Interactive Feature Space Construction using Semantic Information

Dan Roth and **Kevin Small**
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL 61801
{danr,ksmall}@illinois.edu

Abstract

Specifying an appropriate feature space is an important aspect of achieving good performance when designing systems based upon learned classifiers. Effectively incorporating information regarding semantically related words into the feature space is known to produce robust, accurate classifiers and is one apparent motivation for efforts to automatically generate such resources. However, naive incorporation of this semantic information may result in poor performance due to increased ambiguity. To overcome this limitation, we introduce the *interactive feature space construction* protocol, where the learner identifies inadequate regions of the feature space and in coordination with a domain expert adds descriptiveness through existing semantic resources. We demonstrate effectiveness on an entity and relation extraction system including both performance improvements and robustness to reductions in annotated data.

1 Introduction

An important natural language processing (NLP) task is the design of learning systems which perform well over a wide range of domains with limited training data. While the NLP community has a long tradition of incorporating linguistic information into statistical systems, machine learning approaches to these problems often emphasize learning sophisticated models over simple, mostly lexical, features. This trend is not surprising as a primary motivation for machine learning solutions is to reduce the manual effort required to achieve state of the art performance. However, one notable advantage of discriminative classifiers is the capacity to encode arbitrarily complex features, which partially accounts for their popularity. While this flexibility is powerful, it often overwhelms the system designer causing them to resort to simple features. This work presents a method to partially automate feature engineering through an interactive learning protocol.

While it is widely accepted that classifier performance is predicated on feature engineering, designing good features requires significant effort. One underutilized resource for descriptive features are existing *semantically related word lists* (SRWLs), generated both manually (Fellbaum, 1998) and automatically (Pantel and Lin, 2002). Consider the following named entity recognition (NER) example:

> *His father was rushed to [Westlake Hospital]*$_{\mathbf{ORG}}$*, an arm of [Resurrection Health Care]*$_{\mathbf{ORG}}$*, in west suburban [Chicagoland]*$_{\mathbf{LOC}}$.

For such tasks, it is helpful to know that *west* is a member of the SRWL *[Compass Direction]* and other such designations. If extracting features using this information, we would require observing only a subset of the SRWL in the data to learn the corresponding parameter. This statement suggests that one method for learning robust classifiers is to incorporate semantic information through features extracted from the more descriptive representation:

> *His father was rushed to Westlake [Health Care Institution], an [Subsidiary] of Resurrection Health Care, [Locative Preposition] [Compass Direction] suburban Chicagoland.*

Deriving discriminative features from this representation often results in more informative features and a correspondingly simpler classification task. Although effective approaches along this vein have been shown to induce more accurate classifiers (Boggess et al., 1991; Miller et al., 2004; Li and Roth, 2005), naive approaches may instead result in higher sample complexity due to increased ambiguity introduced through these semantic resources. Features based upon SRWLs must therefore balance the tradeoff between descriptiveness and noise.

This paper introduces the *interactive feature space construction* (IFSC) protocol, which facilitates coordination between a domain expert and learning algorithm to interactively define the feature space during training. This paper describes the particular instance of the IFSC protocol where semantic information is introduced through abstraction of lexical terms in the feature space with their SRWL labels. Specifically, there are two notable contributions of this work: (1) an interactive method for the expert to directly encode semantic knowledge into the feature space with minimal effort and (2) a querying function which uses both the current state of the learner and properties of the available SRWLs to select informative instances for presentation to the expert. We demonstrate the effectiveness of this protocol on an entity and relation extraction task in terms of performance and labeled data requirements.

2 Preliminaries

Following standard notation, let $x \in \mathcal{X}$ represent members of an input domain and $y \in \mathcal{Y}$ represent members of an output domain where a learning algorithm uses a training sample $\mathcal{S} = \{(x_i, y_i)\}_{i=1}^{m}$ to induce a prediction function $h : \mathcal{X} \rightarrow \mathcal{Y}$. We are specifically interested in discriminative classifiers which use a feature vector generating procedure $\Phi(x) \rightarrow \mathbf{x}$, taking an input domain member x and generating a feature vector $\mathbf{x}$. We further assume the output assignment of h is based upon a scoring function $f : \Phi(\mathcal{X}) \times \mathcal{Y} \rightarrow \mathbb{R}$ such that the prediction is stated as $\hat{y} = h(x) = \text{argmax}_{y' \in \mathcal{Y}} f(\mathbf{x}, y')$.

