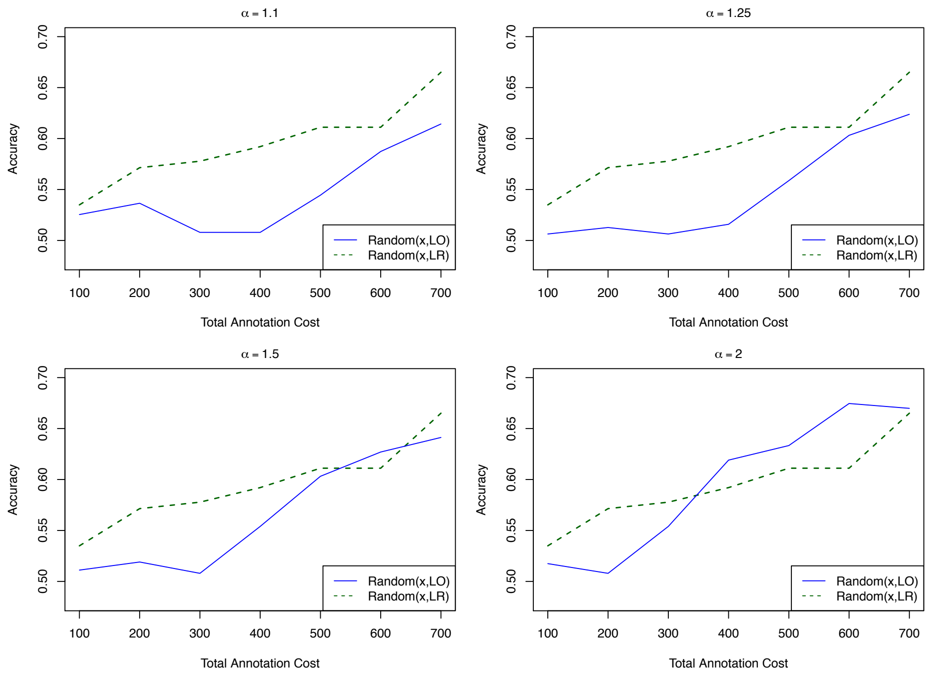


(a) Random Instance Selection. Significance results: (α=1.1): SSSSIII, (α=1.25): IMSSIII, (α=1.5): ISIIIII, (α=2): ISIIISS.

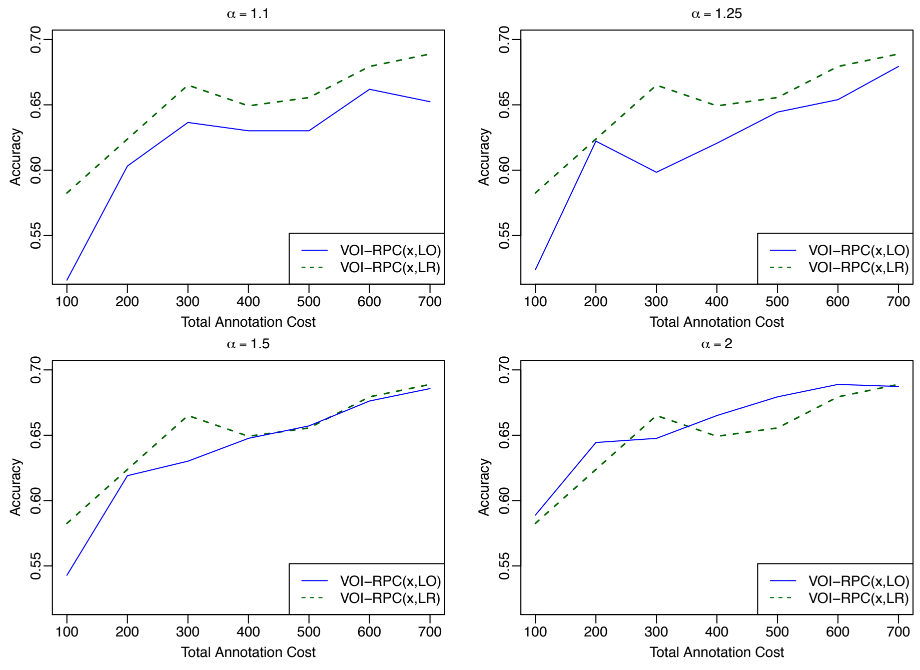


(b) VOI Instance Selection. Significance results: (α=1.1): ISSSSII, (α=1.25): ISSSSII, (α=1.5): ISISISI, (α=2): ISIIIII.

Figure 3.14: Comparison of LO (solid lines) and LR (dashed lines) strategies for $\alpha \in \{1.1, 1.25, 1.5, 2\}$, for PR vs. RD task in ASR dataset. Significance strings summarize results from 100 to 700 (at every 100) cost milestone.

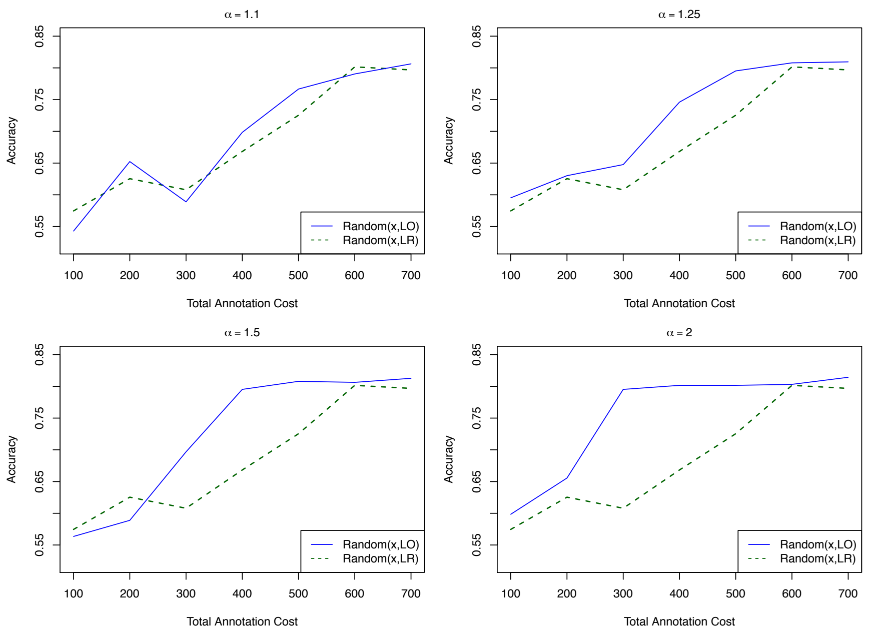


(a) Random Instance Selection. Significance results: (α=1.1): `IMSSSMS`, (α=1.25): `ISSSSIS`, (α=1.5): `ISSMIIM`, (α=2): `ISIIISI`.

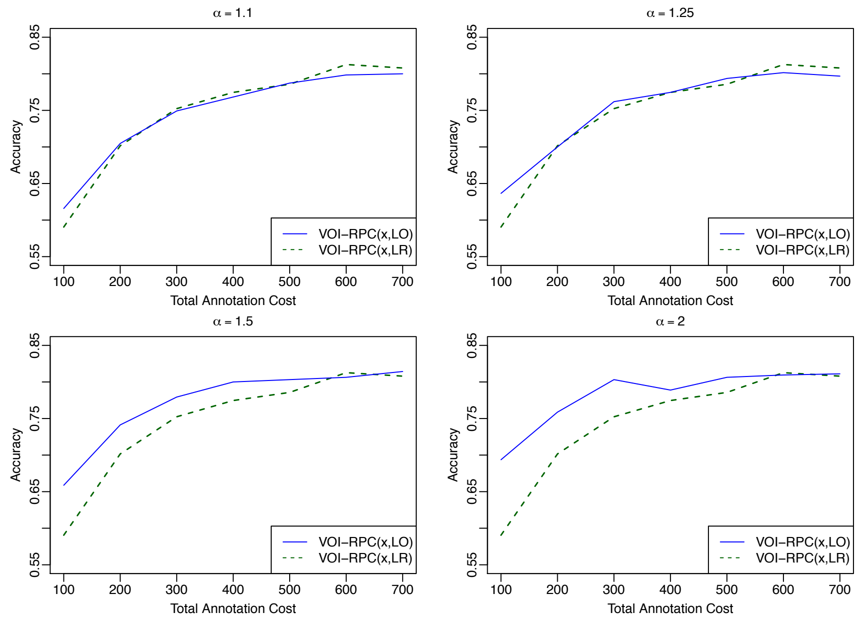


(b) VOI Instance Selection. Significance results: (α=1.1): `SIIIIIS`, (α=1.25): `SISMIMI`, (α=1.5): `SIMIIII`, (α=2): `IIIIIII`.

Figure 3.15: Comparison of LO (solid lines) and LR (dashed lines) strategies for $\alpha \in \{1.1, 1.25, 1.5, 2\}$, for CE vs. PR task in ASR dataset. Significance strings summarize results from 100 to 700 (at every 100) cost milestone.



(a) Random Instance Selection. Significance results: (α=1.1): `IIIISII`, (α=1.25): `IIISSII`, (α=1.5): `IISSSII`, (α=2): `IISSSIM`



(b) VOI Instance Selection. Significance results: (α=1.1): `IIIIIII`, (α=1.25): `MIIIIII`, (α=1.5): `SSIIIII`, (α=2): `SSSIMII`.

Figure 3.16: Comparison of LO (solid lines) and LR (dashed lines) strategies for $\alpha \in \{1.1, 1.25, 1.5, 2\}$, for CE vs. RD task in ASR dataset. Significance strings summarize results from 100 to 700 (at every 100) cost milestone.

Comparison of Strategy Selection with Fixed Strategies for Variants of VOI

As mentioned before, we may not benefit equally from rationales for all instances. For a fixed strategy of LR, i.e. rationales for all instances, VOI approach selects instances that are expected to bring the most benefit from rationales. However, when rationales are very expensive, rationales may not be beneficial for all instances, for the extra cost. Thus, we propose an approach for joint selection of instance and strategy, to seek rationales on only selected instances. When the rationale cost factor (α) is high, that is rationales are expensive, we expect this active learning approach to seek rationales for a few instances, based on the benefit to cost tradeoff. When rationales are cheap, we expect to seek rationales on several instances.

We first compare strategy selection (using variants of VOI discussed in Section 3.3) with a fixed strategy for two scenarios: 1) Rationales come for a small extra cost ($\alpha = 1.25$), and 2) Rationales are very expensive to annotate ($\alpha = 3$). We consider these two extremes, as we expect LR strategy to perform better than LO strategy, for $\alpha = 1.25$, and the opposite for $\alpha = 3$. We choose $\alpha = 3$ and not $\alpha = 2$ for this comparison, since for $\alpha = 2$ (Figure 3.13b) we saw a little difference between the two strategies of LO and LR. For $\alpha = 3$, we expect LO strategy to significantly outperform LR strategy at most cost milestones. We evaluate whether joint selection of instance and strategy is able to follow the better strategy for each rationale cost factor, and if it is better than both fixed strategies. We compare different variants of VOI discussed in Section 3.3 for MR dataset only. This is because, risk estimation is integral to VOI. For an accurate estimate of the model's risk, a reasonably sized unlabeled set is needed. Since ASR dataset is relatively small, a comparison of different VOI methods on this dataset may be misleading. Nonetheless, we later present a comparison of strategy selection (using the most appropriate VOI method) with fixed strategies for several values of $\alpha \leq 2$, for both MR and ASR datasets.

As discussed in Section 3.3, we have a second classifier (R vs. NR) that predicts what sub-instances in an instance are rationales. This classifier's prediction is used to estimate the expected reduction in risk with LR strategy, and it is trained on the data annotated so far. To allow the model to make appropriate strategy selection from the start, we initialize the classifiers (P vs. N and R vs. NR) with 20 instances per class labeled with rationales (for MR dataset). Note that it is particularly important to start with some initial set labeled with rationales for strategy selection (than strategy sensitive instance selection for LR strategy), otherwise without a reasonable classifier for rationales we may underestimate the utility of the LR strategy and may not select the LR strategy as often, especially for a higher value of α. If the LR strategy is not selected, the R vs. NR classifier is not updated. This suggests that we may want to also consider the uncertainty of the R vs. NR classifier when selecting the instance and strategy. We leave this extension to the future work. In Section 3.6, we suggest some ideas for how we may extend the current VOI framework to account for the uncertainty of the R vs. NR classifier. Using a very large initial set defeats the purpose for using active learning to save annotation cost. Thus, we choose to use 20 instances per class, which is about 2% of the whole dataset (1800 instances). We use the same initial set for all methods we compare. As before, we measure significance at every 500 cost milestones up to a total annotation cost of $10,000$.

Figure 3.17 shows the results for comparison of strategy selection and fixed strategies for variants of VOI. As expected, for $\alpha = 1.25$, the fixed strategy of LR is better than LO, for three ways to compute VOI: RTC, SS, and RPC. The difference is significant for several cost milestones in the first half of the learning curve. Table 3.3 shows the complete significance results for the comparisons discussed here. For all variants of VOI, joint selection of instance and strategy performs better

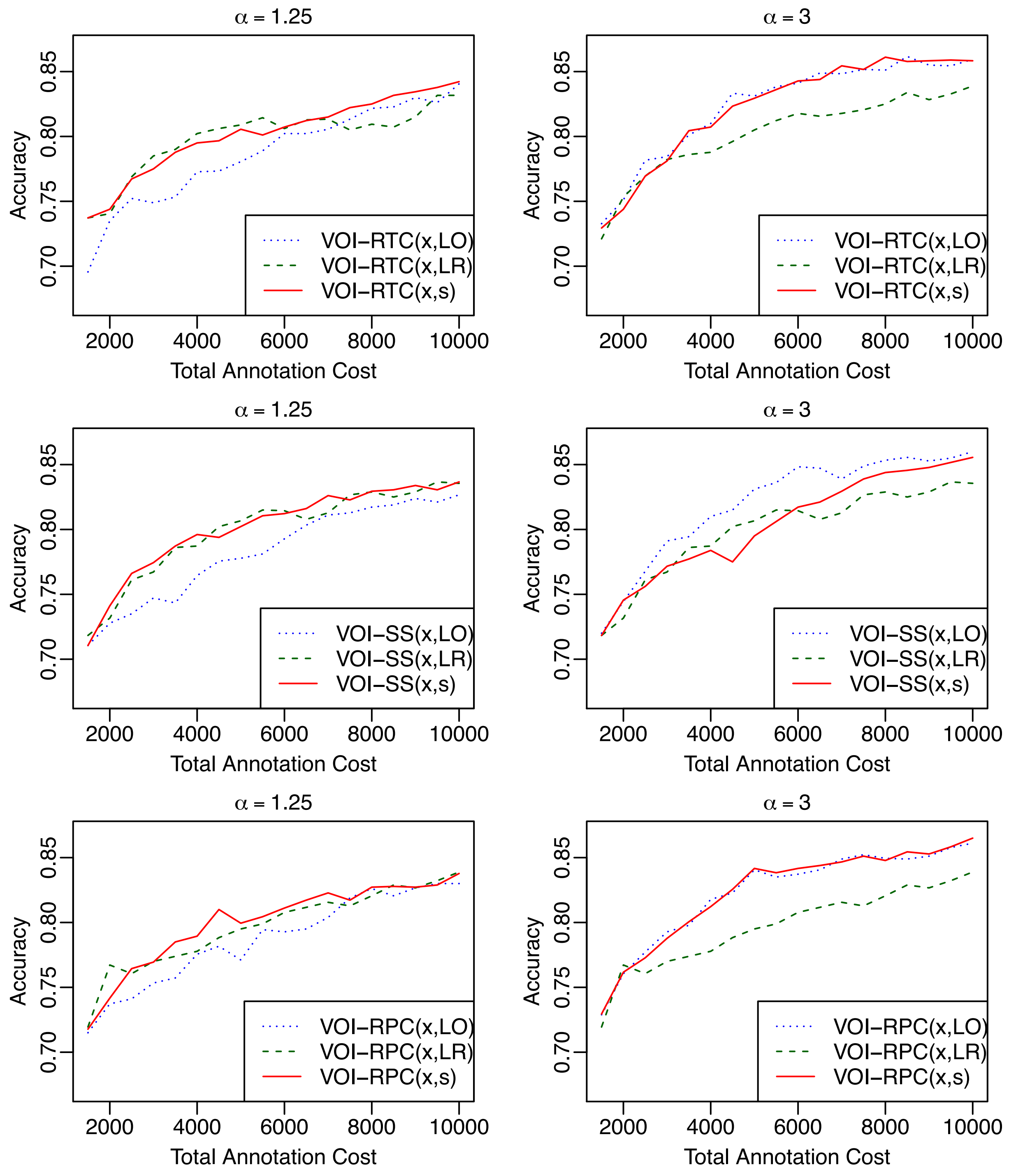


Figure 3.17: Strategy selection vs. fixed strategies for variants of VOI, for MR dataset. Red and solid lines represent joint selection of instance and strategy. Blue and dotted lines represent instance selection for fixed strategy of LO. Green and dashed lines represent instance selection for fixed strategy of LR.

1.25	(x,LO)	(x,LR)	(x,s)
(x,LO)	-	SIISSSSSSIIIIIIIII	SIISSSSIIIIIIIMIII
(x,LR)	SIISSSSSSIIIIIIIII	-	IIIIIIIIIIIIIISSII
(x,s)	SIISSSSIIIIIIIMIII	IIIIIIIIIIIIIISSII	-

3	(x,LO)	(x,LR)	(x,s)
(x,LO)	-	IIIIIISSSSSSSSSSSS	IIIIIIIIIIIIIMIIII
(x,LR)	IIIIIISSSSSSSSSSSS	-	IIIIIIIMSMSSSSSSSS
(x,s)	IIIIIIIIIIIIIMIIII	IIIIIIIMSMSSSSSSSS	-

(a) VOI-RTC

1.25	(x,LO)	(x,LR)	(x,s)
(x,LO)	-	IIMISSSSSSIIIIIIII	IISSSSIMMIIIIIIIII
(x,LR)	IIMISSSSSSIIIIIIII	-	IIIIIIIIIIIIIIIIII
(x,s)	IISSSSIMMIIIIIIIII	IIIIIIIIIIIIIIIIII	-

3	(x,LO)	(x,LR)	(x,s)
(x,LO)	-	IIIMISISSSSSSSSSMS	IIISISSSSSSIIIIIII
(x,LR)	IIIMISISSSSSSSSSMS	-	IIIIIISIIIIIIIMIIM
(x,s)	IIISISSSSSSIIIIIII	IIIIIISIIIIIIIMIIM	-

(b) VOI-SS

1.25	(x,LO)	(x,LR)	(x,s)
(x,LO)	-	ISIIMIISIIIIIIIIII	IIIISISSISMSIIIIII
(x,LR)	ISIIMIISIIIIIIIIII	-	IMIIIISIIIIIIIIIII
(x,s)	IIIISISSISMSIIIIII	IMIIIISIIIIIIIIIII	-

3	(x,LO)	(x,LR)	(x,s)
(x,LO)	-	IIMMISSSSSSSSSSSSS	IIIIIIIIIIIIIIIIII
(x,LR)	IIMMISSSSSSSSSSSSS	-	IIIMISSSSSSSSSSSSS
(x,s)	IIIIIIIIIIIIIIIIII	IIIMISSSSSSSSSSSSS	-

(c) VOI-RPC

Table 3.3: Significance results from 1500 to 10000 (at every 500) cost milestone for a comparison of strategy selection and fixed strategies of LR and LO for variants of VOI, for MR dataset.

than the fixed strategy of LO, for $\alpha = 1.25$. This difference is significant for a few cost milestones in the first half of the learning curve. Actively selecting the strategy jointly with the instance follows the fixed strategy of LR for the most part. For RTC, joint selection of instance and strategy is significantly better than LR strategy towards the end of the learning curve. For RPC, joint selection is marginally significantly worse than LR strategy at one point in the beginning, although it is significantly better than LR at about half way in the learning curve. Thus, for a small rationale cost factor, the fixed strategy of LR is better than the fixed strategy of LO. VOI based joint selection of instance and strategy follows the best fixed strategy in performance, and it is even better than both fixed strategies, at some points for some VOI models (RTC and RPC).

As can be seen from Figure 3.17, for $\alpha = 3$, when rationales are expensive to annotate, the fixed strategy of LO performs better than the fixed strategy of LR, for all three ways of computing VOI. The difference is significant at most cost milestones in the second half of the learning curve. For all VOI methods except SS, joint selection of instance and strategy performs significantly better than

1.25	VOI-SS	VOI-RTC	VOI-RPC
VOI-SS	-	IIMIIIIIIIIIIIIIIM	IIIIIIIIIIIIIIIIII
VOI-RTC	IIMIIIIIIIIIIIIIIM	-	SISIIIIIIIIIIIIIIS
VOI-RPC	IIIIIIIIIIIIIIIIII	SISIIIIIIIIIIIIIIS	-

3	VOI-SS	VOI-RTC	VOI-RPC
VOI-SS	-	IIIIIISIIIIMIIIIII	IIIIIIIMIMIIIIIIII
VOI-RTC	IIIIIISIIIIMIIIIII	-	IIIIIIIIIIIIIISIII
VOI-RPC	IIIIIIIMIMIIIIIIII	IIIIIIIIIIIIIISIII	-

(a) Fixed Strategy of LO

1.25	VOI-SS	VOI-RTC	VOI-RPC
VOI-SS	-	MIIIIIIIIIIIMIIMII	ISIIIISIIIIIIIIIII
VOI-RTC	MIIIIIIIIIIIMIIMII	-	ISIIISSIIIIIIISIII
VOI-RPC	ISIIIISIIIIIIIIIII	ISIIISSIIIIIIISIII	-

3	VOI-SS	VOI-RTC	VOI-RPC
VOI-SS	-	IIIIIIIIIIIIIIIIII	ISIIIISIIIIIIIIIII
VOI-RTC	IIIIIIIIIIIIIIIIII	-	IIIIIIIIIIIIIIIIII
VOI-RPC	ISIIIISIIIIIIIIIII	IIIIIIIIIIIIIIIIII	-

(b) Fixed Strategy of LR

1.25	VOI-SS	VOI-RTC	VOI-RPC
VOI-SS	-	SIIIIIIIIIIIIIIIII	IIIIIIIIIIIIIIIIII
VOI-RTC	SIIIIIIIIIIIIIIIII	-	IIIIIMIIIIIIIIIIII
VOI-RPC	IIIIIIIIIIIIIIIIII	IIIIIMIIIIIIIIIIII	-

3	VOI-SS	VOI-RTC	VOI-RPC
VOI-SS	-	IIIIIISSSSSSSSIIII	IIIIISSSSSSIIIIIII
VOI-RTC	IIIIIISSSSSSSSIIII	-	IIIIIIIMIIIMIMIIII
VOI-RPC	IIIIISSSSSSIIIIIII	IIIIIIIMIIIMIMIIII	-

(c) Strategy Selection

Table 3.4: Significance results from 1500 to 10000 (at every 500) cost milestone for a comparison between variants of VOI for fixed strategy and joint selection of instance and strategy, for MR dataset.

the fixed strategy of LR at most cost milestones[10], and it is not significantly different from the fixed strategy of LO, except that $RTC(x, s)$ is marginally significantly better than $RTC(x, LO)$ at 8000. For SS, joint selection is marginally significantly better than LR in the second half, however it is significantly worse than LR at one point in the middle of the learning curve. It is also significantly worse than LO at several cost milestones in the middle of the learning curve. Thus, VOI-SS fails to select the appropriate strategy for a high rationale cost factor.

Figure 3.18 compares the variants of VOI for a fixed or actively selected strategy. Table 3.4 shows the significance results. For the fixed strategies of LO and LR, the three approaches for computing VOI are quite similar, with only a few significant differences. However, there is no one clear winner among the variants of VOI for a fixed strategy. For joint selection of instance and strategy, for $\alpha = 1.25$, there is little significant difference between the three VOI variants (Only

[10]For joint selection of instance and strategy, LR strategy is selected for a few instances, even for $\alpha = 3$.

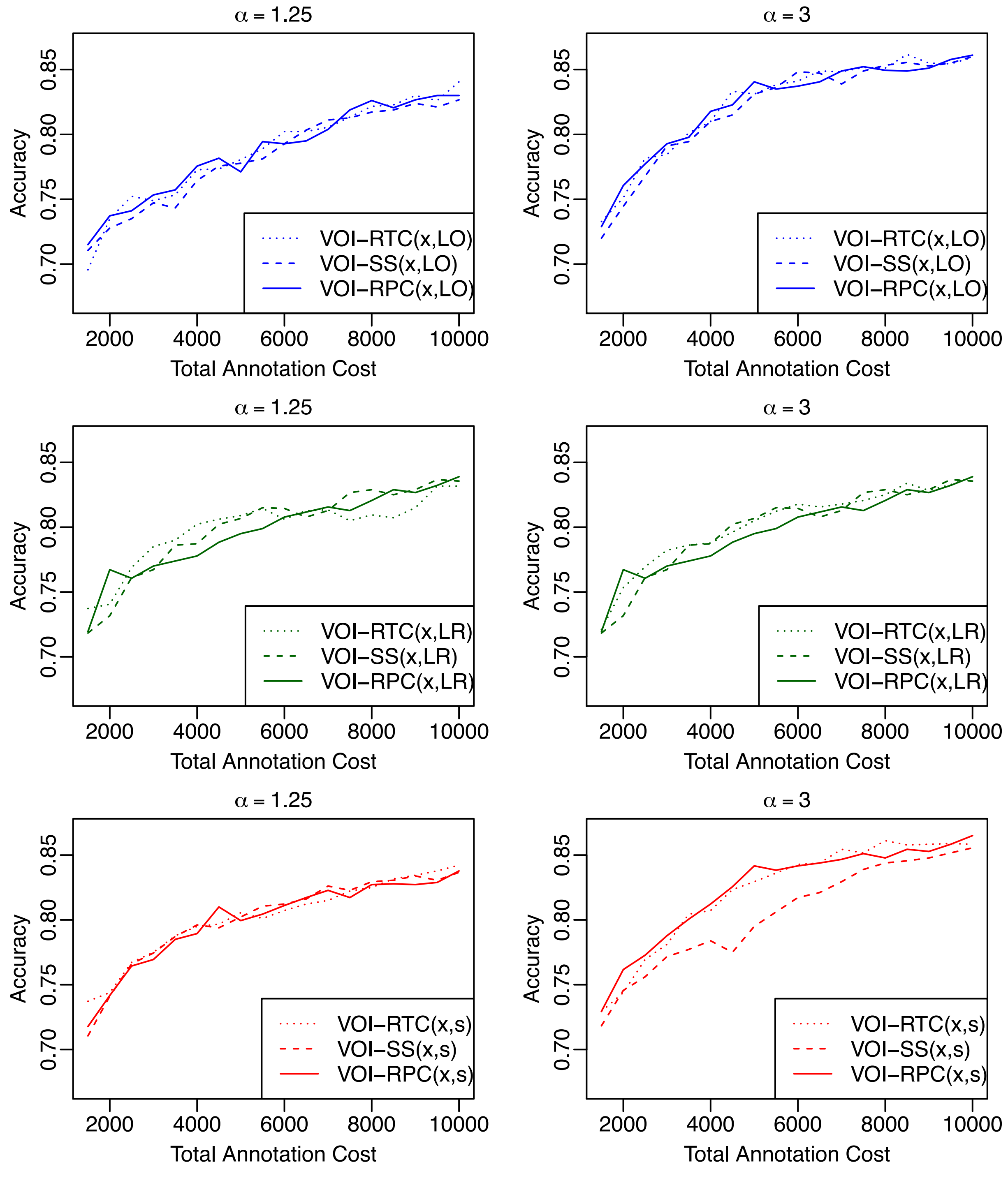


Figure 3.18: Comparison of VOI approaches for a fixed strategy (LR or LO) and joint selection of instance and strategy, for MR dataset.