The feature vector generating procedure is composed of a vector of feature generation functions (FGFs), $\Phi(x) = \langle \Phi_1(x), \Phi_2(x), \ldots, \Phi_n(x) \rangle$, where each feature generation function, $\Phi_i(x) \rightarrow \{0, 1\}$, takes the input x and returns the appropriate feature vector value. Consider the text "*in west suburban Chicagoland*" where we wish to predict the entity classification for *Chicagoland*. In this case, example active FGFs include $\Phi_{text=Chicagoland}$, $\Phi_{isCapitalized}$, and $\Phi_{text(-2)=west}$ while FGFs such as $\Phi_{text=and}$ would remain inactive. Since we are constructing sparse feature vectors, we use the *infinite attribute model* (Blum, 1992).

Semantically related word list (SRWL) feature abstraction begins with a set of variable sized word lists $\{\mathcal{W}\}$ such that each member lexical element (i.e. word, phrase) has at least one sense that is semantically related to the concept represented by $\mathcal{W}$ (e.g. $\mathcal{W}_{compass\_direction} = north, east, \ldots, southwest$). For the purpose of feature extraction, whenever the sense of a lexical element associated with a particular $\mathcal{W}$ appears in the corpus, it is replaced by the name of the corresponding SRWL. This is equivalent to defining a FGF for the specified $\mathcal{W}$ which is a disjunction of the functionally related FGFs over the member lexical elements (e.g. $\Phi_{text \in \mathcal{W}_{compass\_direction}} = \Phi_{text=north} \vee \Phi_{text=east} \vee \ldots \vee \Phi_{text=southwest}$).

3 Interactive Feature Space Construction

The machine learning community has become increasingly interested in protocols which allow interaction with a domain expert during training, such as the active learning protocol (Cohn et al., 1994). In active learning, the learning algorithm reduces the labeling effort by using a querying function to incrementally select unlabeled examples from a data source for annotation during learning. By carefully selecting examples for annotation, active learning maximizes the quality of inductive information while minimizing label acquisition cost.

While active learning has been shown to reduce sample complexity, we contend that it significantly underutilizes the domain expert – particularly for complex annotation tasks. More precisely, when a domain expert receives an instance, world knowledge is used to reason about the instance and supply an annotation. Once annotated and provided for training, the learner must recover this world knowledge and incorporate it into its model from a small number of instances, exclusively through induction.

Learning algorithms generally assume that the feature space and model are specified before learning begins and remain static throughout learning, where training data is exclusively used for parameter estimation. Conversely, the *interactive feature space construction* (IFSC) protocol relaxes this static feature space assumption by using information about the current state of the learner, properties of knowledge resources (e.g. SRWLs, gazetteers, unlabeled data, etc.), and access to the domain expert during training to interactively improve the feature space. Whereas active learning focuses on the labeling effort, IFSC reduces sample complexity and improves performance by modifying the underlying representation to simplify the overall learning task.

The IFSC protocol for SRWL abstraction is presented in Algorithm 1. Given a labeled data set $\mathcal{S}$, an initial feature vector generating procedure Φ_0, a querying function $\mathcal{Q} : \mathcal{S} \times h \rightarrow \mathcal{S}_{select}$, and an existing set of semantically related word lists, $\{\mathcal{W}\}$ (line 1), an initial hypothesis is learned (line 3). The querying function scores the labeled examples and selects an instance for interaction (line 6). The expert selects lexical elements from this instance for which feature abstractions may be performed (line 8). If the expert doesn't deem any elements viable for interaction, the algorithm returns to line 5. Once lexical elements are selected for interaction, the SRWL $\mathcal{W}_\epsilon$ associated with each selected element is retrieved (line 11) and refined by the expert (line 12). Using the validated SRWL definition $\mathcal{W}^*_\epsilon$, the lexical FGFs are replaced with the SRWL FGF (line 14). This new feature vector generating procedure Φ_{t+1} is used to train a new classifier (line 18) and the algorithm is repeated until the annotator halts.

Algorithm 1 Interactive Feature Space Construction

1: **Input:** Labeled training data $\mathcal{S}$, feature vector generating procedure Φ_0, querying function $\mathcal{Q}$, set of known SRWLs $\{W\}$, domain expert $\mathcal{A}^*$

2: $t \leftarrow 0$
3: $h_t \leftarrow \mathcal{A}(\Phi_t, \mathcal{S})$; learn initial hypothesis
4: $\mathcal{S}_{selected} \leftarrow \emptyset$
5: **while** annotator is willing **do**
6: $\quad \mathcal{S}_{select} \leftarrow \mathcal{Q}(\mathcal{S} \backslash \mathcal{S}_{selected}, h_t)$; $\mathcal{Q}$ proposes (labeled) instance for interaction
7: $\quad \mathcal{S}_{selected} \leftarrow \mathcal{S}_{selected} \cup \mathcal{S}_{select}$; mark selected examples to prevent reselection
8: $\quad E_{select} \leftarrow \mathcal{A}^*(\mathcal{S}_{select})$; the expert selects lexical elements for semantic abstraction
9: $\quad \Phi_{t+1} \leftarrow \Phi_t$; initialize new FGF vector with existing FGFs
10: $\quad$ **for each** $\epsilon \in E_{select}$ **do**
11: $\quad\quad$ Retrieve word list $\mathcal{W}_\epsilon$
12: $\quad\quad \mathcal{W}^*_\epsilon \leftarrow \mathcal{A}^*(\mathcal{W}_\epsilon)$; the expert refines the existing semantic class $\mathcal{W}_\epsilon$ for this task
13: $\quad\quad$ **for each** $\Phi \sim \epsilon$ **do**
14: $\quad\quad\quad \Phi_{t+1} \leftarrow (\Phi_{t+1} \backslash \Phi) \cup \Phi_{\mathcal{W}^*_\epsilon}$; replace features with SRWL features (e.g. $\Phi_{text=\epsilon} \rightarrow \Phi_{text \in \mathcal{W}^*_\epsilon}$)
15: $\quad\quad$ **end for**
16: $\quad$ **end for**
17: $\quad t \leftarrow t + 1$
18: $\quad h_t \leftarrow \mathcal{A}(\Phi_t, \mathcal{S})$; learn new hypothesis
19: **end while**

20: **Output:** Learned hypothesis h_T, final feature space Φ_T, refined semantic classes $\{\mathcal{W}^*\}$

3.1 Method of Expert Interaction

The method of interaction for active learning is very natural; data annotation is required regardless. To increase the bandwidth between the expert and learner, a more sophisticated interaction must be allowed while ensuring that the expert task of remains reasonable. We require the interaction be restricted to mouse clicks. When using this protocol to incorporate semantic information, the primary tasks of the expert are (1) selecting lexical elements for SRWL feature abstraction and (2) validating membership of the SRWL for the specified application.

3.1.1 Lexical Feature Selection (Line 8)

Once an instance is selected by the querying function (line 6), the the domain expert selects lexical elements (i.e. words, phrases) believed appropriate for SRWL feature abstraction. This step is summarized by Figure 1 for the example introduced in Section 1.

For this NER example, features extracted include the words and bigrams which form the named entity and those within a surrounding two word window. All lexical elements which have membership to at least one SRWL and are used for feature extraction are marked with a box and may be selected by the user for interaction. In this particular case, the system has made a mistake in classification of

His father was rushed *to [Westlake* Hospital *]*$_{\mathbf{ORG}}$*, an* arm *of [Resurrection* Health Care *]*$_{\mathbf{ORG}}$*, in* west suburban *[Chicagoland]*$_{\mathbf{ORG}}$.

Figure 1: Lexical Feature Selection – All lexical elements with SRWL membership used to derive features are boxed. Elements used for the incorrect prediction for *Chicagoland* are double-boxed. The expert may select any boxed element for SRWL validation.

Chicagoland and the lexical elements used to derive features for this prediction are emphasized with a double-box for expository purposes. The expert selects lexical elements which they believe will result in good feature abstractions; the querying function must present examples believed to have high impact.

3.1.2 Word List Validation (Lines 11 &12)

Once the domain expert has selected a lexical element for SRWL feature abstraction, they are presented with the SRWL $\mathcal{W}$ to validate membership for the target application as shown in Figure 2. In this particular case, the expert has chosen to perform two interactions, namely for the lexical elements *west* and *suburban*. Once they have chosen which words and phrases will be included in this particular feature abstraction, $\mathcal{W}$ is updated and the associated features are replaced with their SRWL counterpart. For example, $\Phi_{text=west}$, $\Phi_{text=north}$, etc. would all be replaced with $\Phi_{text \in \mathcal{W}_{A1806}}$ later in lines 13 & 14.