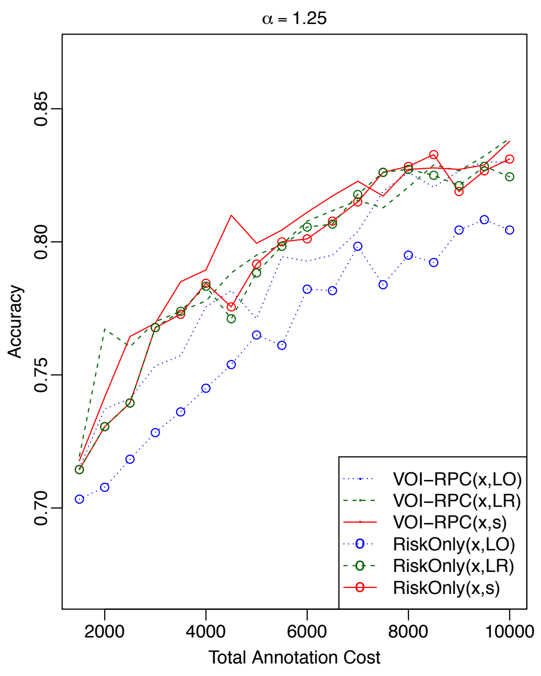
significant difference between RTC and SS at 1500 and marginally significant difference between RPC and RTC at 4500). However, for $\alpha = 3$, there are some significant differences between the variants of VOI. SS is significantly worse than RPC and RTC, for several cost milestones in the middle. VOI-SS treats utility and cost as two independent terms for ranking the candidates, and combines their standardized scores. An alternative to consider is to learn the weights for the utility and cost terms in VOI-SS, similar to (Tomanek and Hahn, 2010), where they use a weighted linear combination of the ranks based on utility and cost scores. Overall, RPC seems to be the best VOI approach. It assumes a large enough unlabeled set for accurate estimation of model's expected risk, however it is not sensitive to the actual size of the data (like RTC). It also has an intuitive reasoning behind it, of selecting the instance and strategy with maximum reduction in expected risk per unit cost. Thus, going forward we use RPC for computing VOI.

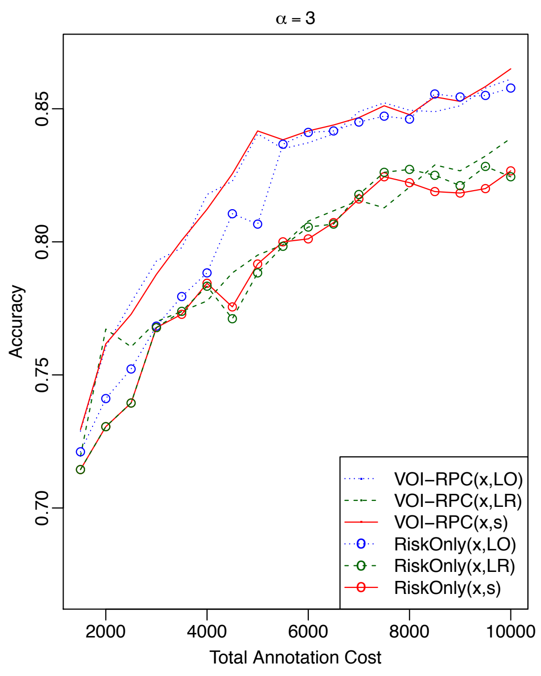
Comparison with other baselines: RiskOnly and CostOnly

VOI considers both reduction in model's risk and cost in active selection. We compare it with risk-only and cost-only selection criteria, to understand the value in using a combined criterion. Figures 3.19 and 3.20 present the results. For the fixed strategy of *LO*, VOI-RPC always performs better than *RiskOnly*, for both rationale cost factors. This difference is significant at several cost milestones. For the fixed strategy of *LR*, except for the cost milestone of 7500 where *RiskOnly* is marginally significantly better than VOI, VOI is better than *RiskOnly* at most cost milestones (significantly at a few cost milestones). For joint selection of instance and strategy, VOI-RPC performs better than *RiskOnly* for both rationale cost factors. This difference is significant at several cost milestones for $\alpha = 3$, and significant for a few cost milestones, for $\alpha = 1.25$. This suggests that considering cost together with risk in active selection is important, especially when the rationale cost factor is high. VOI based joint selection of instance and strategy is also significantly better than *RiskOnly* criterion for instance selection for a fixed strategy of *LO* or *LR*.

For a fixed strategy of *LO*, VOI-RPC is significantly better than *CostOnly* at a few cost milestones, and *CostOnly* is significantly better than VOI-RPC at a few other cost milestones, for both values of α. For *LR* strategy, there isn't much difference between *CostOnly* and VOI-RPC, except that VOI-RPC is significantly better than *CostOnly* at a few milestones in the beginning. For joint selection of instance and strategy, *CostOnly* criterion will always select the *LO* strategy for $\alpha > 1$, and hence it is the same as *CostOnly(x,LO)*. Note that although there isn't much difference between VOI-RPC and *CostOnly* for a fixed strategy, the best fixed strategy for *CostOnly* criterion changes from $\alpha = 1.25$ to $\alpha = 3$, like for the VOI criterion. We see that joint selection of instance and strategy is able to follow the best fixed strategy for both values of α. For $\alpha = 1.25$, it is even significantly better than both fixed strategies (*CostOnly(x,LO)* and *CostOnly(x,LR)*), at a few cost milestones. For $\alpha = 3$, except for two milestones in the first half of the learning curve and one milestone towards the end of the learning curve, where *CostOnly(x,LO)* is better than VOI-$RPC(x, s)$ (significant at one and marginally significant at the other two points), VOI based joint selection of instance and strategy performs as well as or better than both *CostOnly(x,LO)* and *CostOnly(x,LR)*.

Thus, a combined criterion in VOI is better than *RiskOnly* and *CostOnly* criteria for both fixed strategies of *LO* and *LR*, and joint selection of instance and strategy, when both rationale cost factors are considered.





	RiskOnly(x,LO)
VOI-RPC(x,LO)	IIIIMSSISIIISMSMIS
VOI-RPC(x,s)	IISSSSSSSSSSSMSMMS

	RiskOnly(x,LR)
VOI-RPC(x,LR)	ISIIIIIIIIIIMIIIIS
VOI-RPC(x,s)	IISIIISIIIIIIIIIII

	RiskOnly(x,s)
VOI-RPC(x,s)	IISIIISIIIIIIIIIII

(a) $\alpha = 1.25$

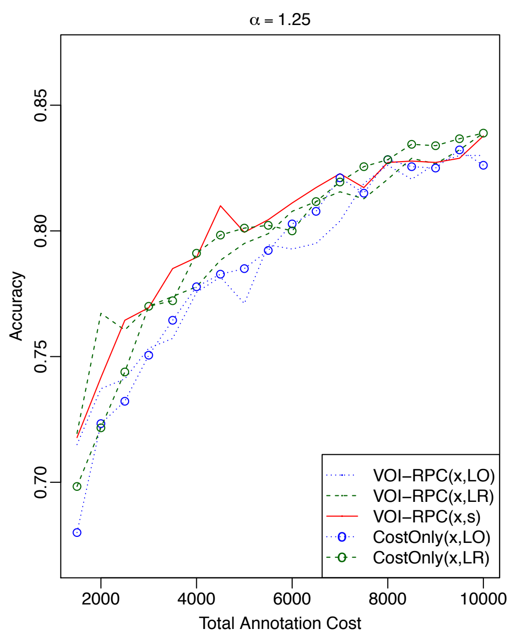
	RiskOnly(x,LO)
VOI-RPC(x,LO)	IMSSMSISIIIIIIIIII
VOI-RPC(x,s)	IMMSMSISIIIIIIIIII

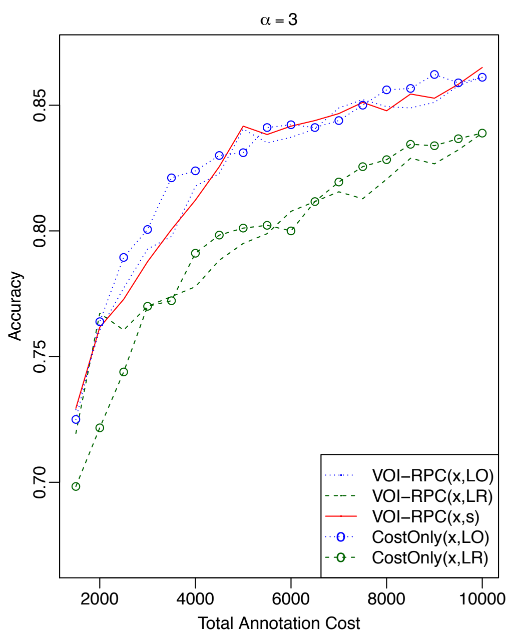
	RiskOnly(x,LR)
VOI-RPC(x,LR)	ISIIIIIIIIIIMIIIIS
VOI-RPC(x,s)	ISSSMSSSSSSSSSSSSS

	RiskOnly(x,s)
VOI-RPC(x,s)	ISSSMSSSSSSSSSSSSS

(b) $\alpha = 3$

Figure 3.19: Comparison of VOI with *RiskOnly* criteria for instance selection for a fixed strategy (LR or LO) and joint selection of instance and strategy, for MR dataset. Red and solid lines represent joint selection of instance and strategy. Blue and dotted lines represent instance selection for fixed strategy of LO. Green and dashed lines represent instance selection for fixed strategy of LR. Significance strings represent results from 1500 to 10000 (at every 500) cost milestone.





	CostOnly(x,LO)
VOI-RPC(x,LO)	`SMIIIIIIIMIMIIIIII`
VOI-RPC(x,s)	`MISIIISIIIIIIIIIII`

	CostOnly(x,LR)
VOI-RPC(x,LR)	`SSIIIIIIIIIIIIIIII`
VOI-RPC(x,s)	`IIMIIISIIIIIIIIIII`

(a) $\alpha = 1.25$

	CostOnly(x,LO)
VOI-RPC(x,LO)	`IIMISIISIIIIIIMSII`
VOI-RPC(x,s)	`IISIMIISIIIIIIIMII`

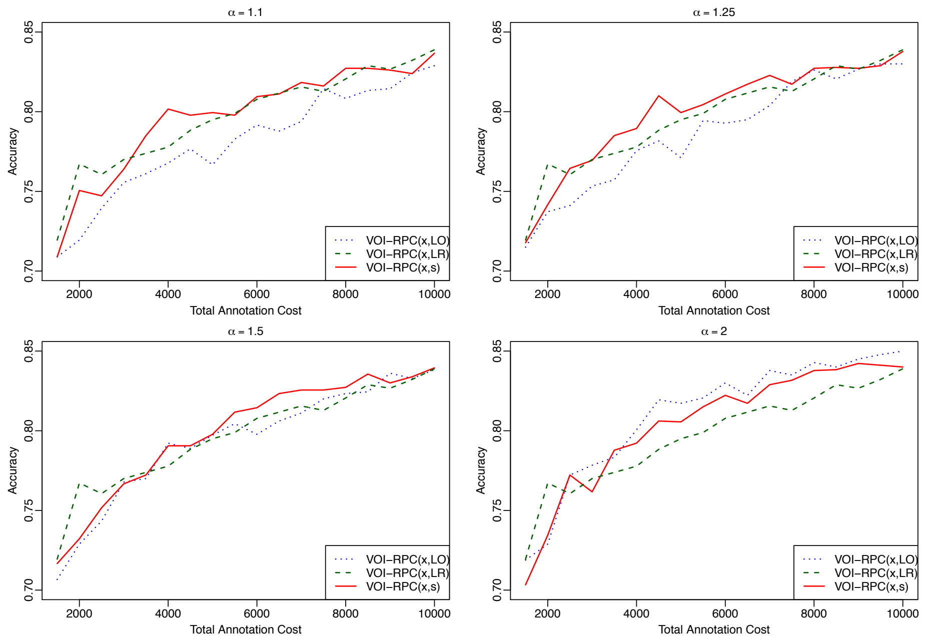
	CostOnly(x,LR)
VOI-RPC(x,LR)	`SSIIIIIIIIIIIIIIII`
VOI-RPC(x,s)	`SSSIMMSSSSSSSMMSSS`

(b) $\alpha = 3$

Figure 3.20: Comparison of VOI with *CostOnly* criteria for instance selection for a fixed strategy (LR or LO) and joint selection of instance and strategy, for MR dataset. Red and solid lines represent joint selection of instance and strategy. Blue and dotted lines represent instance selection for fixed strategy of LO. Green and dashed lines represent instance selection for fixed strategy of LR. Note that *CostOnly(x,s)* is same as *CostOnly(x,LO)* for $\alpha > 1$, that is the expensive strategy of LR will never be selected. Significance strings represent results from 1500 to 10000 (at every 500) cost milestone.

Comparison of Strategy Selection vs. Fixed Strategies for Additional Values of α for MR dataset

So far, we have only compared strategy selection with fixed strategies for two values of $\alpha \in \{1.25, 3\}$. Figure 3.21 compares strategy selection with fixed strategies for several values of $\alpha \leq 2$. Specifically, we consider $\alpha \in \{1.1, 1.25, 1.5, 2\}$, that is $\{10\%, 25\%, 50\%, 100\%\}$ extra cost for annotating rationales. Like before, we initialize the classifiers with 20 instances labeled with rationales.



Comp.	1.1	1.25	1.5	2
LO/LR	ISIIIIISMSSSIISIII	ISIIMIISIIIIIIIIII	ISIIIIIIIIIIIIIIII	ISIIISSSMSISMMMMII
LO/(x,s)	IIIIISISISSSISIIII	IIIISISSISMSIIIIII	IIIIIIIIISIIIIIIII	IIIMIIIIIIIIIIIIIM
LR/(x,s)	IIIIISIIIIIIIIIIII	IMIIIISIIIIIIIIIII	IMIIIIIIIIIIIIIIII	ISIIIISIIIIIIIIIII

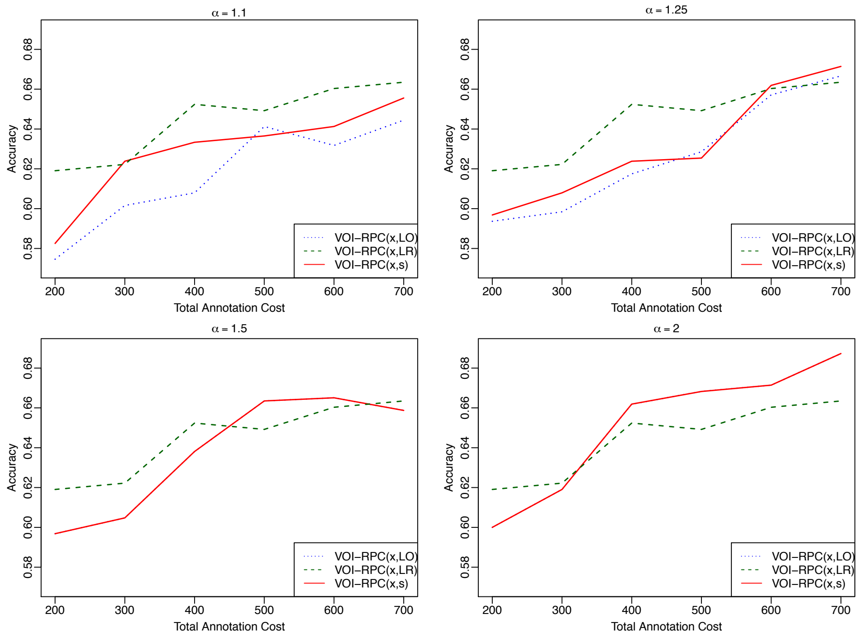
Figure 3.21: Comparison of the two strategies of LO (blue and dotted lines) and LR (green and dashed lines), and strategy selection (red and solid line) for $\alpha \in \{1.1, 1.25, 1.5, 2\}$, for MR dataset. Significance strings summarize results from 1500 to 10000 (at every 500) cost milestone.

As can be seen from Figure 3.21, the best strategy among LO and LR changes as α increases. As observed before, when the additional cost for rationales is 50% or less, getting rationales for all instances (LR) is significantly better than not getting rationales for any (LO). At 100% additional cost, not getting rationales at all is significantly better than getting rationales for all instances, at several points in the learning curve. However, there is significant benefit from rationales, earlier in the learning curve, even at 100% extra cost. Joint selection of instance and strategy is able to follow the best fixed strategy among LO and LR almost always (except at two cost milestones for $\alpha = 2$, where it is marginally worse than LO strategy, although it is significantly better than LR strategy). Joint selection of instance and strategy is worse than fixed strategy of LR at one point in the beginning. This difference is significant for $\alpha = 2$. For selecting LR strategy over LO at $\alpha = 2$, the reduction in model's risk with LR strategy should be more than double that with LO

strategy. As mentioned before, the rationale predictor may not be as accurate in the beginning. Also, risk estimation relies on model's estimate of probability, which may not be accurate in the beginning, after training on only a few labeled instances. Hence, we may underestimate the benefit from rationales in the initial few iterations. Joint selection of instance and strategy performs significantly better than both fixed strategies of LO and LR for $\alpha = 1.1$ and $\alpha = 1.25$, at about half way in the learning curve. This suggests that selectively asking for rationales performs better than asking for rationales for all instances (even with strategy sensitive instance selection) or none.

Comparison of Strategy Selection and Fixed Strategy for ASR dataset

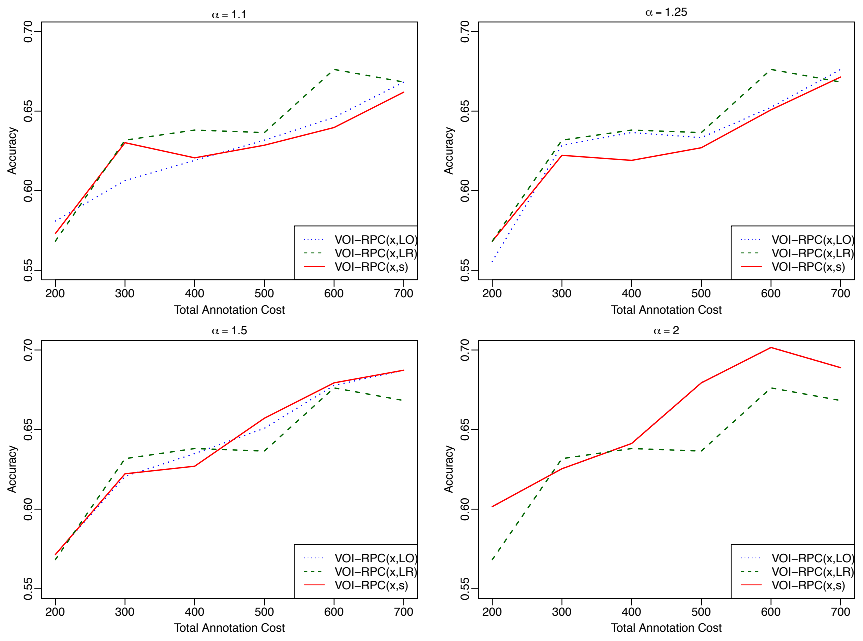
For ASR dataset, since the total dataset is small, we initialize the classifiers with five instances per class, labeled with rationales. Figures 3.22, 3.23 and 3.24 summarize the results for comparison of strategy selection and fixed strategies of LO and LR for three classification tasks in ASR dataset, for $\alpha \in \{1.1, 1.25, 1.5, 2\}$.



Comparison	1.1	1.25	1.5	2
LO/LR	SISIII	MMSIII	IIIIII	IIIIIM
LO/(x,s)	IMSIII	IIIIII	IIIIII	IIIIII
LR/(x,s)	SIIIII	IIMIII	IIIIII	IIIIIM

Figure 3.22: Comparison of the two strategies of LO (blue and dotted lines) and LR (green and dashed lines), and strategy selection (red and solid line) for $\alpha \in \{1.1, 1.25, 1.5, 2\}$, for ASR dataset (PR vs. RD). Significance strings summarize results from 200 to 700 (at every 100) cost milestone.

For PR vs. RD task (Figure 3.22), the fixed strategy of LR is significantly better than LO for up to an additional cost of 25%. At 50% additional cost, the two fixed strategies are not significantly different, and at 100% additional cost, LO is marginally significantly better than LR towards the end of the learning curve. Joint selection of instance and strategy is significantly better than the fixed strategy of instance's label only (LO), when the additional cost for rationales (α) is low. For higher values of α, it is not different from LO, which is better than LR. Joint selection of instance and strategy is better than fixed strategy of instance's label and rationales (LR), when the cost annotating rationales is high. However, for lower values of α, it is worse at a few points in the beginning. As mentioned before, in the initial few iterations, the rationale classifier may not have seen enough examples of rationales in order to accurately predict rationales. Hence, we may be underestimating the benefit from rationales, and may not select the LR strategy. For ASR dataset, particularly for PR and CE classes, the percentage of rationales per instance is smaller than the movie review dataset (Table 3.2), that is there is a higher class imbalance for R vs. NR classifier. One may try to under sample the negative class when training the R vs. NR classifier. Another alternative is to start with a larger initial set labeled with rationales. However, since the total dataset we have for this task is small, starting with a larger initial set would leave us with a shorter learning curve for evaluation. As also mentioned before, we expect some noise in rationale annotations for this dataset. Abedin et al. (2011) did not retain rationales inline, and every occurrence of a rationale sub-string is considered a rationale in the ASR dataset. This, however, may not be true, specifically for short rationale strings. This noise in rationale annotations may also effect the performance of the R vs. NR classifier.

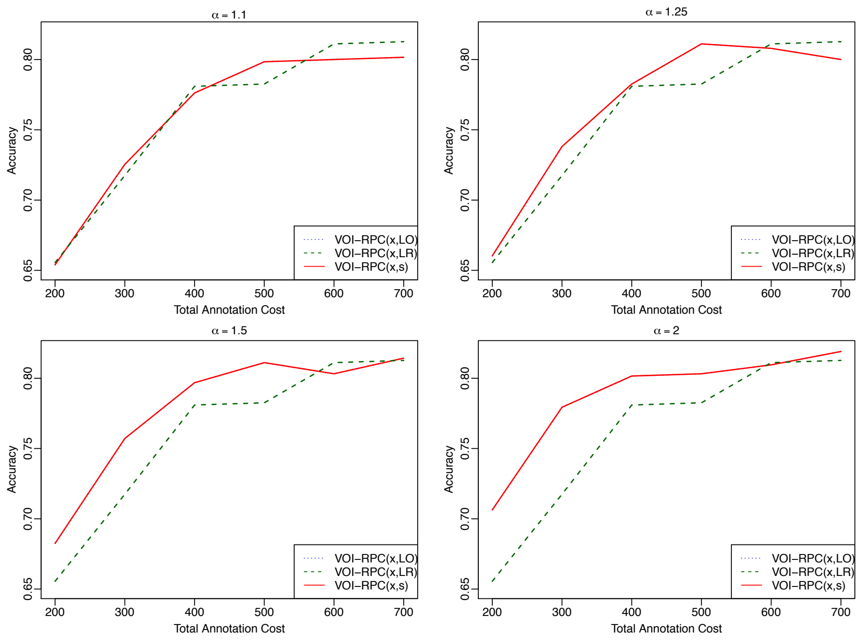


Comparison	1.1	1.25	1.5	2
LO/LR	IIMIMI	IIIIII	IIIIII	MIIMII
LO/(x,s)	IMIIII	IIMIII	IIIIII	IIIIII
LR/(x,s)	IIIISI	IIIIMI	IIIIII	MIIMII

Figure 3.23: Comparison of the two strategies of LO (blue and dotted lines) and LR (green and dashed lines), and strategy selection (red and solid line) for $\alpha \in \{1.1, 1.25, 1.5, 2\}$, for ASR dataset (CE vs. PR).

For CE vs. PR task (Figure 3.23), fixed strategy of LR is marginally significantly better than LO at 10% additional cost. The two strategies are not significantly different at 25% and 50% additional cost. At 100% additional cost, LO is marginally significantly better than LR for both tasks. Joint selection approach is able to follow the best strategy in most cases. For $\alpha = 1.1$, it follows the best strategy initially, but later on it fails to select the appropriate strategy. This may be because accurately estimating the model's risk (in VOI) assumes a large enough unlabeled set. With a small dataset, after several iterations, we are left with a small unlabeled set, and risk estimated on this set may not be an accurate estimate of the model's misclassification risk. For $\alpha = 1.25$, strategy selection is not significantly different from the fixed strategies of LO and LR, except at 400 and 600, where it is marginally worse than the LO and LR respectively. For the other values of α, strategy selection works as well as the best fixed strategy.

For CE vs. RD task (Figure 3.24), as observed before, there is little benefit from rationales. Only significant (or marginally significant) benefit from rationales is for $\alpha = 1.1$ and $\alpha = 1.25$ at the last point in the learning curve. For higher values of α (≥ 1.25), LO is significantly better than LR. Joint selection of instance and strategy follows the LO strategy for the most part.



Comparison	**1.1**	**1.25**	**1.5**	**2**
LO/LR	IIIIIS	IIISIM	ISISII	SSIMII
LO/(x,s)	IIIIII	IIIIII	IIIIII	IIIIII
LR/(x,s)	IIIIIS	IIISIM	ISISII	SSIMII

Figure 3.24: Comparison of the two strategies of LO (blue and dotted lines) and LR (green and dashed lines), and strategy selection (red and solid line) for $\alpha \in \{1.1, 1.25, 1.5, 2\}$, for ASR dataset (CE vs. RD). Significance strings summarize results from 200 to 700 (at every 100) cost milestone.

Thus, for ASR dataset, we see a similar trend as MR dataset. Fixed strategy of LR is better than LO when the cost for annotating rationales is small, and the opposite is true when the cost for annotating rationales is high. Joint selection of instance and strategy follows the best fixed strategy for the most part.

Key Observations: To summarize the results, the key observations are as follows:

- Obtaining rationales for all instances (i.e. fixed strategy of LR) is significantly better than not obtaining rationales at all (i.e. fixed strategy of LO), when the additional cost for annotating rationales is small (up to 50% extra cost). At higher costs for rationales, the fixed strategy of LO is significantly better than the fixed strategy of LR. In other words, the best fixed strategy among LO and LR depends on the cost for annotating rationales, which we call the rationale cost factor (α).

- For the fixed strategy of LR, strategy sensitive instance selection, i.e. selecting an instance for which rationales are expected to bring the most benefit to the model, performs significantly better than strategy independent (random or active) instance selection.

- At high cost for annotating rationales, seeking rationales for all instances performs worse than seeking rationales for none. Cost-sensitive joint selection of instance and strategy automatically determines in each iteration which instance to query and whether to ask for rationales. We find that joint selection of instance and strategy, i.e. selectively asking for rationales, performs as well as or better than the best fixed strategy for a given additional cost for annotating rationales (α), except for a few points in the beginning or towards the end of the learning curve (11 out of 144 points of comparison between strategy selection and best fixed strategy, for two datasets and $\alpha \in \{1.1, 1.25, 1.5, 2\}$, where joint selection is marginally worse than the best fixed strategy at 7 points and significantly worse at 4 points). At a few points it is even significantly better than both fixed strategies. The additional cost for annotating rationales (i.e. α) may vary with the annotator, annotation task, user interface design, etc. We show that in most cases the proposed approach is able to select the appropriate strategy for each scenario, with a given rationale cost factor (α).

- VOI based instance selection is significantly better than the random baseline and common active selection method of uncertainty sampling. VOI criteria that combines reduction in risk and cost together outperforms selection criteria based on one of these scores.

Effect of Number of Rationales

So far, we have only considered two annotation strategies where the annotator provides: 1) instance's $\underline{l}$abel only (LO) or 2) instance's $\underline{l}$abel together with $\underline{r}$ationales (LR). For LR strategy, it is assumed that the annotator provides almost all rationales (as in (Zaidan et al., 2007)). In this section, we evaluate how the benefit from rationales varies with the number of rationales. To understand the effect of number of rationales on the model's performance, we consider several variations of the LR strategy. $LR(N)$ denotes a strategy where we seek N rationales per instance. There can be many ways to ask the annotators for N rationales. We may ask them to provide *any* N rationales, the first ones they spot. Alternatively, we may ask the annotator to provide the *top* N rationales. Annotating each rationale would require some user interface time. Thus, the annotation time should be more when we ask for more rationales. However, if we ask the annotators to provide *top* N rationales, the annotation time may not be less than providing all rationales. This is because the annotators would spend some extra time ranking the rationales by importance in their minds and selecting the top N. The annotation cost may not be a linear function of N when asking for *top* N rationales. Thus, whether asking for a fixed number of rationales allows us to save any time over almost all rationale strategy would depend on the type of annotations we seek from the annotator, that is, whether we ask for *any* N rationales or *top* N rationales, and the value of N.