A1806: southeast, northeast, south
southeast, northeast, south, north, southwest, west, east, northwest, ~~inland~~, ~~outside~~

A1558: suburban, ~~nearby~~, downtown
suburban, ~~nearby~~, downtown, urban, metropolitan, ~~neighboring~~, ~~near~~, coastal

Figure 2: Word List Validation – Completing two domain expert interactions. Upon selecting either double-boxed element in Figure 1, the expert validates the respective SRWL for feature extraction.

Accurate sense disambiguation is helpful for effective SRWL feature abstraction to manage situations where lexical elements belong to multiple lists. In this work, we first disambiguate by predicted part of speech (POS) tags. In cases of multiple SRWL senses for a POS, the given SRWLs (Pantel and Lin, 2002) rank list elements according their semantic representativeness which we use to return the highest ranked sense for a particular lexical element. Also, as SRWL resources emphasize recall over precision, we reduce expert effort by using the Google n-gram counts (Brandts and Franz, 2006) to automatically prune SRWLs.

3.2 Querying Function (Line 6)

A primary contribution of this work is designing an appropriate querying function. In doing so, we look to maximize the impact of interactions while minimizing the total number. Therefore, we look to select instances for which (1) the current hypothesis indicates the feature space is insufficient and (2) the resulting SRWL feature abstraction will help improve performance. To account for these two somewhat orthogonal goals, we design two querying functions and aggregate their results.

3.2.1 Hypothesis-Driven Querying

To find areas of the feature space which are believed to require more descriptiveness, we look to emphasize those instances which will result in the largest updates to the hypothesis. To accomplish this, we adopt an idea from the active learning community and score instances according to their margin relative to the current learned hypothesis, $\rho(f_t, x_i, y_i)$ (Tong and Koller, 2001). This results in the *hypothesis-driven* querying function

$$\mathcal{Q}_{margin} = \underset{i=1,\ldots,m}{\operatorname{argsort}} \rho(f_t, x_i, y_i)$$

where the $\operatorname{argsort}$ operator is used to sort the input elements in ascending order (for multiple instance selection). Unlike active learning, where selection is from an unlabeled data source, the quantity of labeled data is fixed and labeled data is selected during each round. Therefore, we use the true margin and not the expected margin. This means that we will first select instances which have large mistakes, followed by those instances with small mistakes, and finally instances that make correct predictions in the order of their confidence.

3.2.2 SRWL-Driven Querying

An equally important goal of the querying function is to present examples which will result in SRWL feature abstractions of broad usability. Intuitively, there are two criteria distinguishing desirable SRWLs for this purpose. First of all, large lists are desirable as there are many lists of cities, countries, corporations, etc. which are extremely informative. Secondly, preference should be given to lists where the distribution of lexical elements within a particular word list, $\epsilon \in \mathcal{W}$, is more uniform. For example, consider $\mathcal{W} = \{devour, feed\_on, eat, consume\}$. While all of these terms belong to the same SRWL, learning features based on *eat* is sufficient to cover most examples. To derive a *SRWL-driven* querying function based on these principles, we use the word list entropy, $H(\mathcal{W}) = -\sum_{\epsilon \in \mathcal{W}} p(\epsilon) \log p(\epsilon)$. The querying score for a sentence is determined by its highest entropy lexical element used for feature extraction, resulting in the querying function

$$\mathcal{Q}_{entropy} = \underset{i=1,\ldots,m}{\text{argsort}} \left[\underset{\epsilon \sim \Phi_{x_i}}{\text{argmin}} -H(\mathcal{W}_\epsilon) \right]$$

This querying function is supported by the underlying assumption of SRWL abstraction is that there exists a true feature space $\Phi^*(x)$ which is built upon SRWLs and lexical elements but is being approximated by $\Phi(x)$, which doesn't use semantic information. In this context, a lexical feature provides one bit of information to the prediction function while a SRWL feature provides information content proportional to its SRWL entropy $H(\mathcal{W})$.

To study one aspect of this phenomena empirically, we examine the rate at which words are first encountered in our training corpus from Section 4, as shown by Figure 3. The first observation is the usefulness of SRWL feature abstraction in general as we see that when including an entire SRWL from (Pantel and Lin, 2002) whenever the first element of the list is encountered, we cover the unigram vocabulary much more rapidly. The second observation is that when sentences are presented in the order of the average SRWL entropy of their words, this coverage rate is further accelerated. Figure 3 helps explain the recall focused aspect of SRWL abstraction while we rely on hypothesis-driven querying to target interactions for the specific task at hand.

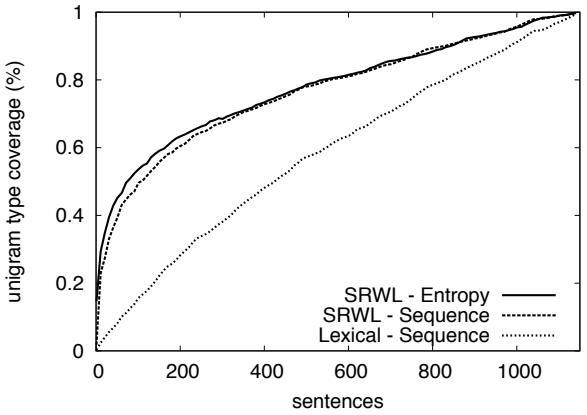

Figure 3: The Impact of SRWL Abstraction and SRWL-driven Querying – The first occurrence of words occur at a much lower rate than the first occurrence of words when abstracted through SRWLs, particularly when sentences are introduced as ranked by average SRWL entropy calculated using (Brandts and Franz, 2006).

3.2.3 Aggregating Querying Functions

To combine these two measures, we use the Borda count method of rank aggregation (Young, 1974) to find a consensus between the two querying functions without requiring calibration amongst the actual ranking scores. Defining the rank position of an instance by $r(x)$, the Borda count based querying function is stated by

$$\mathcal{Q}_{Borda} = \underset{i=1,\ldots,m}{\text{argsort}} [r_{margin}(x_i) + r_{entropy}(x_i)]$$

$\mathcal{Q}_{Borda}$ selects instances which consider both wide applicability through $r_{entropy}$ and which focus on the specific task through r_{margin}.

4 Experimental Evaluation

To demonstrate the IFSC protocol on a practical application, we examine a three-stage pipeline model for entity and relation extraction, where the task is decomposed into sequential stages of segmentation, entity classification, and relation classification (Roth and Small, 2008). Extending the standard classification task, a pipeline model decomposes the overall classification into a sequence of D stages such that each stage $d = 1, \ldots, D$ has access to the input instance along with the classifications from all previous stages, $\hat{y}^{(d)}$. Each stage of the pipeline model uses a feature vector generating procedure

$\Phi^{(d)}(x, \hat{y}^{(0)}, \ldots, \hat{y}^{(d-1)}) \rightarrow \mathbf{x}^{(d)}$ to learn a hypothesis $h^{(d)}$. Once each stage of the pipelined classifier is learned, predictions are made sequentially, where

$$\hat{\mathbf{y}} = h(x) = \left\langle \underset{y' \in \mathcal{Y}^{(d)}}{\operatorname{argmax}} f^{(d)}\left(\mathbf{x}^{(d)}, y'\right) \right\rangle_{d=1}^{D}$$

Each pipeline stage requires a classifier which makes multiple interdependent predictions based on input from multiple sentence elements $x \in \mathcal{X}_1 \times \cdots \times \mathcal{X}_{n_x}$ using a structured output space, $y^{(d)} \in \mathcal{Y}_1^{(d)} \times \cdots \times \mathcal{Y}_{n_y}^{(d)}$. More specifically, segmentation makes a prediction for each sentence word over $\mathcal{Y} \in \{begin, inside, outside\}$ and constraints are enforced between predictions to ensure that an *inside* label can only follow a *begin* label. Entity classification begins with the results of the segmentation classifier and classifies each segment into $\mathcal{Y} \in \{person, location, organization\}$. Finally, relation classification labels each predicted entity pair with $\mathcal{Y} \in \{located\_in,\ work\_for,\ org\_based\_in,\ live\_in,\ kill\} \times \{left, right\} + no\_relation$.

The data used for empirical evaluation was taken from (Roth and Yih, 2004) and consists of 1436 sentences, which is split into a 1149 (80%) sentence training set and a 287 (20%) sentence testing set such that all have at least one active relation. SRWLs are provided by (Pantel and Lin, 2002) and experiments were conducted using a custom graphical user interface (GUI) designed specifically for the IFSC protocol. The learning algorithm used for each stage of the classification task is a regularized variant of the structured Perceptron (Collins, 2002). Resources used to perform experiments are available at `http://L2R.cs.uiuc.edu/~cogcomp/`.