To understand how the annotation cost varies with the number of rationales, when the number is pre-specified, one should conduct a user study with several annotators, different ways to seek N rationales (*any* N or *top* N) and for different values of N. In this thesis, we present an initial investigation of the effect of number of rationales on model's performance. We consider the scenario of *any* N rationales and run several simulation experiments. We consider several variants of $LR(N)$ strategy where $N \in \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$. We also compare these variants of fixed number of rationales, with LO (i.e. no rationales) and LR strategies (i.e. almost all rationales). We assume a fixed cost for all strategies because how cost varies with N when asking for a fixed number of rationales needs to be understood through a user experiment. Our goal with this experiment is to answer the following question: *Assuming no difference in cost, how does the benefit to the model change with different number of rationales ?* We run this evaluation for random instance selection only, since to select an instance using VOI for a strategy with fixed number of rationales, we would need to modify the R vs. NR classifier which currently assumes that all rationales in an instance are annotated. We suggest some ideas for this extension towards the end of this discussion. For $LR(N)$ strategy, we select the rationales to reveal to the model randomly, to simulate the 'label any N rationales' scenario. We average results over three random selections of rationales (with 3 random seeds for selecting among given rationales in an instance). For an instance that may not have N rationales, we select all the rationales that exist. This is a more realistic scenario as we may not want the annotator to annotate more rationales than there are in an instance. Thus, $LR(N)$ strategy asks the annotator to provide *up to* N rationales in an instance. Note that, we may choose N based on the number of sub-instances in an instance. That is, we may ask the annotator to provide rationales up to 10% or 20% of the sub-instances in an instance. In our initial investigation, we use the same N for all instances. We run this experiment on MR dataset, since there are very few rationales for ASR dataset (Table 3.2). Similar to the comparison of LO and LR strategies in Figure 3.6, we initialize the model with one instance per class.

Figure 3.25 presents the results. As can be seen, we get better performance with more rationales. We see a clear advantage from each additional rationale up to 6 rationales per instance. Strategies $LR(7)$ to $LR(10)$ are indistinguishable in the graph. We get the best performance when we use all rationales available for an instance. Table 3.6, at the end of this chapter, presents the significance results. To evaluate the significance of the observed differences, we fit a standard least squares model (using JMP software[11]) with fold as a random effect and strategy as a fixed effect. As mentioned before, we select N rationales to reveal randomly from the given rationales in an instance. We run three experiments with three different seeds for random selection of rationales to reveal. Thus, we have three results per fold per strategy (for $LR(N)$). We add a variable to uniquely identify a fold and strategy pair. This variable is added as a random effect nested under strategy, since the two are correlated. We use Tukey HSD (Tukey, 1953) to measure significance of pairwise comparisons between multiple strategies. Tukey method is recommended over Student's T method when making multiple comparisons (SasInstitute and SASPublishing, 2008). Tukey's HSD test sets the experiment-wise error rate to the error rate for the collection of all pairwise comparisons. That is, the probability of making one error (observing a significant difference by chance) in the set of pairwise comparisons is equal to 0.05. In other words, the test is adjusted for the number of

[11] `http://www.jmp.com/software/`

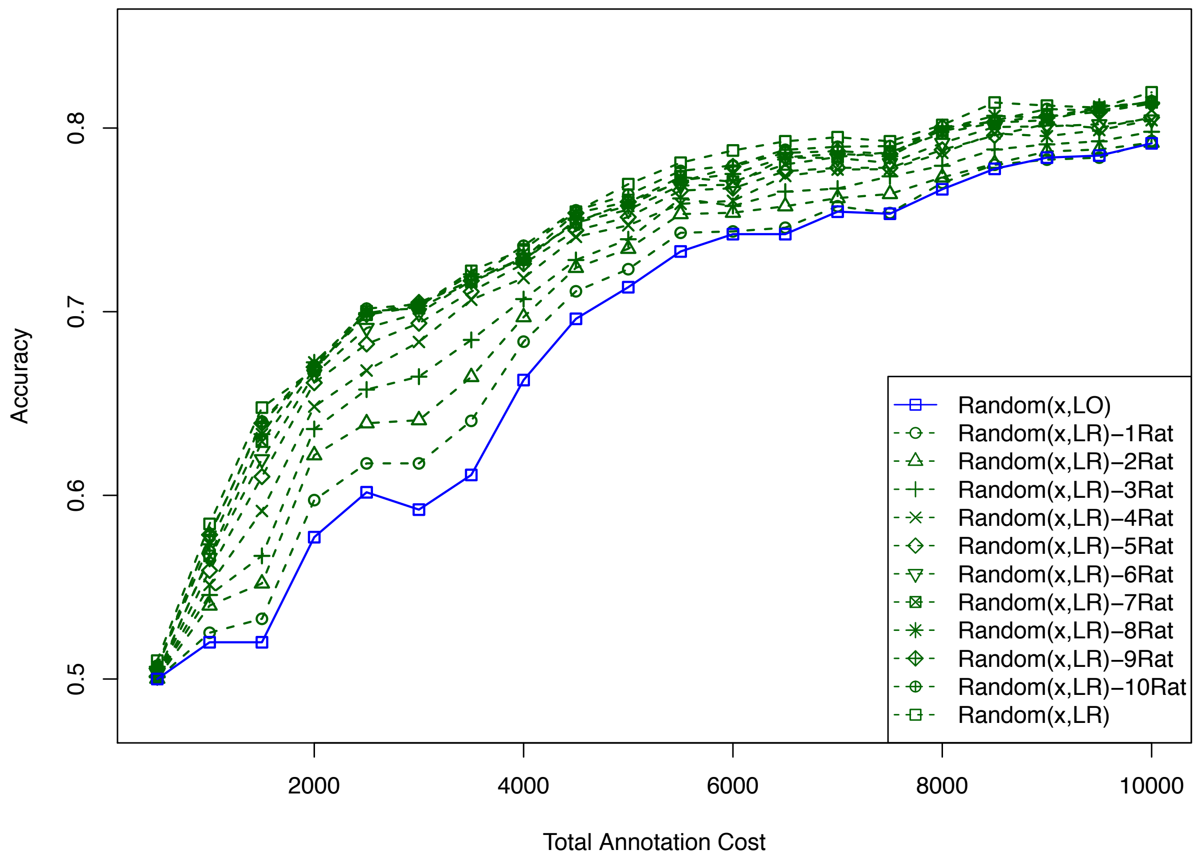


Figure 3.25: Comparison of different number of rationales for random instance selection, for MR dataset.

comparisons made. If we only care to compare two strategies, say LO and $LR(1)$, then we should do a single pairwise t-test.

From Table 3.6, we see that seeking all rationales almost always performs the best and it is never significantly worse than fewer rationales. The performance often increases with more rationales. However, there is no additional significant benefit from more rationales beyond 8 rationales per instance. This may also be because there are on average about 8 rationales per instance in MR dataset (Table 3.2). We found that the median for number of rationales is 7. That is, half of the instances have 7 or fewer rationales. However, we found that the maximum number of rationales in an instance is 30 (in an instance with 67 total sub-instances). Thus, assuming each additional rationale comes with an additional cost, we may want to ask for at most 8 rationales per instance. At higher cost milestones, i.e. later in the learning curve, there is no additional significant benefit from getting more rationales beyond 4 per instance. Thus, we may want to seek more rationales per instance in the beginning and fewer rationales per instance as learning progresses.

Here, we studied the effect of different number of rationales, when the annotator is asked to provide *any* N rationales. Similar simulation experiments can be done for a scenario where the annotator provides the *top* most important rationales. An oracle, a model trained on rationale and not-rationale annotations on a large set, can be used to rank rationales (by the model's confidence) and the *top* N rationales can be revealed for $LR_{top}(N)$ strategy.

The optimum number of rationales per instance may also vary across instances. The current VOI framework can be extended to jointly select an instance and strategy, where we choose the strategy from a set of strategies, one each for a fixed number of rationales. As mentioned in Section 3.3, we estimate the risk after training on instance X_i annotated with strategy $LR(all)$ using predictions for rationales in instance X_i from a model trained (on the data annotated so far) to predict rationales. To estimate the risk after training on instance X_i annotated with strategy $LR_{top}(N)$, i.e. with top N rationales, we may use the *top* N sub-instances predicted to be rationales (ranked by model's confidence). If instead the annotator is asked to provide *any* N rationales ($LR_{any}(N)$), we may estimate the expected risk as the average risk for k random samples for N rationales, from the sub-instances in an instance predicted to be rationales. However, there is one caveat here. In the current VOI framework with almost all rationales strategy, the rationale classifier (R vs. NR) is trained on instances annotated with rationales. Sub-instances in an instance not marked as rationale are considered as not-rationale (NR). However, if all the rationales are not annotated in an instance (for $LR(N)$ strategy), then we cannot treat all sub-instances not annotated as rationale as NR. One simple approach to address this issue is to use the current model's belief about which remaining sub-instances (those that are not marked as rationale) in an instance are likely to be not-rationale and train the classifier with only those labeled as NR.

3.5 Related Work

Alternate forms of feedback from the annotator, in addition to the instance's label, have been explored in the literature. One such form of feedback is direct feedback on features (discussed in detail in Section 3.1), that is, indicating whether a given feature is relevant for the classification task, and what class it supports. However, such feedback is restricted to simple features, such as words, and is often solicited without the context of an instance. Often linguistic features based on dependency triples, parts of speech, etc., are used to represent a text instance (Gamon, 2004; Pradhan et al., 2004; Joshi and Rosé, 2009). Additionally, it may be difficult to judge the relevance of a feature without the context of an instance.

In this thesis, we consider an alternate annotation strategy where the annotator identifies the parts of an instance that are rationales for the instance's label. Rationales have been shown to reduce the total number of labeled instances required to achieve the desired performance for both text and image classification tasks (Zaidan et al., 2007; Arora and Nyberg, 2009; Abedin et al., 2011; Donahue and Grauman, 2011). However, rationales would usually come with an extra cost. The additional cost for annotating rationales is not accounted for in prior work. The additional cost for annotating rationales may vary across instances, annotators, user interface design, etc. We evaluate the benefit from rationales at different additional costs for annotating them. We also propose a cost-sensitive active learning approach for joint selection of instance and strategy. We show that the proposed approach is able to select the appropriate strategy for each annotator and/or user interface, with a given rationale cost factor (α).

Yessenalina et al. (2010) found that rationale sentences automatically generated using a sentiment lexicon or an off-the-shelf opinion analysis tool performed as well as human annotated rationales on the movie review dataset. However, such resources are also developed by human experts and are not available for all domains, for example, aviation safety reports.

Fine-grained annotations have been found to be helpful in learning coarse-grained annotations. For example, Bennett and Carbonell (2005) found that for classifying an email as containing an action-item or not, a sentence classifier trained to predict sentence labels that are then agglomerated to predict the email label, outperforms an email classifier. Settles et al. (2008b) show that instance (sentence) labels in addition to bag (document) labels can be used to improve performance of a multiple instance learner (performance on bags/documents). However, in above scenarios, sentence annotations are assumed to be independent of the document. That is, if a sentence occurs in a different document, it will still have the same label. On the other hand, rationale annotations are specific to an instance. A sub-instance may not be a rationale for every instance it occurs in.

Cost-sensitive active learning with multiple annotation strategies has received recent attention in the literature. Vijayanarasimhan and Grauman (2011) use the VOI criterion in (Kapoor et al., 2007) that considers the model's risk on both labeled and unlabeled data, to select between bag and instance annotations for a multiple-instance learning task. Attenberg et al. (2010) use utility per unit cost to select between instance and feature annotation. The utility is measured in terms of the log gain, defined as the sum of model's confidence on true label for the training instances. Since our goal is to improve the model's accuracy at predicting the label for an unseen test instance, reduction in risk on unlabeled data is a more relevant criterion for active selection. Apart from the reduction in risk, other measures of generalization error may be used in the VOI framework, for example, 0/1 loss or log loss (Roy and Mccallum, 2001). To the best of our knowledge, ours is the first work where the model actively decides which instances to query and whether to ask for rationales.

In the literature (Melville et al., 2005; Kapoor et al., 2007; Donmez and Carbonell, 2008b; Haertel et al., 2008; Vijayanarasimhan and Grauman, 2011), one of the VOI definitions presented in Section 3.3 has been used to combine utility and cost scores for active learning. To the best of our knowledge, no prior work except (Tomanek and Hahn, 2010) has tried to compare more than one way to combine utility and cost scores. In their work, they consider two approaches: 1) *Benefit to Cost Ratio* (BCR), where *benefit* is a measure of model's uncertainty on the candidate instance, and *cost* is the cost of labeling that instance[12], and 2) *Linear Combination of Ranks* (LRK) for the candidates based on the uncertainty and cost scores. The task in their work is Named Entity Recognition, and they consider two ways to infer the model's uncertainty for a sentence, based on the confidence of the model on the classification of tokens in the sentence. For one of them, they find that LRK and BCR are not different. For the other one, they find that LRK is better than BCR, although the difference seems small and they do not report significance results. As mentioned before, the difference between scores for candidates with the same ranks based on the two criteria, may be quite different. This difference is not accounted for when combining the ranks, as in (Tomanek and Hahn, 2010). Nonetheless, a future direction is to compare VOI based approach with combinations of ranks approach, where ranks are determined based on utility and cost measures used in this work.

In this work, we measure the utility of a candidate in terms of the reduction in model's misclassification risk. An alternative is to measure the utility in terms of model's uncertainty on the candidate instance, as in (Donmez and Carbonell, 2008b; Tomanek and Hahn, 2010). Another alternative is to measure the utility of a candidate instance in terms of how representative it is of other instances. For example, Donmez and Carbonell (2008a) use a density-based sampling approach, where an instance representative of a cluster of similar instances is selected. However,

[12]BCR is similar to RPC function with different measures for utility and cost.

note that these measures of utility cannot distinguish between the two strategies. If we combine them directly with cost, then LR strategy will never be selected for $\alpha > 1$. In our results, we demonstrated that VOI based instance selection performs better than uncertainty based instance selection for a fixed strategy of LO and LR.

3.6 Summary and Future Work

In order to reduce the total annotation cost for the desired performance, alternative forms of feedback, such as feedback on features, in addition to the instance's label have been proposed in the literature. However, direct feedback is limited to simple features, such as unigrams. In this Chapter, we proposed an alternate annotation strategy where in addition to the instance's label, the annotator indicates parts of an instance that are rationales for its label. In order to determine the label for an instance, the annotator perhaps already makes this distinction. We ask them to provide their reason to assign the indicated label to an instance, in form of rationales. Rationales provide indirect feedback on features, since features that overlap with rationales should be important for the classification task and this indication is only indirect. While direct feature feedback has been modeled in prior work as a separate task from labeling instances, rationales are solicited together with the instance's label.

Zaidan et al. (2007) seek rationales as spans of text in a document. In this work, we seek rationales as votes on sub-instances in an instance. Highlighting spans of text as rationales should require more annotation time than voting on sub-instances. Additionally, exact spans for rationales may vary across annotators, while we can expect to see more agreement between the annotators for rationales as sub-instances in an instance. Zaidan et al. (2007) found that it takes twice as much time to annotate rationales as spans of text in addition to providing the instance's label. We expect a lower cost for voting on sub-instances in an instance as rationales. The additional cost for rationales may depend on several factors, such as the user interface design, annotation task, annotator, etc. Thus, we compare the two strategies of instance's label only (LO) and instance's label together with rationales (LR) at several additional costs for annotating rationales. Note that in prior work, the additional cost for annotating rationales is not accounted for when comparing the two strategies of LO and LR (Zaidan et al., 2007; Arora and Nyberg, 2009; Abedin et al., 2011; Donahue and Grauman, 2011). From the results presented in this chapter, we can conclude that rationales (as sub-instances in an instance) in addition to the instance's label provide a significant improvement in performance for a given annotation cost, when the cost for annotating rationales is small (up to 50% extra cost for annotating rationales in addition to the instance's label).

Instances (with or without rationales) may provide different incremental value to the learning algorithm. Thus, we propose a cost-sensitive active selection criteria for actively selecting an instance for a given strategy. We showed that strategy sensitive instance selection performs better than strategy insensitive instance selection. That is, seeking rationales for instances expected to bring the most benefit to the model from rationale annotations, performs better than seeking rationales for instances selected randomly or actively (using VOI or uncertainty) independent of the strategy. When the cost for annotating rationales is high, rationales may not be useful for all instances, and we may want to selectively ask for rationales. We further enhance the proposed approach to jointly select the best instance and strategy in each iteration. While the best fixed strategy among LO and LR varies with the cost for annotating rationales, we showed that except for a few cases, the proposed approach performs as well as or better than the best fixed strategy for a given cost for annotating rationales. We also showed that the proposed approach performs better than cost-sensitive (VOI) and cost-insensitive (random, uncertainty and risk-only) instance selection for a fixed strategy, and cost-insensitive (risk-only) joint selection of instance and strategy.

We studied three variants for incorporating cost in the active selection criteria (VOI), inspired by the prior work in the literature: 1) A decision theoretic framework proposed in (Kapoor et al., 2007), and applied to a similar scenario as ours in active learning (Vijayanarasimhan and Grauman, 2011), where VOI is computed as the reduction in total cost of the use of the classifier minus the annotation cost for the candidate (VOI-RTC), 2) Difference in standardized scores for reduction in model's risk and annotation cost (VOI-SS), and 3) Reduction in model's expected risk on an unseen instance per unit annotation cost (VOI-RPC). For a fixed strategy, we found little difference between the variants of VOI. For joint selection of instance and strategy, we found that for a high rationale cost factor ($\alpha = 3$), difference in standardized scores for reduction in risk and cost is significantly worse than the other two approaches, and it also performs worse than the best fixed strategy among LR and LO. A future direction is to consider a weighted combination of the standardized scores.

In this work, we used the contrast constraint approach in (Zaidan et al., 2007) for incorporating information about rationales in the learning algorithm. We proposed a modification to their approach for constructing contrast instances, which provided us a significant improvement in performance. In Section 3.2, we suggested other ways to improve on their approach, such as constructing a single contrast instance from all rationales together, or constructing several from a subset of rationales. The proposed active learning approach is general and can also be applied to other methods for incorporating information about rationales.

Next, we discuss the following two points: 1) how to choose an appropriate granularity for rationales for a new learning task, and 2) for what learning problems we expect rationales to be useful.

Rationale Granularity: There are several choices for granularity of rationales for an annotation task. For a text instance of document length, rationales may be annotated as highlighted sub-strings (as in (Zaidan et al., 2007)), or as votes on sentences in a document (as in this work). Rationales may also be annotated as votes on phrases in a sentence. We can determine the phrases in a sentence using a syntactic parser. Alternatively, in a long document, rationales may be annotated as votes on paragraphs. The appropriate granularity for rationales for a learning task will depend on several factors. There are two main factors to consider:

- Votability: As already discussed earlier in this chapter, there are several advantages of annotating rationales as votes on sub-instances in an instance over rationales as highlighted spans or regions. However, even when rationales are annotated as votes on sub-instances in an instance, there are several choices for rationale granularity. For example, in a document, a rationale can be a phrase, sentence, or a paragraph. For a given domain and classification task, an important factor to consider in deciding the appropriate granularity for rationales is that is should be *votable*. That is, the annotator should be able to determine whether the given segment of text (phrase, sentence or paragraph) by itself is a rationale or not. In some cases, a phrase may not be a rationale by itself without the context of the sentence it is part of. In this case, a sentence would be a more appropriate unit of rationale. While sentences can be thought of as independent units of judgement for rationales, in some domains it may be the case that a sentence is a rationale only in context of the paragraph it is part of. In this case, a paragraph may be a more appropriate unit of rationale. Thus, depending on the domain and text instance length, we should decide an appropriate granularity for rationales, based on its *votability*.

- Value: Another factor to consider in deciding the granularity for rationales is the *value* they provide to the model. Allowing the annotators to determine the span for rationales would provide us more precise rationales. The entire span of a rationale is used when incorporating information about rationales into the model, at least in the contrast constraint approach (Zaidan et al., 2007) that we adopt in this work. If the spans for rationales are predetermined as phrases, sentences, or paragraphs, then rationales would be less precise. As mentioned before, rationales provide indirect feedback on features. Features that overlap with rationales should be important for the classification task and this indication is only indirect. The contrast constraint approach (Zaidan et al., 2007) also indirectly influences the weights that are learned for the features present in rationales. If rationales are less precise, we may be putting unnecessary emphasize on some of the features. Preciseness of a given granularity for rationales may vary across learning tasks. For the movie review classification task (Zaidan et al., 2007) with unigram features, Yessenalina et al. (2010) found that there is no significant difference in performance between rationales as sub-strings or sentences in a document (by extending sub-string rationales to sentence boundary). However, this may not be true for all tasks and all feature sets. Also, this may not be true for a different approach to incorporate information about rationales.

Thus, for a new learning task, we must take into account the above factors when deciding the granularity for rationales. One approach to determine the appropriate granularity for rationales for a new task is to conduct a pilot user study to evaluate different granularities for rationales in terms of their *votability*, and analyze their effect on model's performance to evaluate their *value*.

In the current approach, all information in a rationale is treated equally. As mentioned before, when rationales are annotated as votes on pre-segmented spans of text, such as sentences, the rationale information is not precise. We may want to treat parts of a rationale (e.g. phrases in a sentence) differently when incorporating information about rationales. For a sentiment classification task, adjective phrases may be more relevant than noun phrases or verb phrases. These phrases can be determined using a syntactic parser. For the contrast constraint approach for incorporating information about rationales (Zaidan et al., 2007), we may use only parts of a rationale that we expect to be most relevant, to construct a contrast instance. For example, for a sentiment classification task with rationales as sentences in a document, we may construct a contrast instance only from the adjective phrases in a sentence.

Benefit from Rationales: We expect rationales to be useful when some parts of an instance are strong indicators and others are weak indicators of the instance's label. For example, in a movie review, the reviewer often talks about the movie plot, actors, etc., which may not be strong indicators of the reviewer's sentiment, and hence may not be relevant for classifying the review as expressing positive or negative sentiment. Similarly when classifying the scene in an image as that of a kitchen, parts of the image that show a sink, fridge etc., might be key indicators, and parts that show the floor, may not be indicative of the kitchen. On the other hand, product reviews are often shorter than movie reviews, and are often more concise. We expect rationales to be relatively less useful when classifying the sentiment of a product review, compared to a movie review.

In this work, we saw that there is more benefit from rationales earlier on in learning. This is intuitive, as the model matures we may not benefit as much from additional supervision. If labels for a large number of instances can be obtained relatively fast and for cheap, for example, through crowd sourcing sites, such as Amazon's Mechanical Turk[13], then rationales may not be as useful, since we saw a little benefit from rationales when we have a large amount of labeled data. However, annotators in such crowd sourcing sites are non-experts, and annotations are often noisy. For tasks, such as aviation safety report classification, or clinical document and image classification, we may

[13]`https://www.mturk.com/mturk/welcome`

require annotations from domain experts who are often expensive to hire. For certain tasks in a medical domain, it may be costly to even obtain unlabeled data, as it may require several pre-processing steps, such as anonymizing the data, or there may only be a limited amount of data that matches a given criteria. Rationale annotations are expected to be useful for such tasks.

In this thesis, we defined rationales as votes on sub-instances in an instance. Text, image and video instances can be segmented into sub-instances, such as sentences, regions, frames, etc. However, if the span to classify is considerably smaller (e.g. a small text phrase) and there are only a few sub-instances in an instance, most of which are relevant, then rationales may not provide much additional information. For example, when classifying movie review snippets (about a sentence length), in comparison to movie review documents, we can expect rationales to be less useful as snippets are more concise and there would only be one or two sub-instances in an instance, most of which would be relevant. In Chapter 2, we studied the problem of classifying sentences as suggesting that two proteins mentioned in it interact. Rationales as sentences will not be useful here, since an instance contains only a single sentence. Even if we consider rationales as phrases in a sentence, there would only be a few phrases in a sentence, most of which would be relevant. On the other hand, if we were classifying biomedical abstracts or papers as suggesting whether two proteins mentioned in them interact, then most likely this information would be indicated by one or two sentences in the whole abstract or document. Information about what sentences among many are indicators of the protein-protein interaction should certainly be helpful to the model. In Chapter 5, we study the effect of instance granularity and the amount of non-rationale (or irrelevant) text on the benefit from rationales.

Figure 3.26 shows some example learning problems (most of which we have studied in this thesis) in a 3D space of amount of irrelevant text (called slack), annotator expertise required for the annotation task, and the benefit from rationales. Overall, we expect more benefit from rationales for learning problems that have more irrelevant text in its instances and require annotation from experts that are expensive to hire.

In the work presented in this chapter, we only experimented with the unigram features. We expect the benefit from rationales to be more when the features are not hand-crafted and the feature space is large, for example, when other linguistic features are used in addition to the unigram features. Rationales provide indirect feedback on features. However, the above intuition about which learning problems may benefit from rationales, can also be applied to direct feedback on features in context of the instance. In Chapter 5, we present a formal analysis of how benefit from feature feedback varies across different learning problems, and what characteristics of a learning problem have a significant effect on the benefit from feature feedback. We also study measures that can be used to categorize learning problems, and that have a strong correlation with the amount of benefit from feature feedback.

There are several ways in which the work presented in this chapter can be extended. We suggest some of the extensions here:

- **Multiple annotation strategies with different number of rationales**: In this work and prior work with rationales, it is assumed that the annotator provides almost all rationales for an instance. We may not need all rationales for every instance. Instead, we may ask the annotators to provide a fixed number of rationales. There can be many different ways to ask the annotators for a fixed number of rationales per instance. For example, we may ask them to provide *any* N rationales or the *top* most important N rationales. While the annotation cost should be less for smaller N when asking for *any* N rationales, it is not clear if annotating *top* N rationales would require less time than annotating almost all rationales. This is because

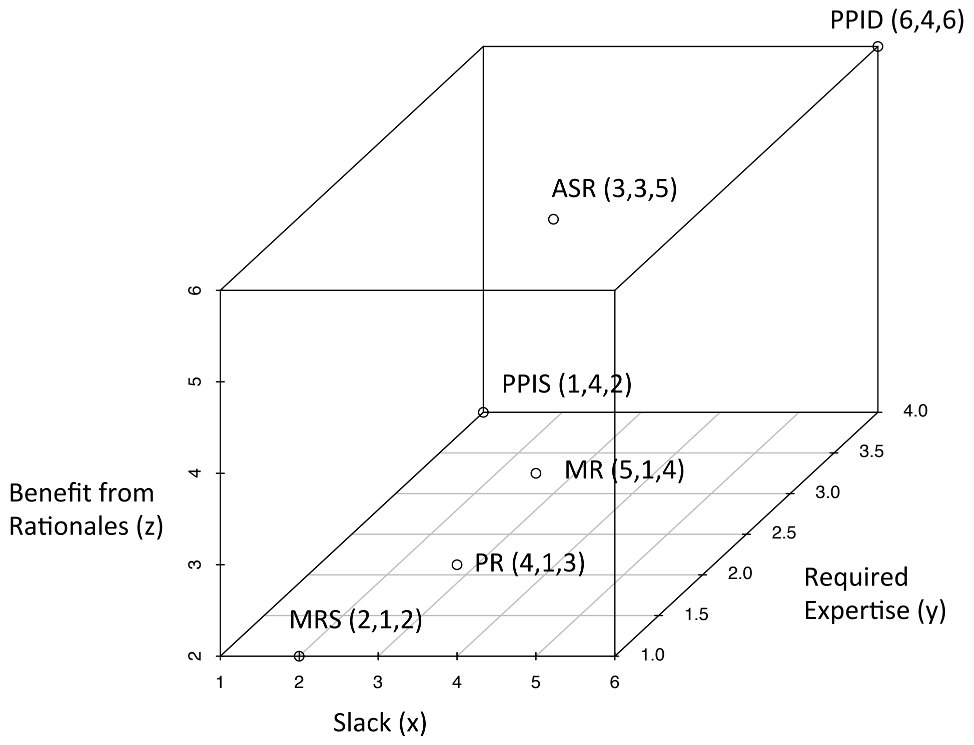


Figure 3.26: Notional problem space illustrating what problems are expected to benefit more or less from rationales. X-axis represents the slack, i.e., the average amount of irrelevant text in an instance; Y-axis represents the annotator expertise required for the annotation task; Z-axis represents the benefit from rationales. Learning Problems: MRS - Movie Review Snippets, MR - Movie Reviews, PR - Product Reviews, ASR - Aviation Safety Reports, PPIS - Protein-Protein Interaction extraction from Sentences, PPID - Protein-Protein Interaction extraction from Documents (e.g. research papers). The numbers on the axes do not indicate actual values. The values shown represent the relative ranks of problems in each dimension.

to determine the top N rationales, the annotators would need to rank the rationales (in their minds) in the order of importance. How annotation time varies across strategies with pre-defined number of rationales requires further investigation through user experiments. In this chapter, we presented an initial investigation of how the benefit to the model varies with the number of rationales (for *any* N rationales strategy), assuming the same cost for all strategies. For a movie review classification task, for random instance selection, we find that performance usually increases with more rationales. However, there is no additional significant benefit from more rationales beyond 8 rationales per instance. Thus, if each additional rationale comes with an additional cost, we may want to ask for fewer rationales. As learning progresses, we may want to ask for fewer rationales, since we found little significant benefit from getting more rationales later in the learning curve. The optimum number of rationales may also vary across instances. In Section 3.4.2, we suggested how we may extend the proposed cost-sensitive active learning framework to select from multiple annotation strategies, where in addition to deciding whether or not to seek rationales, it also determines how many rationales to ask for in a given instance.

- **Annotation cost estimation:** For cost-sensitive active selection, we assumed that the additional cost for annotating rationales (i.e. the rationale cost factor) is known and fixed. The rationale cost factor may be estimated for a given annotator and user interface design from a small sample of data labeled with both strategies. This sample can be the same as the initial set that is used to bootstrap the active learner. The estimate for rationale cost factor can also be updated as we collect more labeled data with annotation time. If the labeled data collected in each iteration is also used to update the annotation cost model, and the annotation cost estimate is used for cost-sensitive active selection, then we should select the instance and strategy that also improves our cost model in addition to our model for the target annotations. That is, we may want to also consider the reduction in risk of our annotation cost estimator, in addition to the model for the target annotations, when selecting the next candidate for annotation. We discuss this extension to our framework in further detail in Chapter 6.

  In this chapter, we used a simple estimate of annotation cost based on instance length and strategy. Annotation cost may also depend on other characteristics of the instance, annotation strategy, and annotator. For example, the cost would be more when more rationales are annotated. In our current cost model, the absolute difference between the costs for two strategies is more for a longer document, than a shorter one. For example, consider two instances X_i and X_j with 4 and 8 sub-instances respectively. The cost for two strategies for these two instances are $C(X_i, LR) = 4$, $C(X_j, LR) = 8$, $C(X_i, LO) = 2$, and $C(X_j, LO) = 4$. Since a longer document is expected to contain more relations, we take into account the effect of number of rationales on annotation time to some extent. However, as we will see Chapter 4, number of rationales only has a low correlation (0.24) with the length of a document. Two documents with same length may contain different number of rationales, and hence the cost for annotating them would be different. In Chapter 4, we propose a more sophisticated supervised regression model for estimating the annotation cost in a multiple annotator environment with rationale annotations. One of the characteristics of the annotation strategy that we use as a feature in our model is the number of rationales. However, when calculating VOI for LR strategy with almost all rationales, we would not know *a priori* how many rationales will be annotated. However, we may use an estimate for it from the current R vs. NR classifier.

  Annotation cost may also vary across annotators, based on whether or not they are native speaker of the language for annotation, their experience with the annotation task, etc. In Chapter 4, for data collected from multiple annotators with rationale annotations, we analyze how the annotation time varies across annotators. We use annotator nativeness as one of the features in our model for annotation cost estimation, and we found that it provides an improvement in performance. Annotation cost may also vary over time. As the annotator becomes familiar with the task, their annotation speed may increase. On the other hand, after labeling several instances, their annotation speed may decrease due to fatigue. In Chapter 4, we suggest some features to include in the estimator to model the effect of time.

  For a new learning task, we should conduct a pilot user study with multiple annotators and instances with different characteristics to understand which of the above factors have a significant effect on the annotation time. In Chapter 4, we present a similar analysis for a movie review classification task with data collected from multiple annotators with rationale annotations.

- **Enhancement to the utility estimate:** In this work, we only considered one way to approximately calculate the expected risk of the model after labeling an instance with rationales. As discussed in Section 3.3, we used the predictions for rationales from a rationale vs. not-rationale (R vs. NR) classifier for sub-instances. For this model, we used unigram features to represent a sub-instance. For the movie reviews, we observed that there were often more rationales at the end of a review than in the beginning. A reviewer would often start off with a discussion about the movie plot, ending the review with his/her opinion. Position of a sub-instance in an instance may be used as a feature in the rationale classifier. In our model, we made an assumption that labels for sub-instances in an instance are independent of each other, given the instance's label. An alternative approach is to model it as a sequence labeling task, where the label for a sub-instance is predicted taking into account its context in the instance. Vijayanarasimhan and Grauman (2011) use a Gibbs sampling approach for estimating the risk with predicted labels for instances in a bag, for a multiple instance learner. Label for an instance in a bag is sampled from the conditional distribution of one instance's label, given the rest. In Gibbs Sampling, for each sample, the classifier is retrained with the given labels for the remaining instances and a label is drawn for the instance under consideration. The expected risk is calculated as the average risk for a large number of samples. This method relaxes the independence assumption we make, however it requires several learn/unlearn operations, and increases the computation cost of the model substantially.

There are also other ways to use the output of the rationale classifier. Since we expect the classifier to be less accurate in the beginning when it has only been trained on rationales for a few instances, instead of using the predictions from the R vs. NR classifier as is, we may sample a label for a sub-instance from the model's probability distribution for the sub-instance's label. That is, if $p^r_{i,j}$ is the model's confidence that a sub-instance $x_{i,j}$ is a rationale in an instance x_i, we may sample a label for $x_{i,j}$ from $bernoulli(p^r_{i,j})$ distribution.

Accurate estimation of the risk for LR strategy relies on the accuracy of the R vs. NR classifier, which is trained on the data annotated so far. For active strategy selection, if the reduction in risk for LR strategy is underestimated, it may not be selected. If the LR strategy is not selected, the rationale classifier is not updated. Thus, we may want to take into account the uncertainty of the R vs. NR classifier when computing VOI for strategy selection. One simple approach to do this is the following. Instead of using the predictions from R vs. NR classifier as is, we instead sample a label (R or NR) for a sub-instance from the model's probability distribution for the sub-instance's label, as suggested above. However, instead of sampling once, we may draw several samples for labels for sub-instances in an instance, and calculate the expected risk for each sample. From the expected risk for these samples, we can calculate the mean and standard deviation of the expected risk, and use the lower confidence interval of the risk in VOI estimation, calculated as follows:

$$LI(\hat{\mathcal{R}}_{\mathcal{X}_l^{t+1}((X_i,Y_i),LR)}(\mathcal{X}_u^{t+1}(X_i))) = \\ m(\hat{\mathcal{R}}_{\mathcal{X}_l^{t+1}((X_i,Y_i),LR)}(\mathcal{X}_u^{t+1}(X_i))) - t^{n-1}_{\alpha/2}\frac{s(\hat{\mathcal{R}}_{\mathcal{X}_l^{t+1}((X_i,Y_i),LR)}(\mathcal{X}_u^{t+1}(X_i)))}{\sqrt{n}} \tag{3.12}$$

where $\hat{\mathcal{R}}_{\mathcal{X}_l^{t+1}((X_i,Y_i),LR)}(\mathcal{X}_u^{t+1}(X_i))$ is an estimate for the risk after training on instance X_i annotated with strategy LR (as discussed in Section 3.3), $m(.)$ and $s(.)$ are the sample mean and standard deviation of the risk for the given samples. n is the number of samples for rationales in an instance, and $t^{n-1}_{\alpha/2}$ is the critical value for the Student's t-distribution with $n-1$ degrees of freedom, at $\alpha/2$ confidence level. Using the lower confidence interval for risk for *LR* strategy, we may select *LR* strategy over *LO* when it has lower average risk or when there is high uncertainty in its risk estimate, due to uncertainty of the rationale classifier. This idea is inspired by the work in (Donmez et al., 2009), where the authors use the upper confidence interval as an estimate for annotator's accuracy to select the annotators with high reliability or high uncertainty in their reliability estimate.

Cost	Strategy	Significance	Accuracy	Cost	Strategy	Significance	Accuracy
500	R(x,LR(All))	A	0.510	1000	R(x,LR(All))	A	0.584
	R(x,LR(10))	A B	0.507		R(x,LR(9))	A B	0.578
	R(x,LR(8))	A B C	0.507		R(x,LR(10))	A B	0.578
	R(x,LR(9))	A B C	0.506		R(x,LR(8))	A B C	0.573
	R(x,LR(6))	A B C	0.503		R(x,LR(7))	A B C D	0.567
	R(x,LR(7))	B C	0.502		R(x,LR(6))	A B C D	0.566
	R(x,LR(5))	B C	0.501		R(x,LR(5))	A B C D	0.559
	R(x,LR(3))	B C	0.501		R(x,LR(4))	B C D E	0.551
	R(x,LR(4))	B C	0.501		R(x,LR(3))	C D E	0.546
	R(x,LR(2))	C	0.500		R(x,LR(2))	D E	0.540
	R(x,LR(1))	C	0.500		R(x,LR(1))	E	0.525
	R(x,LO)	B C	0.500		R(x,LO)	E	0.520
1500	R(x,LR(All))	A	0.648	2000	R(x,LR(8))	A	0.672
	R(x,LR(10))	A	0.640		R(x,LR(10))	A	0.670
	R(x,LR(9))	A	0.639		R(x,LR(9))	A	0.670
	R(x,LR(8))	A	0.633		R(x,LR(7))	A	0.669
	R(x,LR(7))	A	0.629		R(x,LR(All))	A	0.668
	R(x,LR(6))	A B	0.620		R(x,LR(6))	A	0.666
	R(x,LR(5))	A B	0.610		R(x,LR(5))	A	0.661
	R(x,LR(4))	B C	0.591		R(x,LR(4))	A B	0.648
	R(x,LR(3))	C D	0.567		R(x,LR(3))	A B C	0.636
	R(x,LR(2))	D E	0.552		R(x,LR(2))	B C	0.622
	R(x,LR(1))	D E	0.533		R(x,LR(1))	C D	0.597
	R(x,LO)	E	0.520		R(x,LO)	D	0.577
2500	R(x,LR(10))	A	0.702	3000	R(x,LR(9))	A	0.705
	R(x,LR(7))	A	0.700		R(x,LR(10))	A	0.704
	R(x,LR(8))	A B	0.699		R(x,LR(8))	A	0.703
	R(x,LR(9))	A B	0.698		R(x,LR(All))	A B	0.703
	R(x,LR(All))	A B	0.698		R(x,LR(7))	A B	0.702
	R(x,LR(6))	A B	0.691		R(x,LR(6))	A B	0.699
	R(x,LR(5))	A B	0.682		R(x,LR(5))	A B	0.694
	R(x,LR(4))	A B C	0.668		R(x,LR(4))	A B	0.683
	R(x,LR(3))	B C D	0.657		R(x,LR(3))	B C	0.665
	R(x,LR(2))	C D E	0.639		R(x,LR(2))	C D	0.641
	R(x,LR(1))	D E	0.617		R(x,LR(1))	D E	0.617
	R(x,LO)	E	0.602		R(x,LO)	E	0.592

Continued on next page

Table 3.5 – continued from previous page

Cost	Strategy	Significance	Accuracy	Cost	Strategy	Significance	Accuracy
3500	R(x,LR(All))	A	0.722	4000	R(x,LR(10))	A	0.736
	R(x,LR(8))	A	0.720		R(x,LR(All))	A B	0.734
	R(x,LR(10))	A	0.717		R(x,LR(6))	A B	0.729
	R(x,LR(9))	A	0.717		R(x,LR(9))	A B	0.729
	R(x,LR(6))	A	0.717		R(x,LR(7))	A B	0.728
	R(x,LR(7))	A	0.716		R(x,LR(8))	A B	0.727
	R(x,LR(5))	A	0.711		R(x,LR(5))	A B	0.726
	R(x,LR(4))	A	0.706		R(x,LR(4))	A B C	0.718
	R(x,LR(3))	A B	0.685		R(x,LR(3))	A B C	0.707
	R(x,LR(2))	B C	0.664		R(x,LR(2))	B C D	0.697
	R(x,LR(1))	C D	0.640		R(x,LR(1))	C D	0.684
	R(x,LO)	D	0.611		R(x,LO)	D	0.663
4500	R(x,LR(10))	A	0.755	5000	R(x,LR(All))	A	0.769
	R(x,LR(All))	A	0.754		R(x,LR(10))	A	0.764
	R(x,LR(9))	A	0.754		R(x,LR(9))	A B	0.760
	R(x,LR(8))	A	0.750		R(x,LR(7))	A B	0.75
	R(x,LR(7))	A	0.749		R(x,LR(8))	A B	0.758
	R(x,LR(6))	A	0.748		R(x,LR(6))	A B	0.755
	R(x,LR(5))	A B	0.744		R(x,LR(5))	A B	0.751
	R(x,LR(4))	A B	0.741		R(x,LR(4))	A B C	0.747
	R(x,LR(3))	A B C	0.728		R(x,LR(3))	A B C D	0.739
	R(x,LR(2))	A B C	0.724		R(x,LR(2))	B C D	0.734
	R(x,LR(1))	B C	0.711		R(x,LR(1))	C D	0.723
	R(x,LO)	C	0.696		R(x,LO)	D	0.713
5500	R(x,LR(All))	A	0.781	6000	R(x,LR(All))	A	0.788
	R(x,LR(10))	A	0.777		R(x,LR(10))	A B	0.780
	R(x,LR(7))	A B	0.773		R(x,LR(9))	A B	0.779
	R(x,LR(8))	A B	0.771		R(x,LR(8))	A B	0.775
	R(x,LR(9))	A B	0.770		R(x,LR(7))	A B C	0.771
	R(x,LR(6))	A B	0.769		R(x,LR(6))	A B C D	0.769
	R(x,LR(5))	A B	0.766		R(x,LR(5))	A B C D	0.767
	R(x,LR(3))	A B C	0.762		R(x,LR(4))	A B C D	0.760
	R(x,LR(4))	A B C D	0.759		R(x,LR(3))	A B C D	0.757
	R(x,LR(2))	B C D	0.753		R(x,LR(2))	B C D	0.754
	R(x,LR(1))	C D	0.743		R(x,LR(1))	D	0.744
	R(x,LO)	D	0.733		R(x,LO)	C D	0.742
6500	R(x,LR(All))	A	0.793	7000	R(x,LR(All))	A	0.795
	R(x,LR(10))	A	0.788		R(x,LR(10))	A	0.790
	R(x,LR(9))	A	0.786		R(x,LR(9))	A B	0.787
	R(x,LR(8))	A	0.784		R(x,LR(8))	A B	0.786
	R(x,LR(6))	A	0.784		R(x,LR(7))	A B	0.784
	R(x,LR(7))	A B	0.780		R(x,LR(6))	A B	0.784
	R(x,LR(5))	A B	0.776		R(x,LR(5))	A B C	0.778
	R(x,LR(4))	A B	0.774		R(x,LR(4))	A B C D	0.777
	R(x,LR(3))	A B C	0.765		R(x,LR(3))	B C D	0.767
	R(x,LR(2))	B C	0.757		R(x,LR(2))	C D	0.762
	R(x,LR(1))	C	0.746		R(x,LR(1))	D	0.757
	R(x,LO)	C	0.742		R(x,LO)	C D	0.754

Continued on next page

Table 3.5 – continued from previous page

Cost	Strategy	Significance	Accuracy	Cost	Strategy	Significance	Accuracy
7500	R(x,LR(All))	A	0.793	8000	R(x,LR(All))	A	0.802
	R(x,LR(10))	A	0.790		R(x,LR(9))	A	0.798
	R(x,LR(7))	A	0.787		R(x,LR(8))	A	0.800
	R(x,LR(9))	A	0.786		R(x,LR(10))	A	0.799
	R(x,LR(8))	A	0.785		R(x,LR(7))	A	0.797
	R(x,LR(6))	A B	0.781		R(x,LR(6))	A B	0.792
	R(x,LR(5))	A B	0.778		R(x,LR(5))	A B C	0.788
	R(x,LR(4))	A B C	0.777		R(x,LR(4))	A B C	0.786
	R(x,LR(3))	A B C	0.774		R(x,LR(3))	A B C	0.779
	R(x,LR(2))	B C D	0.764		R(x,LR(2))	B C	0.773
	R(x,LR(1))	D	0.754		R(x,LR(1))	C	0.770
	R(x,LO)	C D	0.753		R(x,LO)	B C	0.767
8500	R(x,LR(All))	A	0.814	9000	R(x,LR(All))	A B	0.812
	R(x,LR(8))	A	0.807		R(x,LR(10))	A	0.810
	R(x,LR(10))	A B	0.804		R(x,LR(9))	A B	0.806
	R(x,LR(9))	A B	0.803		R(x,LR(8))	A B C	0.8052
	R(x,LR(7))	A B	0.803		R(x,LR(7))	A B C	0.804
	R(x,LR(6))	A B C	0.800		R(x,LR(5))	A B C	0.802
	R(x,LR(4))	A B C	0.797		R(x,LR(6))	A B C D	0.801
	R(x,LR(5))	A B C	0.796		R(x,LR(4))	A B C D	0.795
	R(x,LR(3))	A B C	0.788		R(x,LR(3))	B C D	0.791
	R(x,LR(2))	C	0.780		R(x,LR(2))	C D	0.787
	R(x,LR(1))	C	0.780		R(x,LO)	B C D	0.784
	R(x,LO)	B C	0.778		R(x,LR(1))	D	0.783
9500	R(x,LR(8))	A	0.812	10000	R(x,LR(All))	A	0.819
	R(x,LR(All))	A B C	0.811		R(x,LR(10))	A	0.815
	R(x,LR(7))	A	0.811		R(x,LR(7))	A	0.814
	R(x,LR(10))	A B	0.809		R(x,LR(8))	A	0.814
	R(x,LR(9))	A B	0.808		R(x,LR(9))	A B	0.813
	R(x,LR(6))	A B C	0.802		R(x,LR(5))	A B C	0.806
	R(x,LR(5))	A B C D	0.800		R(x,LR(6))	A B C	0.805
	R(x,LR(4))	A B C D	0.799		R(x,LR(4))	A B C	0.804
	R(x,LR(3))	B C D	0.793		R(x,LR(3))	A B C	0.798
	R(x,LR(2))	C D	0.788		R(x,LR(1))	C	0.793
	R(x,LO)	C D	0.785		R(x,LR(2))	C	0.792
	R(x,LR(1))	D	0.784		R(x,LO)	B C	0.792

Table 3.5: Effect of different number of rationales (0 to all) for Random (R) instance selection and $\alpha = 1$, for MR dataset. Results are reported from 500 to 10000 (at every 500) cost milestone. Strategies are listed in the decreasing order of accuracy.

Chapter 4

Annotation Cost Estimation

Annotation Cost in interactive annotation learning is the time it takes the annotator to provide the desired annotations. An annotation task often consists of several subtasks, such as perceiving and understanding the instance, making a decision about the number of annotations to add and their spans, and the user interface actions for adding the annotations to the instance. Annotation cost may vary across instances. Longer documents take longer to read, and hence require more time to annotate. With alternate annotation strategies, such as labeling features (Godbole et al., 2004; Raghavan and Allan, 2007; Druck et al., 2008; Melville and Sindhwani, 2009) or providing rationales (Zaidan et al., 2007), the annotator does more work per instance and the annotation cost would be more. Annotation cost also varies across annotators, and multiple annotators are often employed in annotation tasks. Non-native speakers of English were found to require more time than native speakers to annotate parts of speech (Ringger et al., 2008).

In some domains, the annotation cost is known *a priori* or can be calculated exactly. For example, in biological experiments, the cost might be calculable from the cost of the equipment and the material used (King et al., 2004). In annotation learning, usually the annotation cost is not known *a priori* and often cannot be calculated exactly. It can however be estimated. An estimate for the annotation cost can be used to plan a budget for the annotation tasks. Comparing active selection strategies in terms of the real annotation cost savings may require conducting several user studies (one for each sequence of instances selected by different selection strategies), which is often infeasible. An estimate for the annotation cost can be used instead to compare several selection strategies, before deciding which one to use for a real user experiment. Annotation cost estimate can also be used as a criterion in active selection to directly minimize the annotation cost. In Chapter 3, for cost-sensitive active selection, we used a simple estimate for the annotation cost based on instance length, which has been found to correlate strongly with the annotation time (Ringger et al., 2008; Arora et al., 2009b). We also considered different costs for the two annotation strategies of providing instance's label with or without rationales. However, the annotation cost may also depend on other factors, such as the number of rationales annotated, annotator characteristics in a multi-annotator scenario and other instance characteristics beyond instance length. Other simplifying assumptions that are often made for estimating the annotation cost are the following:

- *annotation cost = length/size of an instance* (Kapoor et al., 2007; Tomanek et al., 2007). It is true that longer sentences take more time to read and understand. However, two sentences of similar length and with similar meaning, may have the grammatical structure of varying complexity that may make one harder to interpret than the other. For example, "Town where I grew up is in the United States" and "I grew up in a town in the United States" are two sentences of the same length and convey similar information with varying amount of linguistic complexity. The first one is more complex, since it contains an independent and a dependent clause. The second one is simpler, since it contains a single independent clause.

- *annotation cost = number of user interface (UI) actions*, such as the number of clicks or the number of brackets added (Kristjansson et al., 2004; Hwa, 2000). This implies that for a document classification task, the annotation cost is the same for all documents. However, some documents may be longer and/or more difficult to categorize, and hence may require more annotation time.

- *annotation cost = confidence of a learned model* (class posterior) (Donmez and Carbonell, 2008b). It is reasonable to assume that instances that are difficult for a model are also hard for the human annotator to annotate. However, some instances, specifically documents, might be easy to classify, but they may be long, and hence may take longer for the annotator to read and annotate.

All of the above factors are important for annotation cost estimation. In addition to the above factors, the annotation cost for a given instance may also vary across annotators. In this chapter, we propose a supervised regression model for estimating the annotation cost with the characteristics of the instance, annotator and annotation strategy as features. We collected the annotated data with annotation time from multiple annotators for a fixed strategy of providing the instance's label with rationales. We train and test the supervised regression model on this data. In the literature, most of the work on annotation cost estimation has focused on training and testing for the same annotator (Settles et al., 2008a). However, this requires that we first collect some annotated data with annotation time from each new annotator, in order to build an annotation cost estimator for him/her. Other work in the literature has focused on training and testing with annotation time averaged over multiple annotators (Vijayanarasimhan and Grauman, 2011), or have assumed a similar mix of annotators in train and test set, and do not include annotator characteristics in annotation cost estimation (Ringger et al., 2008). However, annotation times may vary significantly across annotators, and average annotation time may not be a good estimate. In this work, we train and test an annotation cost estimator on different annotators, and use the annotator characteristics as features in our model to represent an annotator. This is important, since we may not always know all our annotators before building the model, and training an estimator for each new annotator can be costly. For a sentiment classification task, we show that an annotation cost estimate from the proposed model based on the characteristics of an instance, strategy and annotator outperforms simpler estimates based on any one of these characteristics.

For the rest of the chapter, we first present our supervised regression model for annotation cost estimation. We then present the experiments and results, followed by the related work, conclusions and suggestions for the future work. The work presented here is an exposition of the work presented in (Arora et al., 2009b).

4.1 Proposed Approach for Annotation Cost Estimation

In this work, we estimate the annotation cost for a movie review classification task (Zaidan et al., 2007), in a multi-annotator environment. Given a movie review, the annotator votes positive if the review expresses a positive opinion about the movie, and suggests watching the movie. The annotator votes negative if the review expresses a negative opinion about the movie, and suggests not watching the movie. In addition to the vote, the annotator provides rationales (Zaidan et al., 2007), spans of text in support of his/her vote. The rationales provide indirect feedback on features. Prior work in the literature has demonstrated that with rationales same performance can be achieved with less annotated data (Arora and Nyberg, 2009; Zaidan et al., 2007; Zaidan and Eisner, 2008; Abedin et al., 2011). In Chapter 3, we showed that rationales (as votes on sub-instances in an instance) provide a significant improvement in performance for a given annotation cost, when the cost for annotating them is small. In this work, we follow the same annotation guidelines as Zaidan et al. (2007) for collecting the annotated data. That is, rationales are annotated as sub-strings in text[1].

We formally describe the annotation cost estimation problem as estimating the cost $C(x_i, s_j, a_k)$ for annotator a_k annotating instance x_i with strategy s_j. We use a supervised regression model for annotation cost estimation and investigate the following three broad categories of features for this model:

- *Instance Characteristics:* Characteristics of the instance to be annotated, such as the character length, average sentence length, readability, instance's difficulty (measured in terms of the learned model's uncertainty), etc., are important for estimating the annotation time, as also suggested in the literature (Kapoor et al., 2007; Tomanek et al., 2007; Donmez and Carbonell, 2008b; Ringger et al., 2008; Settles et al., 2008a). Table 4.1 describes the instance characteristics we use in this work, and the intuition behind them.

Feature	**Definition**	**Intuition**
Character Length (CL)	Length of a review in terms of the number of characters	Longer documents take longer to read and annotate
Polar word Count (PC)	Number of words that are polar (strong subjective words from the lexicon (Wilson et al., 2005))	High subjectivity in text suggests that the annotator may need more time to determine the sentiment of a review
Stop word Percent (SP)	Percentage of words that are stop words	A high percentage of stop words suggests that the text may not be very complex and hence easier to read
Average Sentence Length (SL)	Average of the character length of sentences in the review	Long sentences in a review may make it harder to read

Table 4.1: Instance characteristics used as features in the supervised model for annotation cost estimation.

- *Annotator Characteristics:* Annotator characteristics, such as their fluency with the language, familiarity with the annotation task, annotation experience, familiarity with reading online text, etc., should effect the annotation time. In this work, we use the annotator *nativeness* as a feature, which has been found to have a significant effect on annotation time (Ringger et al., 2008). The reviews to be annotated in our annotation task are in English. A *nativeness* score is assigned to each annotator as follows.

[1]This work was done prior to the work in Chapter 3, where we consider rationales as votes on sub-instances in an instance.

  - A score of 3 is assigned to an annotator whose first language is English.
  - A score of 2 is assigned to an annotator whose first language is not English, but who has done most of his/her education in English, or who has been in the United States or an English speaking country for several years.
  - A score of 1 is assigned to all other annotators.

- *Annotation Task/Strategy Characteristics:* These characteristics may be fixed beforehand or are captured during or after the annotation task is done. For example, number of rationales (NR), number of named entity annotations, etc. Such characteristics are not always known *a priori*. When calculating the annotation cost savings from active learning, they are known *a priori*, as the annotations exist. For selective sampling or annotation budget planning, they may be unknown. For certain annotation tasks, the number of annotations per instance is fixed, for example in a document classification task, there is only one annotation per instance. The rationale annotation strategy may also fix the number of rationales the annotator provides. In such cases, we know the number of annotations *a priori*. However, in certain annotation tasks, such as named entity annotation, the number of entities per sentence or document may vary. The rationale annotation strategy may also leave it to the annotator to decide how many rationales should be annotated (Zaidan et al., 2007). However, we could estimate the number of rationales that an annotator may annotate for a given instance by building a rationale model for each annotator from the instances labeled so far, like the one used in Chapter 3 for estimating the risk. Similarly, we could use the model trained on data annotated so far to estimate the expected number of named entities in an unlabeled instance. In this work, we use the number of rationales as an annotation task characteristic. If the number of rationales is not fixed and unknown at the time of annotation cost estimation, we can use the expected number of rationales.

We use the above characteristics as features in the supervised model. Our hypothesis is that all three categories of features are important for estimating the annotation cost accurately. We show that an annotation cost estimate from a model based on features from all three categories outperforms a simpler estimate based on any one of them.

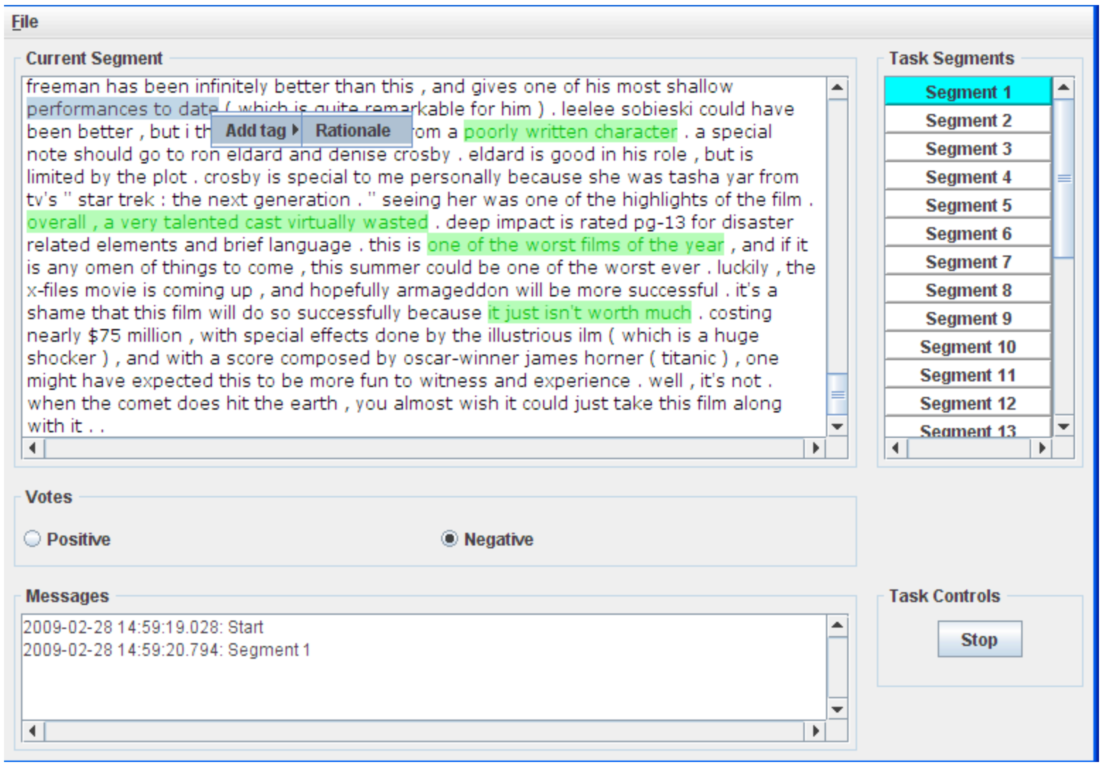
4.2 Experiments and Results

In this section, we first describe our data collection process, and an analysis of the data we collected. Following this, we present our experiments and results for annotation cost estimation on this data, as presented in (Arora et al., 2009b).

4.2.1 Data Collection and Analysis

The data we use was collected as part of a graduate course. Twenty annotators (students and instructors) were grouped into five groups of four each. The groups were created such that each group had a similar variance in annotator characteristics, such as the department, educational experience, programming experience, etc. We used the first 200 movie reviews (100 positive and 100 negative) from the dataset shared by Zaidan et al. (2007). Each group annotated 25 movie reviews, randomly selected from the 200 reviews. All annotators in each group annotated all 25 reviews. The data has been made available for research purposes[2]. Figure 4.1 shows a screenshot of the GUI used. We performed an analysis similar to Settles et al. (2008a) on the data collected, where we answer the following questions:

[2]`www.cs.cmu.edu/~shilpaa/datasets/ial/ial-uee-mr-v0.1.zip`



freeman has been infinitely better than this , and gives one of his most shallow performances to date (which is quite remarkable for him) . leelee sobieski could have been better , but i th om a poorly written character . a special note should go to ron eldard and denise crosby . eldard is good in his role , but is limited by the plot . crosby is special to me personally because she was tasha yar from tv's " star trek : the next generation . " seeing her was one of the highlights of the film . overall , a very talented cast virtually wasted . deep impact is rated pg-13 for disaster related elements and brief language . this is one of the worst films of the year , and if it is any omen of things to come , this summer could be one of the worst ever . luckily , the x-files movie is coming up , and hopefully armageddon will be more successful . it's a shame that this film will do so successfully because it just isn't worth much . costing nearly $75 million , with special effects done by the illustrious ilm (which is a huge shocker) , and with a score composed by oscar-winner james horner (titanic) , one might have expected this to be more fun to witness and experience . well , it's not . when the comet does hit the earth , you almost wish it could just take this film along with it . .

Positive [✓ Negative]

```
2009-02-28 14:59:19.028: Start
2009-02-28 14:59:20.794: Segment 1
```

Figure 4.1: The GUI used for the annotation task. The annotator selects the review (segment) to annotate from the list in the right panel. The review text is displayed in the left panel. The annotator votes positive or negative using the radio buttons. Rationales are added by selecting a span of text and right clicking to select the rationale tag. The *start/stop* button can be used to pause the current task.

- *How do the annotation times vary across instances?* If instances take similar time to annotate, then the number of instances can be used as an approximation for the annotation cost. Figure 4.2 shows the histogram for average annotation time per instance (averaged over 4 annotators in a group). As can be seen from the mean ($\mu = 165$ sec) and the standard deviation ($\sigma = 68.85$ sec), there is a meaningful variance in the annotation times across instances (standard deviation is close to half of the mean).

- *How do the annotation times vary across annotators?* The box plot in Figure 4.3 shows the distribution of annotation times across annotators. As can be seen, some annotators take in general much longer than others (comparing the annotation time of an annotator to the average annotation time), and the distribution of times is very different across annotators. For some annotators, the annotation time varies a lot across instances (large box and long whiskers), but not so much for the others.

4.2.2 Experimental Setup

We learn an annotation cost estimator using the Linear Regression and SMO Regression (Smola and Scholkopf, 1998) learners from the Weka machine learning toolkit (Witten and Frank, 2005).

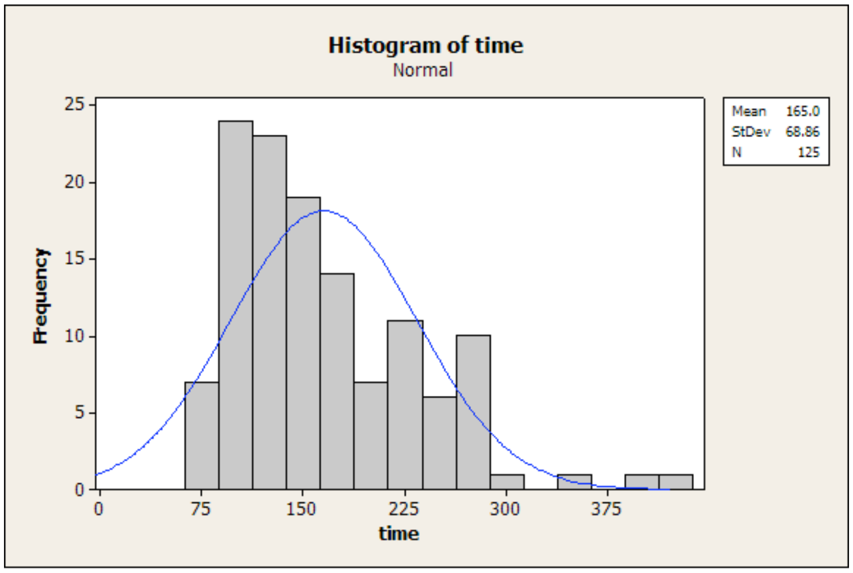


Figure 4.2: Distribution of average annotation time across instances (averaged over 4 annotators).

As mentioned earlier, we have 5 sets of 25 documents each, and each set was annotated by four annotators. The results reported are averaged over the five folds (where each fold is one of the five sets), and the two algorithms (Linear Regression and SMO Regression). Varying the algorithm helps us find the most predictive feature combinations across different algorithms. Each fold/set was annotated by a different group of annotators. That is, we train and test on the data from different annotators. We use JMP[3] and Minitab[4] statistical tools for our analysis. We use an ANOVA model with Standard Least Squares fitting to compare the different experimental conditions. For significance results reported, we used a two-tailed paired T-test, considering ($p < 0.05$) as significant. Next, we suggest two types of metrics for evaluating an annotation cost estimate.

4.2.3 Evaluation Metrics

We suggest the following two types of metrics for evaluating an annotation cost estimate.

- Type A: This type of metric measures the amount by which the estimate differs from the true value. For example, Root Mean Square error (RMS), Mean Absolute Error (MAE), etc. The lower the value of this score, the better the estimate.
- Type B: This type of metric measures the strength and direction of the relationship between the estimate and the true value. For example, Pearson's Correlation Coefficient (CRCoef). The higher the value of this score, the better the estimate.

Depending on the objective for estimating the annotation cost, we may prefer one evaluation metric over the other. For example, when estimating the annotation cost for budget planning, Type A metric is more suitable as it measures the absolute difference between the estimate and the true value, which can be directly interpreted in terms of dollars. On the other hand, if the

[3]`http://www.jmp.com/software/`
[4]`http://www.minitab.com/`

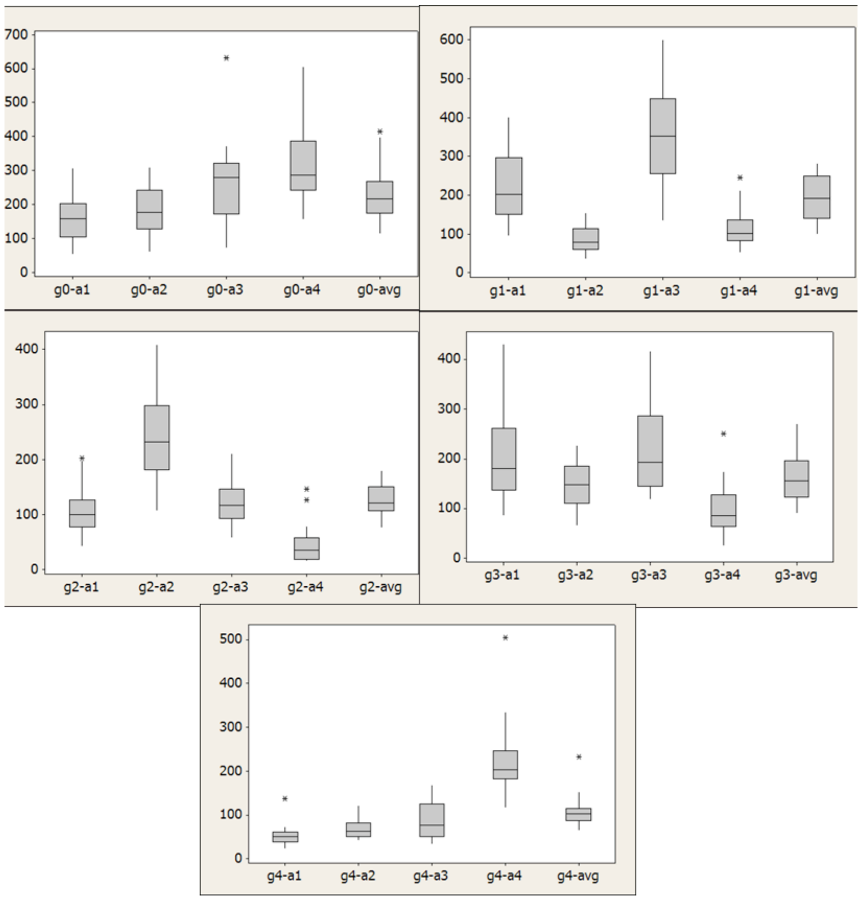


Figure 4.3: Box plot shows the annotation time (in sec) distribution (y-axis) for an annotator (x-axis) for a set of 25 documents. *g0-a1* represents annotator 1 of group 0 and *g0-avg* represents the average annotation time. A box represents the middle 50% of annotation times, with the line representing the median. Whiskers on either side span the 1st and 4th quartiles and asterisks indicate the outliers.

estimated annotation cost is to be used for ranking and selecting among candidates for annotation, it is important that the estimate correlates well with the true annotation cost, and hence Type B metric is more suitable for evaluation. In our experiments, we use RMS error (Type A) and CRCoef (Type B) metrics for evaluation.

4.2.4 Results and Discussion

We first evaluate the instance characteristics individually. Table 4.2 presents the results. As can be seen, among the four instance characteristics in Table 4.1, Character Length (CL) performs the best. This result is consistent with the earlier results in the literature (Tomanek et al., 2007; Haertel et al., 2008). However, CL is not significantly better than Polar Word Count (PC). We found that a combination of CL and PC is not significantly different from CL. However, this combination is significantly better than polar word count alone in CRCoef, although the two are not significantly different in RMS. This suggests that the instance length is a strong instance characteristic for annotation cost estimation, and polar word count is a potential feature to consider for annotation cost estimation in a sentiment classification task.

Feature	**CR-Coef**	**RMS**
CL	**0.358**	**104.51**
PC	0.337	105.92
SP	-0.041*	114.34*
SL	0.042*	114.50*

Table 4.2: Results for Character Length (CL), Polar word Count (PC), Stop word Percent (SP) and average Sentence Length (SL) as features in a supervised regression model for annotation cost estimation. The best performance is highlighted in bold. * marks the results significantly worse than the best. CR-Coef: Correlation Coefficient and RMS: Root Mean Squared error.

We now analyze the performance of combinations of features from the three categories, choosing the best feature from each category. As can be seen from the results in Table 4.3, using features from all three categories performs better than using any subset of them. This result is promising, since character length is often used as an approximation for annotation cost, and we show that a combination of character length with annotator and annotation task characteristics, significantly outperforms character length alone. Even the combination of number of rationales and annotator nativeness, without character length, significantly outperforms character length alone in CRCoef. This suggests that the number of rationales we expect or require in a review and the annotator characteristics are important factors to consider in annotation cost estimation.

The effect of annotator nativeness feature is somewhat mixed. As can be seen from Table 4.3, it always improves the correlation coefficient when added to a feature combination. However, it only improves RMS when added to the feature combination of (CL+NR), although this difference is not significant. To investigate this further, we evaluate our assumption that the native speakers take less time to annotate. We found that the annotators with nativeness value of 3 (i.e. annotators with first language as English) took on average less time than those with nativeness value of 2 or 1. Between annotators with nativeness value of 2 and 1, sometimes annotators with value 1 took less time than annotators with value 2. An alternative is to consider two levels of significance instead of three. We may also consider other ways to measure the nativeness of an annotator, such as their GRE or TOEFL scores. We may also want to consider other annotator characteristics beyond nativeness, for example annotation experience, etc. In Table 4.3, we also present the weights learnt for these features using a linear regression model trained on all the data (not just the training set). We publish these to facilitate others to use our model in a similar setting.

We also analyze how the characteristics correlate with each other and with the annotation time. As can be seen from Table 4.4, all features have a significant correlation with the annotation

CL	**NR**	**AN**	**Const.**	**CR-Coef**	**RMS**
		-29.33	220.77	0.135^*	123.93^*
	17.59		82.81	0.486	95.29
0.027			61.53	0.357^*	104.51^*
	19.11	-40.78	153.21	0.550^+	96.04
0.028		-32.79	120.18	0.397^*	109.85^*
0.02	15.15		17.57	0.553^+	90.27^+
0.021	16.64	-41.84	88.09	$\mathbf{0.626}^+$	$\mathbf{88.44}^+$

Table 4.3: Results for seven feature combinations of Character Length (CL), Number of Rationales (NR) and Annotator Nativeness (AN). The values in feature and 'Const.' columns are weights and constant for the linear regression model trained on all the data. A missing weight for a feature indicates that it wasnt used in that feature combination. The numbers in bold are the results for the best feature combination. * marks the results significantly worse than the best. $^+$ marks the results significantly better than CL. CR-Coef: Correlation Coefficient and RMS: Root Mean Squared error.

time, except Stop-word Percent (SP) and average Sentence Length (SL). Number of rationales (NR) has a higher correlation with annotation time ($R = 0.529$) than Character Length (CL) ($R = 0.417$), which suggests that the number of rationales has more influence on annotation time than character length. A low correlation between Number of Rationales (NR) and Character Length (CL) ($R = 0.238$) implies that the longer documents do not necessarily contain more rationales. Polar word Count (PC) has similar correlation with Annotation Time (AT) as Character Length (CL). Strong correlation between CL and PC ($R = 0.89$) is expected, since reviews are essentially people's opinions, longer documents are expected to have more polar words. Due to a strong correlation between character length and polar word count, we only used one of them in our combined model with annotator and annotation task characteristics. Since character length gave us better performance than polar word count, as shown in Table 4.2, we used the character length as the instance characteristic in the combined model. Nonetheless, we evaluated adding polar word count feature to the best feature combination in Table 4.3. However, it did not make a significant difference in performance.

In Section 4.2.3, we proposed two types of metrics for evaluating annotation cost estimates. As can be seen in Table 4.5, the ranking of feature combinations based on performance is different for the two metrics. As discussed in Section 4.2.3, the two metrics evaluate an estimate from different perspectives. Hence, the best estimator should be chosen based on the evaluation metric suitable for the given objective.

To the best of our knowledge, there is no previous work in the literature on estimating the annotation cost for the movie review dataset we use in this work. The most similar to our annotation task is the task of classifying text as speculative or definite studied in (Settles et al., 2008a), since both tasks involve subjective language and require classifying the text into one of the two classes. However, our task also involves annotating rationales, and we consider multiple annotators. Although the results are not directly comparable, our model accounts for 39% of the variance in the model ($R^2 = 0.39$, where R is the correlation (0.626))[5], compared to 34% ($R^2 = 0.34$, $R = 0.587$)

[5]The percentage of variance accounted for by a model is obtained by squaring the correlation coefficient to get what is called the *Coefficient of Determination*(`http://en.wikipedia.org/wiki/Coefficient_of_determination`)

	AT	**CL**	**NR**	**AN**	**PC**	**SP**	**SL**
AT	1						
CL	**0.42**	1					
NR	**0.53**	**0.24**	1				
AN	**-0.22**	0.06	**0.15**	1			
PC	**0.4**	**0.89**	**0.28**	**0.11**	1		
SP	0.03	0.06	**0.14**	0.03	0.04	1	
SL	0.08	**0.15**	0.01	-0.01	**0.14**	**-0.13**	1

Table 4.4: Correlation coefficient between Character Length (CL), Number of Rationales (NR), Annotator Nativeness (AN), Polar word Count (PC), Stop word Percent (SP), average Sentence Length (SL) and Annotation Time (AT), calculated over all documents (125) and all annotators (20). Significant correlations are highlighted in bold.

Rank	**CR-Coef**	**RMS**
1	(CL+NR+AN)	(CL+NR+AN)
2	(CL+NR)	(CL+NR)
3	(NR+AN)	(NR)
4	(NR)	(NR+AN)
5	(CL+AN)	(CL)
6	(CL)	(CL+AN)
7	(AN)	(AN)

Table 4.5: Ranking of feature combinations based on two metrics: Correlation Coefficient (CRCoef) and Root Mean Squared (RMS) error.

in (Settles et al., 2008a) for speculative text classification task.

4.3 Related Work

There has been some recent research effort in using supervised learning for annotation cost estimation (Settles et al., 2008a; Ringger et al., 2008; Wallace et al., 2010; Vijayanarasimhan and Grauman, 2011). Settles et al. (2008a) present a detailed analysis of the annotation cost for four learning tasks: named entity recognition, image retrieval, speculative vs. definite distinction, and information extraction. They study the effect of domain, annotator, etc., on the annotation cost. Similar to our work, they found a significant variance in annotation time across instances and annotators for a given task. For supervised annotation cost estimation, Settles et al. (2008a) use the SMO algorithm (Smola and Schölkopf, 2004) for Support Vector Regression. They used only instance level characteristics, such as instance length, bag-of-words, number of ASCII characters, number of entities, model's confidence about the instance's label, etc., as features for annotation cost estimation. For three of their tasks, the correlation between the estimated and actual annotation times was in the range (R = 0.587 to 0.852). Settles et al. (2008a) train and test their estimator on data from the same annotator. Thus, in order to use their model for a new annotator, we would need to first collect the data with annotation times for that annotator, and train a model. In our work, we estimate the annotation cost for an unseen annotator. The model is trained on data from several annotators and we represent an annotator by his/her characteristics.

Ringger et al. (2008) use a linear regression model for estimating the annotation cost for tagging parts of speech. In their annotation task, the annotator is provided with the model's predictions and he/she corrects the mistakes the model makes, and also provides any missing annotations. About 30 annotators annotated 36 different instances each. The authors present about 13 characteristics of an instance, annotator and annotation task. However, they only use the number of tokens in a sentence and the number of corrections needed as features in their model. They report that the other variables didn't have a significant effect when evaluated using a Bayesian Information Criterion (from the R package[6]). Ringger et al. (2008) noticed that nativeness of the annotator effects the annotation time, but they chose not to include that feature in their model, as they expected to have a similar mix of skills and background for annotators in their test set, as in their training data. A low adjusted correlation value for their model ($R^2 = 0.181$) indicates that there is only a weak correlation between annotation time and a linear combination of the length of an instance and the number of corrections.

Annotation cost may also vary with time. It should be more in the beginning and gradually decrease as the annotator becomes familiar with the task. Settles et al. (2008a) and Wallace et al. (2010) in their user experiments observed that the annotation time per instance decreases as the annotator labels more data and becomes familiar with the task. To model this observation, Wallace et al. (2010) use a linear spline regression model with the length of an instance and number of instances labeled so far as predictive variables. The spline function with a knot captures the behavior that the learning rate increases rapidly in the beginning and slowly thereafter. The parameters for this model are estimated online as more data is collected in active learning. They show that using the predicted annotation time in active selection criterion provides better performance for the same annotation cost.

In a more recent work, Vijayanarasimhan and Grauman (2011) estimate the cost for annotating an image with the objects it contains. They use features based on image characteristics, such as a histogram of oriented gradients, a gray-scale histogram, features based on the edge density and color uniformity, etc., to train a Support Vector Regression model. They show that using an estimate of the annotation cost in the active learning criterion can help to reduce the total annotation cost for the desired performance, in comparison to using a fixed annotation cost for all instances and annotations. They collect the data from multiple annotators and use the average annotation time for an image and its annotation. They observed a strong linear correlation between the predicted and actual annotation times, and a root mean square error of 11.1 seconds (Vijayanarasimhan and Grauman, 2011).

In our work, we use a supervised model for estimating the annotation cost for a given instance, annotator, and annotation strategy. As described above, in the literature, it has been shown that annotation times for instances can be used to train a model, and predict annotation cost for an unseen instance. However, the annotation times for the annotators in the training set, has not been used to predict the annotation time for a new annotator. This is one of the novel contributions of our work. Additionally, we consider a new annotation scenario of providing rationales in addition to the instance label, when estimating the annotation cost.

4.4 Summary and Future Work

In this section, we presented our work on annotation cost estimation using a supervised regression model. An annotation cost estimate is useful for three purposes: 1) evaluating active selection

[6]`http://www.r-project.org/`

strategies, 2) selecting among candidates for annotation, to directly minimize the total annotation cost for the desired performance, and 3) planning a budget for an annotation task.

We presented a detailed investigation of annotation cost estimation for active learning in a multi-annotator environment with indirect feature feedback through rationales. We showed that a combination of characteristics from the three categories of instance, annotation task and annotator characteristics, performs better than any subset of them. In this work, we evaluated a few characteristics from each category. There are many more characteristics to explore, we suggest a few below:

- *Instance Characteristics:* Readability scores such as Flesch score (Flesch, 1948), bag-of-words and other linguistic features, model's confidence on instance's label, etc.
- *Annotator Characteristics:* Other methods for assessing nativeness of an annotator, such as GRE/TOEFL score, etc. Other characteristics such as annotation experience, experience reading online text, etc. We may also use a questionnaire to assess the annotators and assign them a score.
- *Annotation Task Characteristics:* Features such as length of rationales, distance between rationales, vote, etc.

Some of the instance based features we used for annotation cost estimation in this work are domain independent, e.g. character length, average sentence length, and percentage of stop words. In a heterogenous corpus, instances from two different genres may have same length and other domain independent characteristics, however, they may require different annotation time. Some of the instance based features suggested above, such as readability score, bag-of-words, etc., may help to distinguish among instances from different genres. Some of the other instance features we used for annotation cost estimation, such as polar word count, are domain specific. We found that the polar word count is predictive of the annotation cost for a sentiment classification task. However, it may not be relevant for a topic classification task. We may have similar domain specific features for each task. The features to include in a linear annotation cost estimation model can be determined by evaluating their correlation with the annotation time on a small sample of data collected with annotation time. This would allow us to make a well-reasoned decision about whether or not to include domain-specific features in a model for annotation cost estimation .

In our experiments, we found that annotation time varies across annotators, and annotator nativeness is an important factor to consider in annotation cost estimation. In the work on annotation cost estimation in this chapter, rationales were annotated as sub-strings of text. As mentioned in Chapter 3, for annotating rationales as sub-strings, the annotator needs to perform the cognitive task of identifying the appropriate boundaries for a rationale. If an annotator tends to highlight longer rationales, we would expect annotation time to be more for him/her, since it would require dragging the mouse for a longer time. The time to determine the appropriate boundaries for the rationales may also vary across annotators. If rationales are instead annotated as votes on sentences in a document (as in Chapter 3), we may find that the annotation time does not vary as much across annotators, since they don't need to determine the appropriate spans for the rationales. A future user experiment should analyze the variation in time across annotators when rationales are annotated as votes on sentences in a document, instead of sub-strings in a document.

Annotation cost may also vary over time. As the annotators become familiar with the task, their annotation speed may increase. On the other hand, annotation speed may decrease after labeling several instances, due to fatigue. Similar to Wallace et al. (2010), we may capture this intuition by adding a feature to the model for the number of instances labeled so far. We may also use a spline regression model like in their work (also described in Section 4.3), with two knots instead of one, to capture the increase in annotation speed with familiarity and decrease in annotation speed due to fatigue.

For evaluating the performance of an annotation cost estimator, we used two evaluation metrics: Root Mean Squared (RMS) error and Correlation Coefficient (CRCoef). Depending on the goal for estimating the annotation cost, appropriate evaluation measure should be used. When estimating the annotation cost for budget planning, RMS is more suitable. When the estimated annotation is used in active selection, CRCoef is more suitable.

As mentioned before, an annotation task consists of the following sub-tasks: 1) reading the instance, 2) making a decision about the annotations, and 3) adding the annotations to the instance using the interface. Different characteristics may be useful for estimating the annotation time for these sub-tasks. A possible direction for future work is to model each of these sub-tasks separately in annotation cost estimation.

Chapter 5

Assessment of Benefit from Feature Feedback

Direct and indirect feature feedback have been shown to reduce the total number of instances required to achieve the desired performance (Raghavan et al., 2006; Druck et al., 2008; Druck et al., 2009; Zaidan et al., 2007; Abedin et al., 2011; Donahue and Grauman, 2011). However, feature feedback comes with an additional cost. In Chapter 3, we showed that indirect feature feedback through rationales, in addition to the instance's label, provides better performance for a given annotation cost, when the additional cost for annotating them is small. The cost for annotating rationales may vary across instances, annotators, annotation tasks, user interface design, etc. For different additional costs for annotating rationales, we showed that selectively asking for rationales for a few instances performs as well as or better than the best fixed strategy of annotating instance's label only or instance's label with rationales.

Learning problems may benefit differently from feature feedback (direct or indirect). In Chapter 3, we evaluated the benefit from indirect feature feedback through rationales for two datasets and four classification tasks. We observed some differences in the benefit from rationales across different tasks. In this chapter, we formally analyze how benefit from feature feedback varies across learning problems. It is intuitive that after the model has been trained on several labeled instances, feature feedback may not provide much benefit. If the feature space is large, we may need several labeled instances to identify the relevant features, while relatively fewer labeled features may help us quickly find these relevant features. When hand crafted features from a domain expert are used (Pradhan et al., 2004), we expect to gain less from feature feedback, as most of the features will be relevant. On the other hand, when features are extracted automatically as patterns in the annotation graphs, like the approach discussed in Chapter 2, feature feedback can help to identify a few relevant features from the large feature space. In active learning, instances to be labeled are selectively sampled in each iteration. The benefit from feature feedback will depend on the instances that are used to train the model in each iteration. In the case of indirect feature feedback through rationales or direct feature feedback in context, instances selected will also determine what features receive feedback. Hence, *instance selection strategy* should affect the benefit from feature feedback. In text classification, an instance, such as a document, may contain a large amount of text, and even a simple unigram representation will generate a lot of features. Often only a part of the text is relevant for the classification task. For example, in movie reviews, often the reviewers talk about the plot and characters, in addition to providing their opinion about the movie. Often this extra information is not relevant to the classification task, and bloats the feature space without adding many useful features. With feature feedback, we hope to filter out some of this noise and improve the model. Thus, the *amount of irrelevant information* in an instance should play an important role in determining the benefit from feature feedback. We expect to see less of such noise when the text instance is more concise. For example, a movie review snippet (about a sentence length) tends to have less irrelevant text than a full movie review (several sentences). In addition to analyzing document instances with varying amount of noise, we also compare the benefit from feature feedback for problems at different instance *granularity.* Instance granularity for a learning problem can be defined in terms of the average amount of text in its instances.

Formally, we define a *learning problem* ($P = \{D, G, F, L, I, S\}$) as a tuple of the domain (D), instance granularity (G), feature representation (F), amount of labeled data (L), amount of irrelevant text (I) and instance selection strategy (S); and analyze how benefit from feature feedback varies with these characteristics. The benefit from feature feedback will also depend on how the feedback is solicited from the user, and how it is incorporated back into the model. Independent of these factors, we estimate the maximum benefit from feature feedback, and analyze how it varies across problems. Thus, our analysis is not specific to any particular approach for seeking and incorporating feature feedback. We also study measures proposed in the literature and propose some new ones for categorizing learning problems, and evaluate their correlation with the maximum benefit from feature feedback. We present this analysis for two sentiment classification tasks with different instance granularity, and a webpage category classification task.

For the remaining chapter, we first discuss measures for categorizing learning problems and how we estimate the maximum benefit from feature feedback. We then discuss our experimental setup and analysis, as presented in (Arora and Nyberg, 2011), followed by the related work, conclusions and suggestions for the future work.

5.1 Measures for Categorizing Learning Problems

There has been little work in the literature on categorizing learning problems and analyzing how the benefit from feature feedback varies with them. To the best of our knowledge there is only one work in this area by Raghavan et al. (2007). They categorize problems in terms of their *feature complexity.* Feature complexity is defined in terms of the minimum number of features required to learn a good classifier (close to the maximum performance). If the concept can be described by a weighted combination of a few well-selected features, it is considered to be of low complexity.

In this estimate of complexity, an assumption is made that the best performance is achieved when the learner has access to all available features, and not for any subset of the features. This is a reasonable assumption for text classification problems with robust learners like SVMs, together with appropriate regularization and/or sufficient training data. It is also assumed that there is ample amount of training data available to achieve an acceptable level of performance (of above 75% Maximum F1) using a linear SVM.

Evaluating all possible combinations of features to determine the minimum number of features required to achieve close to the best performance can be computationally very expensive. Instead, the feature complexity is estimated using an intelligent ranking of the features. This ranking is based on the discriminative ability of the features, determined using a large amount of labeled data (referred to as *oracle*), and a feature selection criterion such as Information Gain (Rijsbergen, 1979).

It is intuitive that the rate of learning, i.e. the rate at which the performance improves as we add more features to the model is also associated with the problem complexity. Raghavan et al. (2007) define another feature complexity measure called the *feature learning convergence profile*

(p_{fl}) as the normalized area under the feature learning curve (performance vs. log of the number of features (t)), given by:

$$p_{fl} = \frac{\sum_{t=1}^{log_2 N} F1(M, 2^t)}{log_2 N \times F1(M, N)} \tag{5.1}$$

where $F1(M, 2^t)$ is the F1 score on the test data, when using all M instances for training, with top ranked 2^t features. $F1(M, N)$ is the F1 score when using all N features. The feature learning curve is plotted using the performance of the classifier with k top-ranked features (based on information gain score from the oracle), at exponentially increasing intervals of k to emphasize the relative increase in the feature space size[1]. The three feature complexity measures proposed by Raghavan et al. (2007) are the following: 1) *Feature size complexity (N_f)*: Logarithm (base 2) of the number of features needed to achieve 95% of the best performance (when all instances are available), 2) *Feature profile complexity (F_{pc})*, given by ($F_{pc} = 1 - p_{fl}$), where p_{fl} is the feature learning convergence profile (Equation 5.1), and 3) *Combined feature complexity (C_f)* , $C_f = F_{pc} * n_f$, which combines both the learning profile and the number of features required.

The feature complexity measures discussed above assume that we have a large amount of labeled data already available. Hence, these measures cannot be applied to new annotation tasks where labeled data does not exist. Thus, these measures are good for understanding when feature feedback helps, but not for predicting if feature feedback will be helpful for a new annotation task for which we are actively collecting annotations. We suggest some measures that do not assume a large amount of labeled data. The benefit from feature feedback at any given point in learning will depend on the uncertainty of the current model on its predictions, since it suggests uncertainty on the features, and hence the scope for benefit from feature feedback. We use the probability of the predicted label from the model as an estimate of the model's uncertainty. We evaluate how the benefit from feature feedback varies with summary statistics such as mean, median and maximum probability from the model on labels for instances in a held out dataset.

5.2 Estimating the Maximum Benefit

As mentioned before, the benefit from feature feedback will depend on how the feedback is solicited and how it is incorporated back into the model. In this work we analyze the expected maximum benefit to a learning problem from feature feedback, independent of the feedback solicitation and incorporation approach. Annotating instances with or without feature feedback may incur different annotation time/cost. It is only fair to compare different annotation strategies at the same annotation cost. Raghavan et al. (2006) found that on average labeling an instance takes the same amount of time as direct feedback on 5 features. Zaidan et al. (2007) found that on average it takes twice as much time to annotate an instance with rationales than to annotate one without rationales. In our analysis, we focus on feedback on features in context of the instance, i.e., indirect feature feedback through rationales or direct feedback on features that occur in the instance being labeled. Thus, based on the findings in Zaidan et al. (2007), we assume that on average annotating an instance with feature feedback takes twice as much time as annotating an instance without feature feedback. We define a currency for annotation cost as *Annotation cost Units (AUs)*. For an annotation budget of a AUs, we compare the two annotation strategies of annotating a instances without feature feedback or $\frac{a}{2}$ instances with feature feedback.

[1]Adding 50 features to a total 50 features, will give a significantly higher boost in performance, than adding 50 features to a total of 1000 features.

In this work, we only focus on using feature feedback as an alternative to the labeled data, i.e., to provide evidence about features in terms of their relevance and class association. Thus, the best feature feedback can do is provide as much evidence about features as evidence from a large amount of labeled data (oracle). Let $F1(k, N_m)$ be the F1 score of a model trained with features that occur in m training instances (N_m) and evidence for these features from k instances ($k \geq m$). For an annotation budget of a AUs, we define the maximum $\underline{i}$mprovement in $\underline{p}$erformance with feature feedback (IP_a) as the difference in performance with feature feedback from oracle on $\frac{a}{2}$ training instances and performance with a training instances without feature feedback.

$$IP_a = F1(o, N_{\frac{a}{2}}) - F1(a, N_a) \tag{5.2}$$

where o is the number of instances in the oracle dataset ($o >> a$). We also compare the two annotation strategies of instance label with or without feature feedback in terms of the maximum improvement in the learning rate. For an annotation budget of a AUs, we define the maximum $\underline{i}$mprovement in $\underline{l}$earning $\underline{r}$ate from 0 to a AUs (ILR_{0-a}) as follows.

$$ILR_{0-a} = {p_{cp}}^{wFF} - {p_{cp}}^{woFF} \tag{5.3}$$

where ${p_{cp}}^{wFF}$ and ${p_{cp}}^{woFF}$ are the convergence profiles with and without feature feedback at given annotation cost. The convergence profile, defined similar to the feature convergence profile in Equation 5.1, is calculated as follows[2].

$${p_{cp}}^{wFF} = \frac{\sum_{t=1}^{log_2 \frac{a}{2}} F1(o, N_{2^t})}{log_2 \frac{a}{2} \times F1(o, N_{\frac{a}{2}})} \tag{5.4}$$

$${p_{cp}}^{woFF} = \frac{\sum_{t=2}^{log_2 a} F1(2^t, N_{2^t})}{(log_2 a - 1) \times F1(a, N_a)} \tag{5.5}$$

where 2^t denotes the training data size in iteration t. We increase the training data size exponentially to emphasize the relative increase, since adding a few labeled instances earlier in learning would give a significantly more improvement in performance than adding the same number of instances later on. For example, as illustrated in (Raghavan et al., 2007), adding 50 instances to a total of 50 training instances, will give a significantly higher boost in performance, than adding 50 instances to a total of 1000 training instances.

In the literature (Raghavan et al., 2007), the benefit from feature feedback is evaluated only in terms of the gain in learning speed. However, the learning rate does not tell us how much improvement we get in performance at a given stage in learning. In fact, even if at every point in the learning curve, the performance with feature feedback is lower than the performance without feature feedback, the rate of convergence to the corresponding maximum performance may still be higher, when using feature feedback. Thus, in this work, in addition to evaluating the improvement in the learning speed, we also evaluate the improvement in the absolute performance, at any given point in learning.

5.3 Experiments and Results

In this section, we describe our experimental setup, followed by the results.

[2]$(log_2 a - 1)$ in Eq. 5.5 is because for the same annotation cost, we start with two instances when not seeking rationales ($log_2 2 = 1$), and one instance when seeking rationales ($log_2 1 = 0$).

5.3.1 Data and Experimental Setup

In this set of experiments, we analyze three datasets: 1) Movie reviews with rationale annotations by Zaidan et al. (2007), where the task is to classify the sentiment (positive/negative) of a review, 2) Movie review snippets from Rotten Tomatoes (Pang and Lee, 2005), and 3) WebKB dataset with the task of classifying whether or not a webpage is a faculty member's homepage.

As mentioned before, we formally define a learning problem in terms of its domain, instance granularity, feature representation, amount of labeled data, amount of irrelevant text, and instance selection strategy. Table 5.1 presents these variables and their values in our experiments. We make a distinction for instance granularity based on whether an instance in a learning problem is a document (several sentences) or a single sentence. The labeled data is composed of instances and their class labels, with or without feature feedback. As discussed in Section 5.2, we assume that annotating an instance with feature feedback takes on average twice as much time as annotating an instance without feature feedback, based on the findings in prior work (Zaidan et al., 2007). We measure the labeled data observed by the model so far in terms of the number of annotation cost units spent, which may mean different number of labeled instances based on the annotation strategy (with or without feature feedback). We use two feature configurations of "unigram only" and "unigram+dependency triples". The unigram and dependency annotations are derived from the Stanford Dependency Parser (Klein and Manning, 2003).

Rationales, as defined by Zaidan et al. (2007), are spans of text in a review that convey the sentiment of the reviewer. Hence, they are the parts of the document most relevant for the classification task. In order to vary the amount of irrelevant text in an instance, we vary the amount of text (measured in terms of the number of characters) around the rationales in an instance. We call this the *slack* around rationales. When using the rationales with or without the slack, only features that overlap with the rationales (and the slack) are used to represent an instance. ∞ slack means all the text in an instance is used for feature representation, like in the original document. Since we only have rationales for the movie review documents, we only studied the effect of varying the amount of irrelevant text on this dataset.

Variable	**Possible Values**
Domain (D)	{Movie Review classification (MR), Webpage classification (WebKB)}
Instance Granularity (G)	{document (doc), sentence (sent)}
Feature Space (F)	{unigram only (u), unigram+dependency (u+d)}
Labeled Data (#AUs) (L)	{64, 128, 256, 512, 1024}
Irrelevant Text (I)	{0, 200, 400, 600, ∞ }
Instance Selection Strategy (S))	{deterministic (deter), uncertainty (uncert)}

Table 5.1: Experiment space for analysis of learning problems ($P = \{D, G, F, L, I, S\}$)

For all our experiments, we used Support Vector Machines (SVMs) with linear kernel for learning (libSVM (Chang and Lin, 2001) in Minorthird (Cohen, 2004)). For identifying the discriminative features for computing the feature complexity measures, we used the information gain score. For all datasets we used 1800 total instances with equal number of positive and negative instances. We held out 10% of the data for estimating the model's uncertainty, as explained in Section 5.1. The results we present are averaged over 10 cross validation folds on the remaining 90% of the data (1620 instances). For cross validation fold, one fold is used for testing (162 instances), and all of the remaining 1458 instances are used as the 'oracle' for calculating the feature complexity measures and estimating the maximum benefit from feature feedback as discussed in Sections 5.1 and 5.2 respectively. The training data size is varied from 64 to 1024 instances (from the total of 1458 instances for training in a fold), based on the annotation cost budget. Instances with their label are added to the training set either in the original order they existed in the dataset, i.e. no selective sampling (deterministic), or in the decreasing order of current model's uncertainty on them. Uncertainty sampling in SVMs (Tong and Koller, 2000) selects the instances closest to the decision hyperplane, since the model is expected to be most uncertain about these instances. In each slice of the data, we ensured that there is an equal distribution of the positive and negative class. SVMs do not yield a probabilistic output, but a decision hyperplane. We use the common practice of fitting a sigmoid curve to the decision values from SVMs (distance from the decision hyperplane), to estimate the probability of the predicted class (Platt, 1999).

5.3.2 Results and Discussion

To determine the effect of various factors on benefit from feature feedback, we did an ANOVA analysis with Generalized Linear Model, using a 95% confidence interval. The top part of Table 5.2 shows the overall average $F1$ score for the two annotation strategies. As can be seen, on average with feature feedback, we get a significant improvement in performance.

Next we analyze the effect of various problem characteristics discussed above, on benefit from feature feedback, measured in terms of the improvement in performance (IP) at a given annotation cost and the improvement in learning rate (ILR). As discussed in Section 5.2, the improvement in learning rate with feature feedback is calculated by comparing the learning profile for the two annotation strategies with increasing amount of labeled data, up to a maximum annotation cost of 1024 AUs.

As can be seen from the second part of Table 5.2, most of the factors have a significant effect on the benefit from feature feedback. The benefit is significantly higher for the webpage classification task than the sentiment classification task. We found that the average feature complexity for the webpage classification task ($N_f = 3.07$) is lower than the average feature complexity for the sentiment classification task ($N_f = 5.18$), for 1024 training examples. Lower feature complexity suggests that the webpage classification concept can be expressed with a few keywords such as *professor*, *faculty*, etc., and with feature feedback we can quickly identify these features. Sentiment on the other hand can be expressed in a variety of ways, which explains the higher feature complexity.

The benefit is more for document granularity than sentence granularity, which is intuitive as the feature space will be substantially larger for documents, and we expect to gain more from the user's feedback on which features are important. This difference is significant for the improvement in the learning rate and marginally significant for the improvement in performance. Note that here we are comparing all documents (with or without rationale slack, and from both domains) and sentences. However, documents with low rationale slack should have a similar amount of noise as a sentence. Also, a significant difference between the domains suggests that documents in WebKB domain might be quite different from those in Movie Review domain. This may explain the marginal significant difference between absolute improvement in performance for documents and sentences. To understand the effect of granularity alone, we compared the benefit from feature feedback for documents (without removing any noise, i.e. ∞ slack) and sentences, in movie review domain only. However, we found that this difference is not significant. Thus, contrary to our intuition, sentences and documents seem to benefit equally from feature feedback, when measured in terms of the absolute improvement in performance.

The benefit is more when the feature space is larger and more diverse, i.e., when dependency features are used in addition to the unigram features. We found that on average adding dependency features to unigram features increases the feature space by approximately 10 times. With larger feature space, feature feedback can help to identify a few relevant features. As can also be seen, feature feedback is more helpful when there is more irrelevant text, i.e., there is noise that feature feedback can help to filter out. Unlike the improvement in performance, the improvement in learning rate does not decrease monotonically as the amount of rationale slack decreases. This supports our belief that improvement in performance does not imply improvement in the learning rate and vice versa. We saw a similar result when comparing the benefit from feature feedback at different instance granularities. The difference in the improvement in learning rate for problems with different granularity was statistically significant. However, the difference in the improvement in performance at given annotation cost was not significant. Thus, both metrics should be used when evaluating the benefit from feature feedback.

We also observe that when training instances are selectively sampled as the most uncertain instances, we gain more from feature feedback than without selective sampling. This is intuitive, since the instances the model is most uncertain about are likely to contain features it is uncertain about, and hence the model should benefit more from feedback on features in these instances. Next we evaluate how well the measures for categorizing learning problem discussed in Section 5.1 correlate with the improvement in performance and improvement in learning rate with feature feedback.

$Var.$	$Values$	$AvgF1$	Group
Strat.	wFF	78.2	A
	woFF	68.2	B

$Var.$	$Values$	Avg_{IP}	Grp_{IP}	Avg_{ILR}	Grp_{ILR}
D	WebKB	11.9	A	0.32	A
	MR	8.0	B	0.20	B
G	Doc	10.9	A	0.30	A
	Sent	9.0	A	0.22	B
F	u+d	12.1	A	0.30	A
	u	7.8	B	0.22	B
I	∞	12.8	A	0.34	A
	600	11.2	A B	0.23	B
	400	11.1	A B	0.26	A B
	200	9.8	B	0.26	A B
	0	4.8	C	0.21	B
S	Uncer.	12.7	A	0.32	A
	Deter.	7.1	B	0.20	B

Table 5.2: Effect of variables defined in Table 5.1 on the maximum benefit from feature feedback. Avg_{IP} is the average (over 10 folds) increase in performance ($F1$) and Avg_{ILR} is the average increase in the learning rate. Different letters in Grp_{IP} and Grp_{ILR} indicate significantly different results.

For a given problem with an annotation cost budget of a AUs, we calculate the benefit from feature feedback by comparing the performance with feature feedback on $\frac{a}{2}$ instances and the performance without feature feedback on a instances, as described in Section 5.2. The feature complexity measures are calculated using $\frac{a}{2}$ instances, since it should be the characteristics of these $\frac{a}{2}$ training instances that determine whether we would benefit from feature feedback on these $\frac{a}{2}$

instances, or from labeling new $\frac{a}{2}$ instances. As can be seen from Table 5.3, the correlation of feature complexity measures with both measures of benefit from feature feedback, is strong, negative and significant. This suggests that problems with low feature complexity, i.e., concepts that can be expressed with a few well-selected features, benefit more from feature feedback.

We expect the benefit from feature feedback to decrease as the amount of labeled data increases. We found a significant negative correlation (-0.574) between annotation budget (number of AUs), and the improvement in performance with feature feedback. However, note that this correlation is not very strong, which supports our belief that factors other than the amount of labeled data affect the benefit from feature feedback.

Measure	$\mathbf{R}(IP)$	$\mathbf{R}(ILR)$
N_f	-0.625	-0.615
F_{pc}	-0.575	-0.735
C_f	-0.603	-0.629

Table 5.3: Correlation coefficient (R) for feature size complexity (N_f), feature profile complexity (F_{pc}) and combined feature complexity (C_f) with improvement in performance (IP) and improvement in learning rate (ILR). All results are statistically significant ($p < 0.05$)

As mentioned before, the feature complexity measures from (Raghavan et al., 2007) require an 'oracle', simulated using a large amount of labeled data, which is not available *a priori* for new annotation tasks. Thus, these measures are good for understanding why feature feedback helps more for certain learning problems, and not so much for the others. However, these measures cannot be used for predicting whether and how much benefit there would be from feature feedback, for a new annotation task. In Section 5.1, we proposed a simple measure based on model's uncertainty, that does not require an oracle. We calculate the mean, maximum and median of the probability of the predicted class from the learned model, for instances in the held out dataset. We found a significant but low negative correlation of these measures with the improvement in performance with feature feedback ($maxProb = -0.384$, $meanProb = -0.256$, $medianProb = -0.242$). The low correlation here may seem counterintuitive. However, note that when the training data is very small, the model might be quite certain about its predictions, even when it is wrong, and feature feedback may help by correcting the model's beliefs. We observed that these probability measures have only a medium positive correlation (around 0.5) with training datasize. Also, the held out dataset we used may not be representative of the whole set, and using a larger dataset may give us a more accurate estimate of the model's uncertainty.

5.4 Related Work

There is little work in the literature that analyzes how the benefit from feature feedback varies across learning problems. To the best of our knowledge there is only one work in this area by Raghavan et al. (2007). They categorize learning problems in terms of their *feature complexity*, as discussed in Section 5.1. In order to evaluate the benefit from feature feedback, Raghavan et al. (2007) use their tandem learning approach of interleaving instance and feature feedback (Raghavan et al., 2006), referred to as interactive feature selection (ifs). The features are labeled as 'relevant' (feature discriminates well among the classes), or 'not-relevant/don't know'. The labeled features are incorporated into learning by scaling the value of the relevant features by a constant factor in all instances. The benefit from feature feedback is measured in terms of the gain in the learning speed. The learning speed measures the rate of performance improvement with increasing amount of supervision. It is defined in terms of the convergence profile similar to the feature learning convergence profile in Equation 5.1, except in terms of the number of labeled units instead of the number of features. A labeled unit is either a labeled instance or an equivalent set of labeled features, with the same annotation cost. The benefit from feature feedback is then measured as the difference in the convergence profile with interactive feature selection (p_{ifs}) and with labeled instances only (p_{al}).

Raghavan et al. (2007) analysed 9 corpora and 358 binary classification tasks. Most of these corpora, such as Reuters (Lewis, 1995), 20-newsgroup (Lang, 1995), etc., have topic-based category labels. They observed a negative correlation ($r = -0.65$) between the benefit from feature feedback and combined feature complexity measure (C_f), described in Section 5.1. That is, feature feedback accelerates active learning by an amount that is inversely proportional to the feature complexity of the problem. In other words, if a concept can be expressed using a few well-selected features from a large feature space, we stand to benefit more from feature feedback, since a few labeled features can provide this information. On the other hand, if learning a concept requires all or most of the features in the feature space, there is little knowledge that feature feedback can provide.

Most of the datasets analyzed by Raghavan et al. (2007) were topic classification datasets. They found that the webpage classification task among all has low feature complexity, and benefited the most from feature feedback. We presented our results on this task, and a new domain of sentiment classification. Raghavan et al. (2007) only varied the domain among different problems they analyzed, i.e, only the variable D in our problem definition ($P = \{D, G, F, L, I, S\}$). However, we found that other characteristics such as feature representation, amount of labeled data, amount of irrelevant text and instance selection strategy, have a significant effect on the maximum benefit from feature feedback. We applied the feature complexity measures proposed in (Raghavan et al., 2007) to problems that differ in these characteristics, in addition to the domain. In agreement with their results, we found that the feature complexity measures have a strong negative correlation with the benefit from feature feedback.

Analysis in (Raghavan et al., 2007) is specific to their approach for incorporating feature feedback into the model, which may not work well for all domains and datasets, as also mentioned in their work (Section 6.1 in (Raghavan et al., 2007)). It is not clear how their results can be extended to alternate methods for seeking and incorporating feature feedback. Thus, in this work, we analyzed how the expected maximum benefit from feature feedback varies across learning problems.

Raghavan et al. (2007) analyze benefit from feature feedback at a fixed training data size of 42 labeled units. However, the difference between learning problems may vary with the amount of labeled data. Some problems may benefit significantly from feature feedback even at relatively larger amount of labeled data. On the other hand, with very large training set, the benefit from feature feedback can be expected to be small and not significant for all problems. Thus, we evaluated the benefit from feature feedback at different amount of labeled data ({64, 128, 256, 512, 1024}).

For all classification tasks, Raghavan et al. (2007) used a simple and fixed feature space, containing only unigram features (n-gram features were added where it seemed to improve performance). In this work, we also evaluated the benefit from feature feedback for features extracted from the dependency parse trees, in addition to the unigram features.

Raghavan et al. (2007) evaluate the benefit from feature feedback in terms of the gain in learning speed. However, as mentioned before, the learning rate does not tell us how much improvement we get in performance at a given stage in learning. In fact, even if at every point in the learning curve, the performance with feature feedback is lower than the performance without feature feedback, the rate of convergence to the corresponding maximum performance may still be higher, when using feature feedback. Thus, in this work, in addition to evaluating the improvement in the learning speed with feature feedback, we also evaluated the improvement in the absolute performance at several points in the learning curve.

5.5 Summary and Future Work

In this work, we presented an analysis of how the maximum benefit from feature feedback varies across learning problems. The maximum benefit is estimated using an oracle, simulated using a large amount of labeled data. We categorize a learning problem in terms of its domain, instance granularity, feature representation, amount of labeled data, amount of irrelevant text, and instance selection strategy. We show that most of these characteristics have a significant effect on benefit from feature feedback, measured in terms of the absolute improvement in performance, and improvement in the learning rate. The benefit from feature feedback is more for a simpler task of webpage classification, than sentiment classification. The benefit is more when structured features are used in addition to simple unigram features. Also, the benefit is more when there is more irrelevant text in the instances, and when instances for annotation are selected based on model's uncertainty.

We evaluate the feature complexity measures proposed by Raghavan et al. (2007) on problems that vary in the above characteristics. We find a significant negative correlation of their measures with the two measures of maximum benefit from feature feedback, proposed in this work. Their measures require an oracle, which is not available for new annotation tasks. We suggest simple measures based on model's uncertainty, which do not require an oracle. We find a significant but low correlation of these measures with the maximum benefit from feature feedback. It is intuitive that a metric that measures the uncertainty of the model on parameter estimates should also correlate strongly with the benefit from feature feedback. Variance in parameter estimates is one measure of model's uncertainty on them. The Bootstrap or Jacknife method (Efron and Tibshirani, 1994) of resampling from the training data is one way of estimating the variance in parameter estimates. In Chapter 3, we proposed a VOI based active selection criteria based on the expected reduction in model's misclassification risk with indirect feature feedback from rationales and the expected cost for annotating rationales, to automatically determine when it is beneficial to ask for rationales.

So far only a linear relationship of the measures for categorizing learning problems with benefit from feature feedback has been considered. However, some of these relationships may not be linear or a combination of several measures together may be stronger indicators of the benefit from feature feedback. A future direction is to consider these alternate relationships.

We only considered one selective sampling strategy based on model's uncertainty which we found to provide more benefit from feature feedback than deterministic sampling. It will be interesting to study how the benefit from feature feedback varies with other selective sampling strategies. For example, density-based sampling (Donmez and Carbonell, 2008a) selects the instances that are representative of a cluster of similar instances, and may facilitate more effective feedback on a diverse set of features. Other alternatives to study are random instance selection and VOI based instance selection proposed in Chapter 3.

Chapter 6

Conclusions and Future Work

In this chapter, we summarize the main contributions of the thesis, and suggest directions for the future work.

6.1 Contributions of the Thesis

There are several costs associated with supervised annotation learning, such as the cost of obtaining unlabeled data, cost of feature engineering, the cost of expert annotation, etc. In this thesis, our goal is to reduce the total cost of supervised annotation learning. Specifically, we focus on reducing the cost of feature engineering and the total annotation cost for the desired performance. The novel contributions of this thesis, discussed in previous chapters, are as follows:

- **Generalized Automatic Feature Extraction:** In supervised annotation learning, features based on prior linguistic annotations, such as parts of speech, dependency relations, etc., have been shown to improve performance beyond bag-of-words features (Gildea and Jurafsky, 2000; Pradhan et al., 2004; GuoDong et al., 2005; Wilson et al., 2004; Arora et al., 2009a; Gamon, 2004; Joshi and Rosé, 2009). Often these features are hand-crafted by the domain experts, who are expensive to hire, and the process has to be repeated for new domains and tasks. In Chapter 2, we proposed a generic annotation graph representation for instances based on the prior annotations, and an automatic approach for extracting features as patterns from the annotation graphs. For a sentiment classification task and a protein-protein interaction extraction task, we showed that automatically extracted structured features using the proposed approach provide a significant improvement in performance over the bag-of-words features.

- **Indirect Feature Feedback Through Rationales:** To reduce the total annotation cost for the desired performance, alternate forms of feedback from the annotator beyond instance's label can be used, for example, feedback on features. Direct feedback on features (Godbole et al., 2004; Raghavan and Allan, 2007; Druck et al., 2009; Melville and Sindhwani, 2009) is limited to simple features, such as words. Instead, we seek indirect feedback on features by asking the annotator to indicate parts of an instance that are *rationales*, i.e. key indicators, for the instance's label. The additional cost for annotating rationales may vary with the annotation task, annotator, user interface design, etc. In Chapter 3, for a sentiment classification task and an aviation incident cause identification task, we showed that rationales provide a significant improvement in performance for a given annotation cost, when the cost for annotating them is small (up to 50% additional cost).

- **Cost-sensitive Active Learning Framework:** In this thesis, we studied two annotation strategies, where the annotator provides instance's label only (LO), or instance's label together with rationales (LR). The annotation cost may vary across instances and strategies. Each instance (with or without rationales) may provide different incremental value to the learning algorithm. In Chapter 3, we proposed a cost-sensitive active learning approach that selects an instance for a given strategy. For a sentiment classification task and an aviation incident cause identification task, we showed that this strategy sensitive instance selection for seeking rationales performs better than random or active instance selection independent of the strategy. When rationales are expensive to annotate, a fixed strategy of rationales for all instances (LR), performs worse than rationales for none (LO). We use the proposed approach to selectively ask for rationales by jointly selecting an instance and annotation strategy in each iteration. While the best fixed strategy among LO and LR varies with the cost for annotating rationales, we showed that cost-sensitive joint selection of instance and strategy performs as well as or better than the best fixed strategy, for several additional costs for annotating rationales. It also performs better than cost-sensitive and cost-insensitive instance selection for a fixed annotation strategy, and cost-insensitive joint selection of instance and strategy.

- **Annotation Cost Estimation:** In order to directly minimize the total annotation cost for the desired performance, an estimate of the annotation cost can be used in the active selection criteria. Annotation cost may vary with the instance, annotator and annotation strategy. In Chapter 3, for cost-sensitive joint selection of instance and strategy, we used a simple estimate of annotation cost based on the instance length and annotation strategy. However, the annotation cost may also depend on other factors, such as the number of rationales annotated, annotator characteristics in a multi-annotator scenario, etc. In Chapter 4, we proposed a supervised regression model for estimating the annotation cost in a multi-annotator scenario with indirect feedback on features through rationales. For data collected from multiple annotators for a sentiment classification task, we showed that an annotation cost estimate from a model based on the characteristics of an instance, annotation task and annotator outperforms a simpler estimate based on any one of them. In an active learning scenario, this supervised regression model can be trained on the data annotated so far and can be updated with the new data collected in each iteration.

- **Assessment of the Benefit from Feature Feedback:** The benefit from feature feedback (direct or indirect) may vary across learning problems. In Chapter 5, we formally defined a learning problem in terms of the domain, instance granularity, feature representation, amount of labeled data, amount of irrelevant text, and the instance selection strategy. For two sentiment classification tasks with different instance granularity and a webpage category classification task, we showed that all characteristics except instance granularity have a significant effect on the maximum benefit from feature feedback. The benefit from feature feedback is more when there is more irrelevant text in the instances, the feature space is large, and when instances are selected as those the model is uncertain about, compared to a fixed order of instances. Instances at different granularity (document vs. sentences) seem to benefit equally from feature feedback. We also showed that the measures for quantifying the feature complexity of learning problems (Raghavan et al., 2007) correlate strongly and negatively with the expected maximum benefit from feature feedback. That is, learning problems with low feature complexity benefit more from feature feedback, since a few well selected features are able to capture the target concept.

Figure 1.1 in Chapter 1 presented a traditional supervised learning scenario and the associated costs. As mentioned before, in this thesis, our goal is to reduce the total cost of supervised annotation learning. Specifically, we focus on reducing the cost of feature engineering, and the total annotation cost for the desired performance. Figure 6.1 presents an improved interactive learning framework with reduced total costs, due to the proposed enhancements in this thesis. In the traditional approach, an expert defines complex linguistic features by hand and/or through extensive experimentation. In the proposed approach, we reduce this cost by automatically deriving these features from prior linguistic annotations (Steps 3.1 to 3.3). The expert specifies the annotation graph representation for instances based on the prior annotations suitable for the domain. The features are extracted as frequent subgraphs from the annotation graphs for the instances.

In traditional supervised annotation learning, there is a single annotation strategy of providing instance's label only. In this thesis, we propose an alternate strategy where in addition to providing the instance's label, the annotator indicates parts of an instance that are rationales for the instance's label. In order to reduce the total annotation cost for the desired performance, instance and strategy for next annotation are actively selected in each iteration. The annotated instance is added to the pool of labeled data, and the classifier is updated. Steps 4 to 6 are repeated until the desired performance is reached, or we exhaust our budget.

6.2 Future Work

There are several directions in which the work in this thesis can be extended. We discuss some of them here.

6.2.1 Interactive Learning System

In this thesis, we proposed several ways to reduce the total cost for supervised annotation learning. Figure 6.1, discussed above, presents the workflow of an interactive learning system. In this thesis, we have developed several components for such an interactive learning system. Building a functional interactive learning system would require integrating these components together. Figure 6.2, presents a sequence diagram that suggests an interaction protocol between components of an Interactive Annotation Learning (IAL) System. The functions in this sequence diagram would be part of the abstract interfaces for these components. The four main components are the following:

1. Graphical User Interface (GUI): The user interface for seeking annotations.
2. Cost Sensitive Multi-Strategy Active Learner (CSMSAL): Active learner for selecting the instance and strategy to be annotated in each iteration.
3. Annotation Cost Estimator (ACE): Annotation cost estimator for a given instance, strategy and annotator.
4. DBManager: Database manager for storing and retrieving the data and annotations from the database.

In each iteration, the GUI queries CSMSAL for the next annotation task for a given annotator. CSMSAL retrieves the unlabeled data from the database, and queries ACE for an annotation cost estimate for all instances and strategies for the given annotator. CSMSAL then calculates VOI and selects the best instance and strategy pair for the given annotator. The GUI fetches the selected instance and presents it to the annotator for annotation with the selected strategy. When done annotating the instance, the annotator triggers a save action. The GUI writes the annotated

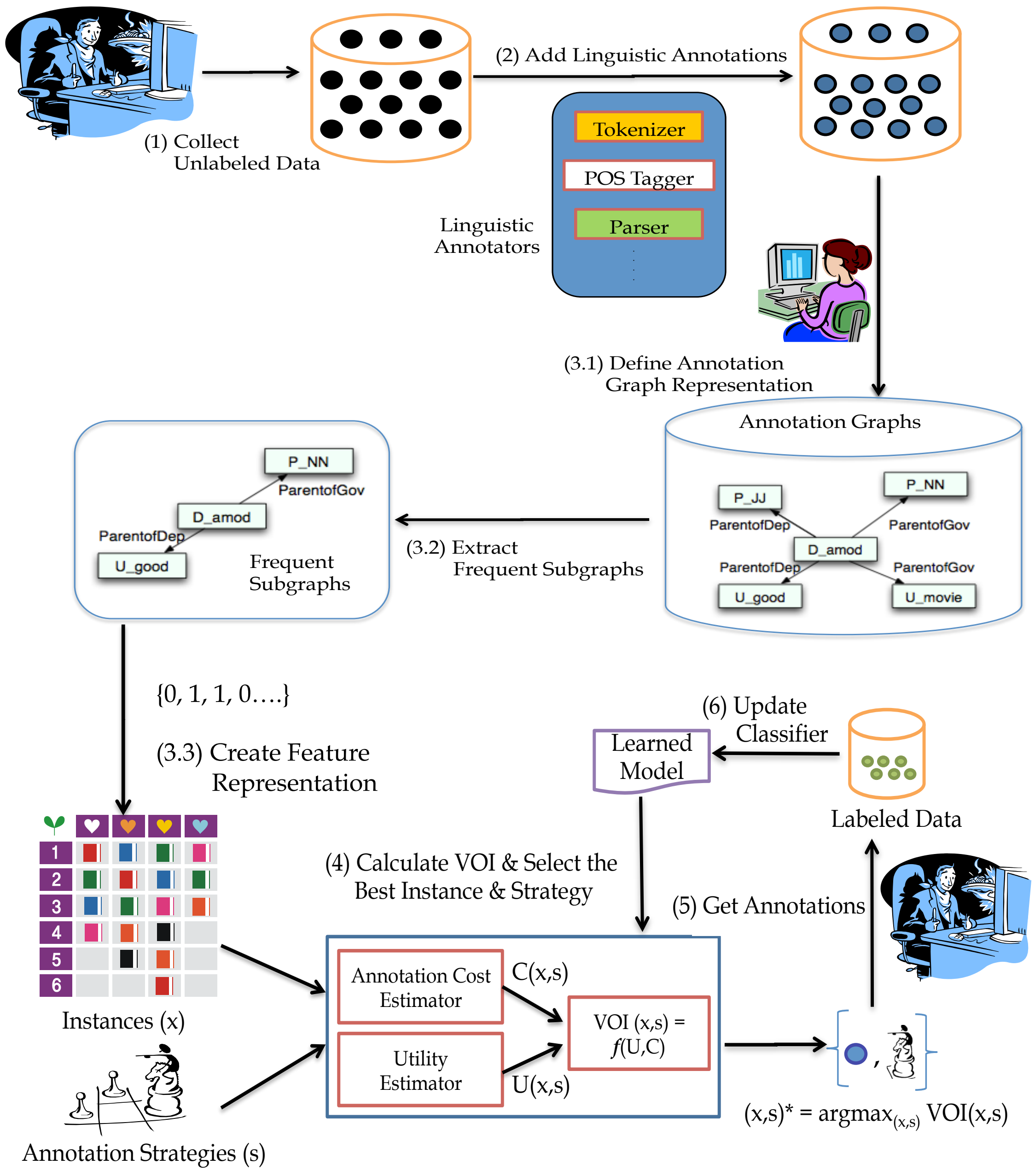


Figure 6.1: Interactive annotation learning with reduced total cost for supervised learning.

instance and annotation time information to the database. It updates the total budget spent, and triggers CSMSAL and ACE to update their models with information about the annotated instance. These steps are repeated until we exhaust our annotation budget.

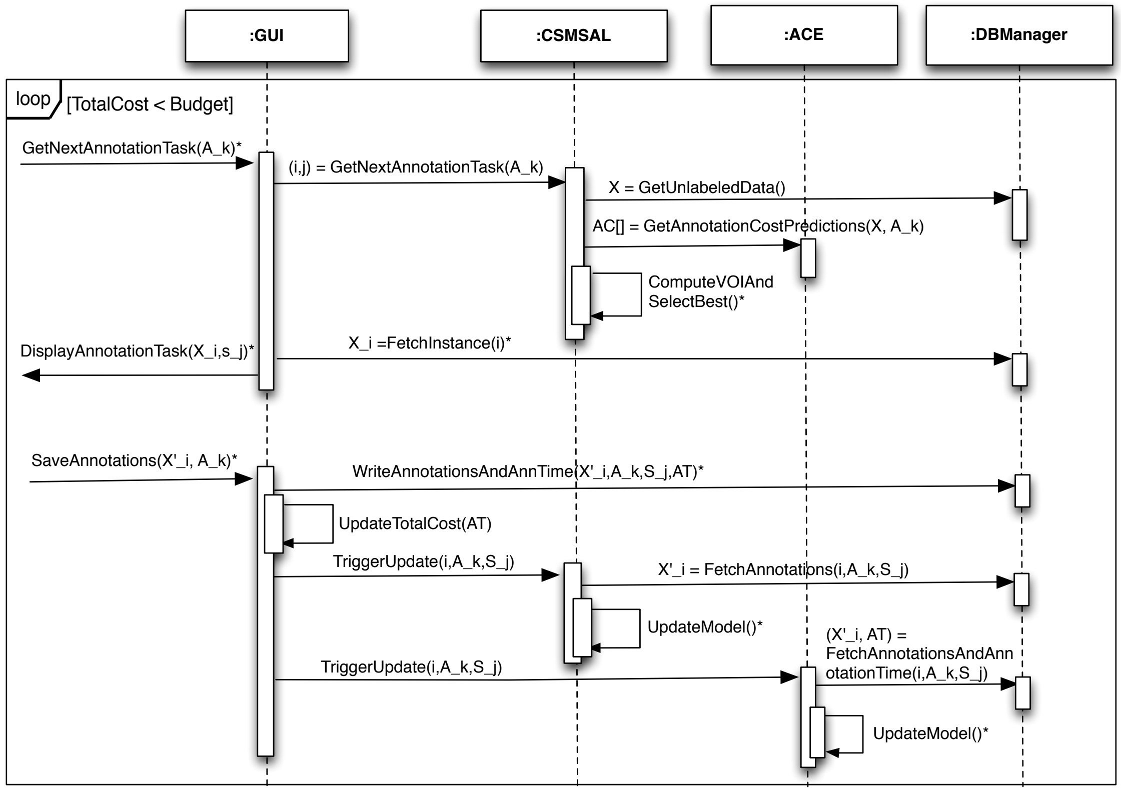

:GUI
:CSMSAL
:ACE
:DBManager
loop [TotalCost < Budget]
GetNextAnnotationTask(A_k)*
(i,j) = GetNextAnnotationTask(A_k)
X = GetUnlabeledData()
AC[] = GetAnnotationCostPredictions(X, A_k)
ComputeVOIAnd SelectBest()*
DisplayAnnotationTask(X_i,s_j)*
X_i =FetchInstance(i)*
SaveAnnotations(X'_i, A_k)*
WriteAnnotationsAndAnnTime(X'_i,A_k,S_j,AT)*
UpdateTotalCost(AT)
TriggerUpdate(i,A_k,S_j)
X'_i = FetchAnnotations(i,A_k,S_j)
UpdateModel()*
TriggerUpdate(i,A_k,S_j)
(X'_i, AT) = FetchAnnotationsAndAnnotationTime(i,A_k,S_j)
UpdateModel()*

Figure 6.2: Sequence Diagram for an Interactive Annotation Learning (IAL) System. GUI: Graphical User Interface, CSMSAL: Cost Sensitive Multi-Strategy Active Learner, ACE: Annotation Cost Estimator, DBManager: Database Manager. `A_k`: Annotator, `X_i`: Selected instance, `S_j`: Selected strategy, X: Unlabeled data, `AC[]`: Annotation cost estimates for instance and strategy pairs for a given annotator, `X'_i`: Annotated instance, `AT`: Annotation Time. Functions marked with an asterisk are already implemented in the current implementations of the abstract interfaces for these components.

For this thesis, we have implemented the four abstract interfaces in Figure 6.2 in Java. For a practical interactive learning system, it is important to use an incremental algorithm to update the classifier in each iteration. A batch learning algorithm, used in traditional supervised learning framework, would train a new model from scratch in each iteration. This can be very slow in practice for an interactive learning system. Additionally, the proposed VOI approach for active selection of instance and strategy requires estimating the reduction in risk by learning and unlearning for each candidate instance and strategy, which can be computationally very expensive with a batch learning algorithm. We have implemented an incremental and decremental SVM algorithm (Cauwenberghs and Poggio, 2000) in Java, based on a C++ implementation by Vijayanarasimhan and Grauman (2011). This can be integrated with our user interface in Java (used in Chapter 4 for data collection) and annotation cost estimator (ACE) in Java (developed for experiments in Chapter 4), to build an interactive learning system. We also have a database manager in Java, with which the user interface communicates to fetch and store data and annotations.

In Figure 6.2, functions marked with an asterisk are already implemented in the current implementations of the abstract interfaces. The GUI does not currently display the annotation strategy to use, although this can be easily added. CSMSAL does not currently interact with the DBManager to fetch the data. It instead loads the data in memory for active selection. In the current implementation, CSMSAL estimates the annotation cost for a given instance and strategy itself, and uses a simple estimate described in Chapter 3. ACE estimates the annotation cost for a given instance and annotator for a fixed strategy of LR (as described in Chapter 4). It uses the number of rationales as a feature. An estimate for the number of rationales can be used from the R vs. NR classifier in CSMSAL (for LO strategy we would use 0 rationales). ACE currently uses a batch regression (linear regression or SVM regression) model. We may want to use an incremental learning algorithm (e.g. `http://onlinesvr.altervista.org/`) for a faster model update. Communication between CSMSAL and DBManager and ACE and DBManager would also need to be added. To build an end-to-end interactive learning system from current implementations, we anticipate that it would take about 2-3 months.

One of the challenges for a real-time interactive learning system is the amount of time the user must wait while the model is updated and the next instance (and strategy) is selected. As mentioned in Chapter 3, with our current implementation of the VOI framework, it takes on average 44 seconds to select an instance and strategy for the movie review dataset. For the aviation safety reports data, which is considerably smaller, it takes only 0.1 seconds. The time for active selection is less in the initial iterations, since model updates are faster when the model is simpler. As the model becomes more complex, the model updates require more time. Annotator wait-time should be taken into account when calculating the annotation cost savings with an interactive learning system.

One way to reduce the time for instance selection is to parallelize the algorithms for model update and candidate scoring procedures. There has been some recent work in parallelizing machine learning algorithms, for example, GraphLab (Low et al., 2010). Another alternative is to score only a subset of instances in each iteration, like in our work and other work in the literature (Roy and Mccallum, 2001; Haertel et al., 2010). However, if the candidate set is too small, then we may not benefit much from active learning. Another approach is to allow some staleness in scores for some candidates (Haertel et al., 2010). In the work so far, the active learner selects a single annotation task (instance and strategy) in each iteration. To reduce the overall wait time, we may consider selecting a small batch of annotation tasks in each iteration. However, this would require evaluating all subsets of candidates of a given batch size, which can be computationally very expensive. A simple heuristic is to select the top N ranked candidates from a single ranking of the candidates in each iteration, although this approach is sub-optimal. There has been work in the literature on selecting optimal batch of queries in each iteration, for example, (Vijayanarasimhan et al., 2010).

6.2.2 Multiple Strategies with Different Number of Rationales

In this thesis, we only considered the two annotation strategies of providing instance's label only and instance's label with almost all rationales. In Chapter 3, we presented an initial investigation of the effect of number of rationales on the model's performance. There can be at least two ways to ask for a fixed number of rationales per instance. The annotator can be asked to provide *any* N rationales or the *top* N most important rationales. Considering the *any* N rationale strategy, we found that for random instance selection the performance usually improves with more rationales, although there is no significant benefit from more than 8 rationales per instance As learning progresses, we found that there is no significant benefit from more than 4 rationales per instance. Thus, if each additional rationale comes with an extra cost, we may want to ask for no more than 8 rationales per instance, and as learning progresses, we may want to ask for fewer rationales. The optimum number of rationales may vary across instances. In Chapter 3, we suggested how the current VOI framework may be extended to multiple annotation strategies with different number of rationales. It is not clear how the annotation cost would vary with N when asked to provide *top* N rationales, since the annotator would spend some time ranking the rationales (in their minds) in the order of importance. However, the annotation cost should increase with more rationales when asked to provide *any* N rationales. The cost estimator proposed in Chapter 4 can be used to estimate the annotation cost for an annotation strategy with *any* N rationales, since it considers the number of rationales in annotation cost estimation. As also discussed in Chapter 3, for each strategy with a fixed number of rationales, we can estimate the utility as the expected reduction in model's risk with the given number of rationales. To estimate the expected risk after training on instance X_i annotated with top N rationales, we may use the *top* N sub-instances predicted to be rationales (ranked by the model's confidence). If instead the annotator is asked to provide *any* N rationales, we may estimate the expected risk as the average risk for k random samples for N rationales from the sub-instances predicted as rationales. As also pointed out in Chapter 3, we would need to modify the current rationale classifier (R vs. NR), used for estimating the risk for LR strategy, when multiple strategies with a fixed number of rationales are used. With almost all rationale strategy, the current rationale classifier treats a sub-instance in an instance not marked as rationale, as not-rationale (NR). However, if all the rationales are not annotated in an instance, then we cannot treat all remaining sub-instances (not annotated as rationale) as NR. One approach to address this issue is to use the current model's belief about which remaining sub-instances (those that are not marked as rationale) in an instance are likely to be not-rationale and train the classifier with only those labeled as NR.

6.2.3 Multi-Annotator Scenarios

In Chapter 3, we showed that the proposed approach for joint selection of instance and strategy performs as well as or better than the best fixed strategy, for several additional costs for annotating rationales. The difference between the annotation costs for the two strategies (called the rationale cost factor) may vary across annotators. In this thesis, we focused on selecting the appropriate annotation task (instance and strategy) for a given annotator, and showed that the proposed active learning approach is able to select the appropriate instance and strategy pair for an annotator with a given rationale cost factor. We did not consider a scenario where we may select an annotator from a pool of annotators with different characteristics, and a limited budget per annotator. There are several multi-annotator scenarios with annotator selection to which the current framework can be extended. We discuss some of these here and suggest how we may extend the current active learning framework for these scenarios:

1. **Reliable annotators with known annotation cost**: If all annotators are experts and reliable (i.e. provide accurate annotations), and their annotation cost for different strategies is known or can be estimated reliably from a small sample of annotated data, then the current framework can be applied straightforwardly to a multi-annotator scenario with annotator selection. The annotation task can be assigned to the annotators in a round robin manner, selecting the annotator with lowest annotation cost for a given strategy.

2. **Annotators with different reliability**: Annotators may have different expertise and may not provide accurate annotations. If the annotators are unreliable, we would not want to use annotations from a single annotator for an instance, as it may not be correct. If each annotator's accuracy is > 0.5, i.e. better than random, we can use annotations from multiple

annotators to determine the final label for an instance as the majority label (Donmez et al., 2009), since it is unlikely that all annotators will make a labeling mistake at the same time. Similarly, we may determine the final label (rationale or not) for a sub-instance in an instance as the majority label based on rationale annotations from the annotators in agreement with the majority label for the instance. Donmez et al. (2009) propose an approach to identify the most reliable annotators and only query these selected annotators instead of querying everyone. The annotator accuracy may not be known beforehand. However, it may be estimated from the data annotated so far. A simple estimate for annotator's accuracy is the mean reward, where an annotator is assigned a reward for each correct answer (when annotator's label matches the majority) (Donmez et al., 2009). Estimating the annotator accuracy from the data labeled so far and using this estimate to select the annotators for future annotation requires balancing between *exploitation* of the current best known annotators, and *exploration* of new annotators we may be uncertain about. To address this exploration vs. exploitation tradeoff, Donmez et al. (2009) use the mean and standard deviation of the rewards received so far to calculate the upper confidence interval (UI) for the reliability for each annotator, given by:

$$UI(a) = m(a) + t_{\alpha/2}^{n-1} \frac{s(a)}{\sqrt{n}} \tag{6.1}$$

where $m(a)$ and $s(a)$ are the sample mean and standard deviation of the rewards for annotator a. n is the number of samples observed from a, and $t_{\alpha/2}^{n-1}$ is the critical value for the Student's t-distribution with $n-1$ degrees of freedom, at $\alpha/2$ confidence level. An annotator has a high value for the upper confidence interval when the expected reward (the mean value) is high and/or there is a lot of uncertainty in the reward (when the standard deviation is high). The uncertainty may be more when we don't have enough labeled data for this annotator, i.e., the annotator hasn't been selected enough times to give a reliable estimate. Selecting an annotator with a high expected reward performs exploitation, and selecting an annotator with a high uncertainty performs exploration. Using the upper confidence interval value instead of the mean as an estimate for the annotator accuracy, allows us to select an annotator with high expected accuracy, and/or with high variance in his/her estimate.

For the scenario where the annotators may provide rationales in addition to the instance's label, we may estimate an annotator's accuracy only from the labels for instances, or we may also consider how well their rationales match with the rationales from other annotators.

The probability of obtaining the true label for an instance from an annotator, i.e. annotator reliability, may also vary across instances, for example, based on the difficulty of an instance (Whitehill et al., 2009). We may want to take this into account when estimating the annotator reliability. Annotator accuracy may also vary over time (Donmez et al., 2010). The annotator may produce more reliable annotations after some experience with the task, or may become careless due to fatigue after labeling several instances. We may also want to take into account the effect of time when estimating the annotator accuracy.

In the works discussed above, the set of annotators expected to be most reliable is chosen in each iteration. An alternative approach is to select a single annotator at a time. In each iteration, we have a choice between querying another annotator for label for an instance whose final label we are uncertain about and querying a label for a new instance. This has recently been explored in the literature (Yan et al., 2012).

When we have a large pool of annotators, for example, at a crowd sourcing site, we may spend substantial budget in exploring new annotators, until we have a fair idea about their reliability. To address this issue, we may pre-select a subset of annotators based on their known reliability on other tasks they have participated in at the crowd sourcing site, for example, *approval rating* of a Turker on Amazon's Mechanical Turk[1].

3. **Reliable annotators with different and unknown cost**: Annotation cost may also vary across annotators. It may be unknown and may not be reliably estimated beforehand from a small sample of annotated data. In Chapter 4, we proposed a supervised regression model for estimating the annotation cost in a multi-annotator scenario. This model can be updated as more data is annotated and the annotation time is collected. If the annotation cost estimator is trained on the data annotated so far and the annotation cost estimate is used in active selection of the next annotation task, we face a similar exploration vs. exploitation tradeoff as discussed above for estimating the annotator reliability. A similar idea as above can be applied to the cost estimate for a strategy and annotator. However, we would use the lower confidence interval value in this case, since we would like to select the candidate with the lowest cost. If we use a simple annotation cost estimate as the mean annotation cost for a given strategy and annotator, then the lower confidence interval can be defined similar to Equation 6.1, as follows:

$$LI_C(s,a) = m(C(s,a)) - t_{\alpha/2}^{n-1}\frac{s(C(s,a))}{\sqrt{n}} \tag{6.2}$$

where $m(C(s,a))$ and $s(C(s,a))$ are the sample mean and standard deviation of the annotation cost for a given strategy and annotator. n is the number of samples observed for strategy s and annotator a. We note that in this simple estimate, the annotation cost does not vary across instances. We may group the instances into categories, for example, based on the instance length, to estimate the mean annotation cost for a category of instances. If we use a supervised regression model to estimate the annotation cost, like in Chapter 4, then we may use the confidence interval for the predicted values from the regression model[2]. In VOI computation, we would use the lower confidence interval value instead of the predicted annotation cost to select a candidate with low expected annotation cost or one with high uncertainty in its estimate. Alternatively, we may use an additional utility term in VOI to consider the uncertainty of the cost estimator, when selecting a candidate for annotation. Settles (2012) discusses several criteria that have been used for active selection for a regression model.

6.2.4 User Studies and Experiments

In this thesis, we proposed an alternate annotation strategy of providing rationales in addition to the instance's label. We also proposed a cost-sensitive active learning approach that selects the appropriate instance and strategy pair for a given annotator, user interface and annotation task with a given additional cost for annotating rationales. For future research in this area, we suggest the following user studies:

1. **Annotation Cost Variance:** For a given user interface, we may conduct user studies with several annotators and for several classification tasks to understand how the additional cost

[1]`https://www.mturk.com/mturk/welcome`
[2]`http://www.weibull.com/DOEWeb/confidence_intervals_in_simple_linear_regression.htm`

for rationales (as sub-instances in an instance) varies across annotators and annotation tasks. Through a pilot user study, we may first evaluate different user interface actions for annotating rationales to understand how the cost for annotating rationales can be reduced.

2. **Annotation Cost and Number of Rationales:** As discussed in Chapter 3, in addition to the two strategies of no rationales and almost all rationales, there can be other strategies in the middle that ask the annotator for fewer rationales. As mentioned before, there can be different ways to ask the annotators for N rationales per instance. We may ask them to provide *any* N rationales or *top* most important N rationales. As we observed for annotation strategies with *any* N rationales, after certain number of rationales there is no additional significant benefit from more rationales. However, we only compared the performance of different annotation strategies for the same annotation cost. The annotation cost would also vary across annotation strategies. While we can expect the annotation cost to increase with more rationales, when the annotator is asked to provide *any* N rationales (i.e. up to a maximum of N rationales), it is not clear how the annotation cost may vary when we ask the annotator to provide the *top* most important N rationales. The annotation cost in this case may not increase linearly with N and it may not be less than the cost for annotating all rationales. This is because providing *top* N rationales requires the annotator to rank the rationales (in their minds) in the order of importance, which may require more time than providing all rationales. A user study with multiple annotators is needed to understand how the annotation cost varies across different annotation strategies with a fixed number of rationales.

3. **User Experiments with IAL:** As mentioned before, for a real-time interactive annotation learning (IAL) system, there are practical concerns such as annotator wait-time while the system updates the model and selects the next instance. User experiments with an end-to-end system are needed to evaluate the amount of wait-time and its impact. To evaluate the savings in annotation time with IAL, user-in-the-loop experiments with multiple annotators should be conducted for both active and passive annotation task selection.

Note that unless we dictate an order in which the annotator provides instance's label and rationales, we cannot obtain the annotation time for multiple annotation strategies from the same annotator for a given instance. Thus, we may need several annotators with similar characteristics to provide us annotation time for different strategies on the same instance. Dictating an order for annotation, such as providing instance's label first and then providing rationales, may unnecessarily increase the annotation time for rationales as it may require the annotator to read an instance more than once.

6.2.5 Enhancements to Joint Active Selection of Instance and Strategy

In Chapter 3, we proposed a VOI approach to automatically determine when to ask for rationales based on the expected benefit and cost for annotating rationales. In Chapter 5, we presented an empirical analysis of how benefit from feature feedback varies across learning problems and what characteristics of a learning problem have a significant effect on benefit from feature feedback. A direction for future work would be to use the insights from the analysis in Chapter 5 to improve on the VOI approach. Knowledge about the characteristics of a learning problem and their association with the benefit from feature feedback can be used to improve our estimate for expected reduction in model's risk with rationales. For example, knowledge about the expected amount of irrelevant text in an instance can be used as a prior for the R vs. NR (rationale vs. not-rationale) classifier.

So far, in active learning, we have considered a scenario where we have a pre-defined budget for learning, and we must use all of it. However, if the performance has reached a plateau for an active selection approach, we should stop and not spend more annotation budget. On the other hand, if the performance is still improving when we have exhausted our annotation budget, we may want to ask for an increase in the annotation budget. The measure for learning rate used in Chapter 5 (based on the area under the learning curve) can be used to determine whether an approach has reached a plateau or if it is continuing to improve performance.

References

[Abedin et al.2011] Muhammad Arshad Abedin, Vincent Ng, and Latifur Rahman Khan. 2011. Learning cause identifiers from annotator rationales. In *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence - Volume Volume Three*, IJCAI'11, pages 1758–1763. AAAI Press.

[Airola et al.2008] Antti Airola, Sampo Pyysalo, Jari Bjrne, Tapio Pahikkala, Filip Ginter, and Tapio Salakoski. 2008. A graph kernel for protein-protein interaction extraction. In *Proceedings of BioNLP workshop at ACL*, Columbus, Ohio, USA.

[Arora and Nyberg2009] Shilpa Arora and Eric Nyberg. 2009. Interactive annotation learning with indirect feature voting. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Student Research Workshop and Doctoral Consortium*, SRWS '09, pages 55–60, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Arora and Nyberg2011] Shilpa Arora and Eric Nyberg. 2011. Assessing benefit from feature feedback in active learning for text classification. In *Proceedings of CONLL 2011*.

[Arora et al.2009a] Shilpa Arora, Mahesh Joshi, and Carolyn Penstein Rosé. 2009a. Identifying types of claims in online customer reviews. In *Proceedings of NAACL-HLT 2009*.

[Arora et al.2009b] Shilpa Arora, Eric Nyberg, and Carolyn Penstein Rosé. 2009b. Estimating annotation cost for active learning in a multi-annotator environment. In *Proceedings of the NAACL HLT 2009 Workshop on Active Learning for Natural Language Processing*, pages 18–26, Boulder, Colorado, June. Association for Computational Linguistics.

[Arora et al.2010] Shilpa Arora, Elijah Mayfield, Carolyn Penstein Rosé, and Eric Nyberg. 2010. Sentiment classification using automatically extracted subgraph features. In *Proceedings of the Workshop on Emotion in Text at NAACL*.

[Arora et al.2012] Shilpa Arora, Pinar Donmez, and Eric Nyberg. 2012. Cost-sensitive multi-strategy active annotation for text classification. In *11th International Conference on Machine Learning and Applications (ICMLA) (Under Review)*.

[Asai et al.2002] Tatsuya Asai, Kenji Abe, Shinji Kawasoe, Hiroshi Sakamoto, and Setsuo Arikawa. 2002. Efficient substructure discovery from large semi-structured data. In *Proceedings of SIAM Int. Conf. on Data Mining (SDM)*, pages 158–174.

[Attenberg et al.2010] Josh Attenberg, Prem Melville, and Foster Provost. 2010. A unified approach to active dual supervision for labeling features and examples. In *Proceedings of the 2010 European conference on Machine learning and knowledge discovery in databases: Part I*, ECML PKDD'10, pages 40–55, Berlin, Heidelberg. Springer-Verlag.

[Bach and Badaskar2007] Nguyen Bach and Sameer Badaskar. 2007. A survey on relation extraction.

[Baldridge and Palmer2009] Jason Baldridge and Alexis Palmer. 2009. How well does active learning actually work?: Time-based evaluation of cost-reduction strategies for language documentation. In *EMNLP '09: Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 296–305, Morristown, NJ, USA. Association for Computational Linguistics.

[Bennett and Carbonell2005] Paul N. Bennett and Jaime Carbonell. 2005. Detecting action-items in e-mail. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '05, pages 585–586, New York, NY, USA. ACM.

[Bilotti et al.2010] Matthew W. Bilotti, Jonathan L. Elsas, Jaime Carbonell, and Eric Nyberg. 2010. Rank learning for factoid question answering with linguistic and semantic constraints. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management (CIKM 2010)*.

[Bunescu and Mooney2005a] Razvan Bunescu and Raymond Mooney. 2005a. Subsequence kernels for relation extraction. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 171–178. MIT Press, Cambridge, MA.

[Bunescu and Mooney2005b] Razvan C. Bunescu and Raymond J. Mooney. 2005b. A shortest path dependency kernel for relation extraction. In *HLT '05: Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 724–731, Morristown, NJ, USA. Association for Computational Linguistics.

[Buscher et al.2012] Georg Buscher, Andreas Dengel, Ralf Biedert, and Ludger V. Elst. 2012. Attentive documents: Eye tracking as implicit feedback for information retrieval and beyond. *ACM Trans. Interact. Intell. Syst.*, 1(2):9:1–9:30, January.

[Cauwenberghs and Poggio2000] Gert Cauwenberghs and Tomaso Poggio. 2000. Incremental and decremental support vector machine learning. In *Proc. of NIPS conference*, volume 13, pages 409–415. MIT Press.

[Chang and Lin2001] Chih-Chung Chang and Chih-Jen Lin, 2001. *LIBSVM: a library for support vector machines*. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[Charniak and Johnson2005] Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 173–180, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Cohen2004] William W. Cohen. 2004. Minorthird: Methods for identifying names and ontological relations in text using heuristics for inducing regularities from data. http://minorthird.sourceforge.net.

[Collins and Duffy2001] Michael Collins and Nigel Duffy. 2001. Convolution kernels for natural language. In *Advances in Neural Information Processing Systems 14*, pages 625–632. MIT Press.

[Collins and Duffy2002] Michael Collins and Nigel Duffy. 2002. New ranking algorithms for parsing and tagging: kernels over discrete structures, and the voted perceptron. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 263–270, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Cordella et al.1999] Luigi P. Cordella, Pasquale Foggia, Carlo Sansone, and Mario Vento. 1999. Performance evaluation of the vf graph matching algorithm. In *ICIAP '99: Proceedings of the 10th International Conference on Image Analysis and Processing*, page 1172, Washington, DC, USA. IEEE Computer Society.

[Cordella et al.2001] Luigi P. Cordella, Pasquale Foggia, Carlo Sansone, and Mario Vento. 2001. An improved algorithm for matching large graphs. In *In: 3rd IAPR-TC15 Workshop on Graph-based Representations in Pattern Recognition, Cuen*, pages 149–159.

[Culotta and Sorensen2004] Aron Culotta and Jeffrey Sorensen. 2004. Dependency tree kernels for relation extraction. In *ACL '04: Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 423, Morristown, NJ, USA. Association for Computational Linguistics.

[Deshpande et al.2005] Mukund Deshpande, Michihiro Kuramochi, Nikil Wale, and George Karypis. 2005. Frequent substructure-based approaches for classifying chemical compounds. *IEEE Trans. on Knowl. and Data Eng.*, 17(8):1036–1050.

[Donahue and Grauman2011] Jeff Donahue and Kristen Grauman. 2011. Annotator rationales for visual recognition. In *ICCV*, pages 1395–1402. IEEE.

[Donmez and Carbonell2008a] Pinar Donmez and Jaime G. Carbonell. 2008a. Paired Sampling in Density-Sensitive Active Learning. In *Proceedings of the International Symposium on Artificial Intelligence and Mathematics*.

[Donmez and Carbonell2008b] Pinar Donmez and Jaime G. Carbonell. 2008b. Proactive learning: cost-sensitive active learning with multiple imperfect oracles. In *CIKM '08: Proceedings of the 17th ACM conference on Information and knowledge management*, pages 619–628, New York, NY, USA. ACM.

[Donmez et al.2009] Pinar Donmez, Jaime G. Carbonell, and Jeff Schneider. 2009. Efficiently learning the accuracy of labeling sources for selective sampling. In *KDD '09: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 259–268, New York, NY, USA. ACM.

[Donmez et al.2010] Pinar Donmez, Jaime G. Carbonell, and Jeff Schneider. 2010. A probabilistic framework to learn from multiple annotators with time-varying accuracy. In *SDM*, pages 826–837.

[Druck et al.2008] Gregory Druck, Gideon Mann, and Andrew McCallum. 2008. Learning from labeled features using generalized expectation criteria. In *SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 595–602, New York, NY, USA. ACM.

[Druck et al.2009] Gregory Druck, Burr Settles, and Andrew McCallum. 2009. Active learning by labeling features. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1 - Volume 1*, EMNLP '09, pages 81–90, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Efron and Tibshirani1994] B. Efron and R.J. Tibshirani. 1994. *An introduction to the bootstrap.* Monographs on Statistics and Applied Probability. Chapman and Hall/CRC, New York.

[Efron et al.2004] Bradley Efron, Trevor Hastie, Iain Johnstone, and Robert Tibshirani. 2004. Least angle regression. *Annals of Statistics*, 32:407–499.

[Engelson and Dagan1996] Sean P. Engelson and Ido Dagan. 1996. Minimizing manual annotation cost in supervised training from corpora. In *In Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pages 319–326. Springer.

[Erkan et al.2007] Gunes Erkan, Arzucan Ozgur, and Dragomir R. Radev. 2007. Semi-supervised classification for extracting protein interaction sentences using dependency parsing. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 228–237, Prague, Czech Republic, June. Association for Computational Linguistics.

[Estrada et al.2004] Francisco Estrada, Allan Jepson, and Chakra Chennubhotla. 2004. Spectral embedding and min-cut for image segmentation. In *In British Machine Vision Conference.*

[Fayruzov et al.2009] Timur Fayruzov, Martine De Cock, Chris Cornelis, and Veronique Hoste. 2009. Linguistic feature analysis for protein interaction extraction. *BMC Bioinformatics.*

[Flesch1948] Rudolf Flesch. 1948. A new readability yardstick. In *Proceedings of Journal of Applied Psychology*, volume 2, pages 221–233.

[Gamon2004] Michael Gamon. 2004. Sentiment classification on customer feedback data: noisy data, large feature vectors, and the role of linguistic analysis. In *COLING '04: Proceedings of the 20th international conference on Computational Linguistics*, page 841, Morristown, NJ, USA. Association for Computational Linguistics.

[Gianfortoni et al.2011] Philip Gianfortoni, David Adamson, and Carolyn P. Rosé. 2011. Modeling of stylistic variation in social media with stretchy patterns. In *Proceedings of the First Workshop on Algorithms and Resources for Modelling of Dialects and Language Varieties*, DIALECTS '11, pages 49–59, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Gildea and Jurafsky2000] Daniel Gildea and Daniel Jurafsky. 2000. Automatic labeling of semantic roles. In *ACL '00: Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pages 512–520, Morristown, NJ, USA. Association for Computational Linguistics.

[Godbole et al.2004] Shantanu Godbole, Abhay Harpale, Sunita Sarawagi, and Soumen Chakrabarti. 2004. Document classification through interactive supervision of document and term labels. In *PKDD '04: Proceedings of the 8th European Conference on Principles and Practice of Knowledge Discovery in Databases*, pages 185–196, New York, NY, USA. Springer-Verlag New York, Inc.

[Grundmann et al.2010] Matthias Grundmann, Vivek Kwatra, Mei Han, and Irfan Essa. 2010. Efficient hierarchical graph based video segmentation. *IEEE CVPR.*

[GuoDong et al.2005] Zhou GuoDong, Su Jian, Zhang Jie, and Zhang Min. 2005. Exploring various knowledge in relation extraction. In *ACL '05: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 427–434, Morristown, NJ, USA. Association for Computational Linguistics.

[Haertel et al.2008] Robbie A. Haertel, Kevin D. Seppi, Eric K. Ringger, and Janes L. Cattoll. 2008. Return on investment for active learning. In *Proceedings of the NIPS Workshop on Cost-Sensitive Learning.*

[Haertel et al.2010] Robbie Haertel, Paul Felt, Eric Ringger, and Kevin Seppi. 2010. Parallel active learning: eliminating wait time with minimal staleness. In *Proceedings of the NAACL HLT 2010 Workshop on Active Learning for Natural Language Processing*, ALNLP '10, pages 33–41, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Haussler1999] David Haussler. 1999. Convolution kernels on discrete structures. Technical report, University of California, Santa Cruz.

[Hwa2000] Rebecca Hwa. 2000. Sample selection for statistical grammar induction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 45–52.

[Inokuchi et al.2000] Akihiro Inokuchi, Takashi Washio, and Hiroshi Motoda. 2000. An apriori-based algorithm for mining frequent substructures from graph data. In *Proceedings of KDD.*

[Joshi and Rosé2009] Mahesh Joshi and Carolyn Penstein Rosé. 2009. Generalizing dependency features for opinion mining. In *ACL-IJCNLP '09: Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 313–316, Morristown, NJ, USA. Association for Computational Linguistics.

[Joshi et al.2010] Mahesh Joshi, Dipanjan Das, Kevin Gimpel, and Noah A. Smith. 2010. Movie reviews and revenues: An experiment in text regression. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 293–296, Los Angeles, California, June. Association for Computational Linguistics.

[Kambhatla2004] Nanda Kambhatla. 2004. Combining lexical, syntactic, and semantic features with maximum entropy models for extracting relations. In *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions*, page 22, Morristown, NJ, USA. Association for Computational Linguistics.

[Kapoor et al.2007] Ashish Kapoor, Eric Horvitz, and Sumit Basu. 2007. Selective supervision: guiding supervised learning with decision-theoretic active learning. In *Proceedings of the 20th international joint conference on Artifical intelligence*, pages 877–882, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

[Kashima and Koyanagi2002] Hisashi Kashima and Teruo Koyanagi. 2002. Svm kernels for semi-structured data. In *Proc. ICML, 2002*, pages 291–298.

[Kashima et al.2003] Hisashi Kashima, Koji Tsuda, and Akihiro Inokuchi. 2003. Marginalized kernels between labeled graphs. In *Proceedings of the Twentieth International Conference on Machine Learning*, pages 321–328. AAAI Press.

[King et al.2004] Ross D. King, Kenneth E. Whelan, Ffion M. Jones, Philip G. K. Reiser, Christopher H. Bryant, Stephen H. Muggleton, Douglas B. Kell, and Stephen G. Oliver. 2004. Functional genomic hypothesis generation and experimentation by a robot scientist. *Nature*, 427(6971):247–252, January.

[Klein and Manning2003] Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *ACL '03: Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, pages 423–430, Morristown, NJ, USA. Association for Computational Linguistics.

[Koza1992] John Koza. 1992. Genetic programming: On the programming of computers by means of natural selection. In *MIT Press*.

[Kristjansson et al.2004] Trausti Kristjansson, Aron Culotta, and Paul Viola. 2004. Interactive information extraction with constrained conditional random fields. In *Proceedings of AAAI*, pages 412–418.

[Kudo et al.2004] Taku Kudo, Eisaku Maeda, and Yuji Matsumoto. 2004. An application of boosting to graph classification. In *Proceedings of NIPS*.

[Kuramochi and Karypis2002] Michihiro Kuramochi and George Karypis. 2002. Frequent subgraph discovery. In *Proceedings of ICDM*.

[Lang1995] K. Lang. 1995. NewsWeeder: Learning to filter netnews. In *12th International Conference on Machine Learning (ICML95)*, pages 331–339.

[Lewis1995] D. Lewis. 1995. The reuters-21578 text categorization test collection.

[Lodhi et al.2000] Huma Lodhi, John Shawe-Taylor, Nello Cristianini, and Christopher J. C. H. Watkins. 2000. Text classification using string kernels. In *Proceedings of NIPS*, pages 563–569.

[Low et al.2010] Yucheng Low, Joseph Gonzalez, Aapo Kyrola, Danny Bickson, Carlos Guestrin, and Joseph M. Hellerstein. 2010. Graphlab: A new parallel framework for machine learning. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, Catalina Island, California, July.

[Manning et al.2008] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press, July.

[Matsumoto et al.2005] Shotaro Matsumoto, Hiroya Takamura, and Manabu Okumura. 2005. Sentiment classification using word sub-sequences and dependency sub-trees. In *Proceedings of PAKDD'05, the 9th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*.

[Mayfield and Rosé2010] Elijah Mayfield and Carolyn Penstein Rosé. 2010. Using feature construction to avoid large feature spaces in text classification. In *Proceedings of the Genetic and Evolutionary Computation Conference*.

[Melville and Sindhwani2009] Prem Melville and Vikas Sindhwani. 2009. Active dual supervision: Reducing the cost of annotating examples and features. In *Proceedings of the NAACL HLT 2009 Workshop on Active Learning for Natural Language Processing*, pages 49–57, Boulder, Colorado, June. Association for Computational Linguistics.

[Melville et al.2005] P. Melville, M. Saar-Tsechansky, F. Provost, and Raymond J. Mooney. 2005. An expected utility approach to active feature-value acquisition. In *Proceedings of the International Conference on Data Mining*, pages 745–748, Houston, TX, November.

[Moschitti2006] Alessandro Moschitti. 2006. Efficient convolution kernels for dependency and constituent syntactic trees. In *Proceedings of the 17th European conference on Machine Learning*, ECML'06, pages 318–329, Berlin, Heidelberg. Springer-Verlag.

[Nguyen et al.2009] Truc-Vien T. Nguyen, Alessandro Moschitti, and Giuseppe Riccardi. 2009. Convolution kernels on constituent, dependency and sequential structures for relation extraction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3 - Volume 3*, EMNLP '09, pages 1378–1387, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Pang and Lee2004] Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the ACL*, pages 271–278.

[Pang and Lee2005] Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of ACL*.

[Pang et al.2002] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: sentiment classification using machine learning techniques. In *EMNLP '02: Proceedings of the ACL-02 conference on Empirical methods in natural language processing*, pages 79–86, Morristown, NJ, USA. Association for Computational Linguistics.

[Pei et al.2004] Jian Pei, Jiawei Han, Behzad Mortazavi-Asl, Jianyong Wang, Helen Pinto, Qiming Chen, Umeshwar Dayal, and Mei-Chun Hsu. 2004. Mining sequential patterns by pattern-growth: The prefixspan approach. *IEEE Transactions on Knowledge And Data Engineering*, 16(11):2004.

[Platt1999] John C. Platt. 1999. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *ADVANCES IN LARGE MARGIN CLASSIFIERS*, pages 61–74. MIT Press.

[Pradhan et al.2004] Sameer Pradhan, Wayne Ward, Kadri Hacioglu, James H. Martin, and Dan Jurafsky. 2004. Shallow semantic parsing using support vector machines. In *Proceedings of the Human Language Technology Conference/North American chapter of the Association of Computational Linguistics (HLT/NAACL)*.

[Pradhan et al.2008] Sameer S. Pradhan, Wayne Ward, and James H. Martin. 2008. Towards robust semantic role labeling. *Comput. Linguist.*, 34(2):289–310.

[Raghavan and Allan2007] Hema Raghavan and James Allan. 2007. An interactive algorithm for asking and incorporating feature feedback into support vector machines. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 79–86, New York, NY, USA. ACM.

[Raghavan et al.2006] Hema Raghavan, Omid Madani, and Rosie Jones. 2006. Active learning with feedback on features and instances. *Journal of Machine Learning Research*, 7:1655–1686.

[Raghavan et al.2007] Hema Raghavan, Omid Madani, and Rosie Jones. 2007. When will feature feedback help? quantifying the complexity of classification problems. In *IJCAI Workshop on Human in the Loop Computing*.

[Ren and Zhao2010] Lijie Ren and Lei Zhao. 2010. Sentiment classification based on uncertain frequent graph pattern mining. Technical report, University of California Santa Barbara.

[Reynar and Ratnaparkhi1997] Jeffrey C. Reynar and Adwait Ratnaparkhi. 1997. A maximum entropy approach to identifying sentence boundaries. In *Proceedings of the fifth conference on Applied natural language processing*, ANLC '97, pages 16–19, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Rijsbergen1979] C. J. Van Rijsbergen. 1979. *Information Retrieval*. Butterworths, London, 2 edition.

[Riloff et al.2006] Ellen Riloff, Siddharth Patwardhan, and Janyce Wiebe. 2006. Feature subsumption for opinion analysis. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 440–448. Association for Computational Linguistics.

[Ringger et al.2008] Eric K. Ringger, Marc Carmen, Robbie Haertel, Kevin D. Seppi, Deryle Lonsdale, Peter McClanahan, James L. Carroll, and Noel Ellison. 2008. Assessing the costs of machine-assisted corpus annotation through a user study. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco, may. European Language Resources Association (ELRA). http://www.lrec-conf.org/proceedings/lrec2008/.

[Rink et al.2010] Bryan Rink, Cosmin Adrian Bejan, and Sanda M. Harabagiu. 2010. Learning textual graph patterns to detect causal event relations. In *FLAIRS Conference*.

[Roy and Mccallum2001] Nicholas Roy and Andrew Mccallum. 2001. Toward optimal active learning through sampling estimation of error reduction. In *In Proc. 18th International Conf. on Machine Learning*, pages 441–448. Morgan Kaufmann.

[SasInstitute and SASPublishing2008] SasInstitute and SASPublishing. 2008. *Jmp 8 Statistics and Graphics Guide*. Sas Inst.

[Schapire and Singer2000] Robert E. Schapire and Yoram Singer. 2000. Boostexter: A boosting-based system for text categorization.

[Settles et al.2008a] Burr Settles, Mark Craven, and Lewis Friedland. 2008a. Active learning with real annotation costs. In *Proceedings of the NIPS Workshop on Cost-Sensitive Learning*, pages 1–10.

[Settles et al.2008b] Burr Settles, Mark Craven, and Soumya Ray. 2008b. Multiple-instance active learning. In *Advances in Neural Information Processing Systems (NIPS)*, volume 20, pages 1289–1296. MIT Press.

[Settles2011] Burr Settles. 2011. Closing the loop: Fast, interactive semi-supervised annotation with queries on features and instances. In *EMNLP*, pages 1467–1478. ACL.

[Settles2012] Burr Settles. 2012. *Active learning*. Morgan & Claypool.

[Shervashidze and Borgwardt2009] Nino Shervashidze and Karsten Borgwardt. 2009. Fast subtree kernels on graphs. In *Advances in Neural Information Processing Systems 22*, pages 1660–1668.

[Shervashidze et al.2009] Nino Shervashidze, SVN Vishwanathan, Tobias Petri, Kurt Mehlhorn, and Karsten Borgwardt. 2009. Efficient Graphlet Kernels for Large Graph Comparison. In *12th International Conference on Artificial Intelligence and Statistics (AISTATS)*, Clearwater Beach, Fl, USA, April, 16-18, 2009. Society for Artificial Intelligence and Statistics.

[Shi and Malik2000] Jianbo Shi and Jitendra Malik. 2000. Normalized cuts and image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(8):888–905, August.

[Smola and Scholkopf1998] Alex J. Smola and Bernhard Scholkopf. 1998. A tutorial on support vector regression. *Statistics and Computing*, 14(3):199+.

[Smola and Schölkopf2004] Alex J. Smola and Bernhard Schölkopf. 2004. A tutorial on support vector regression. *Statistics and Computing*, 14(3):199–222.

[Stoyanov et al.2009] Veselin Stoyanov, Nathan Gilbert, Claire Cardie, Ellen Riloff, David Buttler, and David Hysom. 2009. Reconcile: A coreference resolution research platform. *Computer Science Technical Report*.

[Thoma et al.2009] Marisa Thoma, Hong Cheng, Arthur Gretton, Jiawei Han, Hans peter Kriegel, Alex Smola, Le Song, Philip S. Yu, Xifeng Yan, and Karsten Borgwardt. 2009. Near-optimal supervised feature selection among frequent subgraphs. In *Proceedings of SIAM Intl Conf. on Data Mining*.

[Tomanek and Hahn2010] Katrin Tomanek and Udo Hahn. 2010. A comparison of models for cost-sensitive active learning. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, COLING '10, pages 1247–1255, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Tomanek et al.2007] Katrin Tomanek, Joachim Wermter, and Udo Hahn. 2007. An approach to text corpus construction which cuts annotation costs and maintains reusability of annotated data. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 486–495, Prague, Czech Republic, June. Association for Computational Linguistics.

[Tong and Koller2000] Simon Tong and Daphne Koller. 2000. Support vector machine active learning with applications to text classification. In *JOURNAL OF MACHINE LEARNING RESEARCH*, pages 999–1006.

[Tsuda2007] Koji Tsuda. 2007. Entire regularization paths for graph data. In *ICML '07: Proceedings of the 24th international conference on Machine learning*, pages 919–926, New York, NY, USA. ACM.

[Tukey1953] John Tukey. 1953. The problem of multiple comparisons. Unpublished manuscript, Princeton University.

[Turney2000] Peter Turney. 2000. Types of cost in inductive concept learning. In *In Workshop on Cost-Sensitive Learning at the Seventeenth International Conference on Machine Learning*, pages 15–21.

[Ullmann1976] Julian R. Ullmann. 1976. An algorithm for subgraph isomorphism. *J. ACM*, 23(1):31–42.

[Vijayanarasimhan and Grauman2011] Sudheendra Vijayanarasimhan and Kristen Grauman. 2011. Cost-sensitive active visual category learning. *International Journal of Computer Vision*, 91(1):24–44.

[Vijayanarasimhan et al.2010] Sudheendra Vijayanarasimhan, Prateek Jain, and Kristen Grauman. 2010. Far-sighted active learning on a budget for image and video recognition. In *CVPR*, pages 3035–3042. IEEE.

[Vishwanathan and Smola2002] S.V.N. Vishwanathan and Alexander J. Smola. 2002. Fast kernels on strings and trees. In *Advances in Neural Information Processing Systems*.

[Wallace et al.2010] Byron C. Wallace, Kevin Small, Carla E. Brodley, Joseph Lau, and Thomas A. Trikalinos. 2010. Modeling annotation time to reduce workload in comparative effectiveness reviews. In *Proceedings of the 1st ACM International Health Informatics Symposium*, IHI '10, pages 28–35, New York, NY, USA. ACM.

[Whitehill et al.2009] Jacob Whitehill, Paul Ruvolo, Ting fan Wu, Jacob Bergsma, and Javier Movellan. 2009. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In *Advances in Neural Information Processing Systems 22*, pages 2035–2043.

[Wilson et al.2004] Theresa Wilson, Janyce Wiebe, and Rebecca Hwa. 2004. Just how mad are you? finding strong and weak opinion clauses. In *Proceedings of AAAI*, pages 761–769.

[Wilson et al.2005] Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of Human Language Technologies Conference/Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP 2005)*, Vancouver, CA.

[Witten and Frank2005] Ian H. Witten and Eibe Frank. 2005. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, 2nd edition.

[Yan and Han2002a] Xifeng Yan and Jiawei Han. 2002a. gspan: Graph-based substructure pattern mining. In *ICDM '02: Proceedings of the 2002 IEEE International Conference on Data Mining*, page 721, Washington, DC, USA. IEEE Computer Society.

[Yan and Han2002b] Xifeng Yan and Jiawei Han. 2002b. gspan: Graph-based substructure pattern mining. Technical Report UIUCDCS-R-2002-2296, Department of Computer Science, University of Illinois at Urbana-Champaign.

[Yan et al.2012] Yan Yan, Rmer Rosales, Glenn Fung, Faisal Farooq, Bharat Rao, and Jennifer G. Dy. 2012. Active learning from multiple knowledge sources. *Journal of Machine Learning Research - Proceedings Track*, pages 1350–1357.

[Yang and Pedersen1997] Yiming Yang and Jan O. Pedersen. 1997. A comparative study on feature selection in text categorization. In *Proceedings of the Fourteenth International Conference on Machine Learning*, ICML '97, pages 412–420, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

[Yessenalina et al.2010] Ainur Yessenalina, Yejin Choi, and Claire Cardie. 2010. Automatically generating annotator rationales to improve sentiment classification. In *Proceedings of the ACL 2010 Conference Short Papers*, ACLShort '10, pages 336–341, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Zaidan and Eisner2008] Omar F. Zaidan and Jason Eisner. 2008. Modeling annotators: A generative approach to learning from annotator rationales. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 31–40, Honolulu, October.

[Zaidan et al.2007] Omar Zaidan, Jason Eisner, and Christine Piatko. 2007. Using "annotator rationales" to improve machine learning for text categorization. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 260–267, Rochester, New York, April. Association for Computational Linguistics.

[Zelenko et al.2003] Dmitry Zelenko, Chinatsu Aone, Anthony Richardella, Jaz K, Thomas Hofmann, Tomaso Poggio, and John Shawe-taylor. 2003. Kernel methods for relation extraction. *Journal of Machine Learning Research*, 3:2003.

[Zou and Hastie2005] Hui Zou and Trevor Hastie. 2005. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society, Series B*, 67:301–320.