We extract features in a method similar to (Roth and Small, 2008), except that we do not include gazetteer features in $\Phi_0^{(d)}$ as we will include this type of external information interactively. Secondly, we use SRWL features as introduced. The segmentation features include the word/SRWL itself along with the word/SRWL of three words before and two words after, bigrams of the word/SRWL surrounding the word, capitalization of the word, and capitalization of its neighbor on each side. Entity classification uses the segment size, the word/SRWL members within the segment, and a window of two word/SRWL elements on each side. Relation classification uses the same features as entity classification along with the entity labels, the length of the entities, and the number of tokens between them.

4.1 Interactive Querying Function

When using the interactive feature space construction protocol for this task, we require a querying function which captures the hypothesis-driven aspect of instance selection. We observed that basing $\mathcal{Q}_{margin}$ on the relation stage performs best, which is not surprising given that this stage makes the most mistakes, benefits the most from semantic information, and also has many features which are similar to features from previous stages. Therefore, we adapt the querying function described by (Roth and Small, 2008) for the relation classification stage and define our margin for the purposes of instance selection as

$$\rho_{relation} = \min_{i=1,\ldots,n_y} \left[f_{y_+}(\mathbf{x}, i) - f_{\dot{y}_+}(\mathbf{x}, i) \right]$$

where $\dot{y} = \operatorname{argmax}_{y' \in \mathcal{Y} \setminus y} f_{y'}(\mathbf{x})$, the highest scoring class which is not the true label, and $\mathcal{Y}_+ = \mathcal{Y} \setminus no\_relation$.

4.2 Interactive Protocol on Entire Data Set

The first experiments we conduct uses all available training data (i.e. $|\mathcal{S}| = 1149$) to examine the improvement achieved with a fixed number of IFSC interactions. A single interaction is defined by the expert selecting a lexical element from a sentence presented by the querying function and validating the associated word list. Therefore, it is possible that a single sentence may result in multiple interactions.

The results for this experimental setup are summarized in Table 1. For each protocol configuration, we report F1 measure for all three stages of the pipeline. As our simplest baseline, we first train using the default feature set without any semantic features (**Lexical Features**). The second baseline is to replace all instances of any lexical element with its SRWL representation as provided by (Pantel and Lin, 2002) (**Semantic Features**). The next two baselines attempt to automatically increase precision by defining each semantic class using only the top fraction of the elements in each SRWL (**Pruned Semantic (top {1/2,1/4})**). This pruning procedure often results in smaller SRWLs with a more precise specification of the semantic concept.

	Lexical Features	Semantic Features	Pruned Semantic (top 1/2)	Pruned Semantic (top 1/4)	50 interactions Interactive (select only)	50 interactions Interactive (select & validate)
Segmentation	90.23	90.14	90.77	89.71	92.24	**93.43**
Entity Class.	82.17	83.28	83.93	83.04	85.81	**88.76**
Relation Class.	54.67	55.20	56.34	56.21	59.14	**62.08**

Table 1: Relative performance of the stated experiments conducted over the entire available dataset. The interactive feature construction protocol outperforms all non-interactive baselines, particularly for later stages of the pipeline while requiring only 50 interactions.

Finally, we consider the interactive feature space construction protocol at two different stages. We first consider the case where 50 interactions are performed such that the algorithm assumes $\mathcal{W}^* = \mathcal{W}$, that is, the expert selects features for abstraction, but doesn't perform validation (**Interactive (select only)**). The second experiment performs the entire protocol, including validation (**Interactive (select & validate)**) for 50 interactions. On the relation extraction task, we observe a 13.6% relative improvement over the lexical model and a 10.2% relative improvement over the best SRWL baseline F1 score.

4.3 Examination of the Querying Function

As stated in section 3.2, an appropriate querying function presents sentences which will result in the expert selecting features from that example and for which the resulting interactions will result in a large performance increase. The former is difficult to model, as it is dependent on properties of the sentence (such as length), will differ from user to user, and anecdotally is negligibly different for the three querying functions for earlier interactions. However, we are able to measure the performance improvement of interactions associated with different querying functions. For our second experiment, we evaluate the relative performance of the three querying functions defined after every ten interactions in terms of the F1 measure for relation extraction. The results of this experiment are shown in figure 4, where we first see that the $\mathcal{Q}_{random}$ generally leads to the least useful interactions. Secondly, while $\mathcal{Q}_{entropy}$ performs well early, $\mathcal{Q}_{margin}$ works better as more interactions are performed. Finally, we also observe that $\mathcal{Q}_{Borda}$ exceeds the performance envelope of the two constituent querying functions.

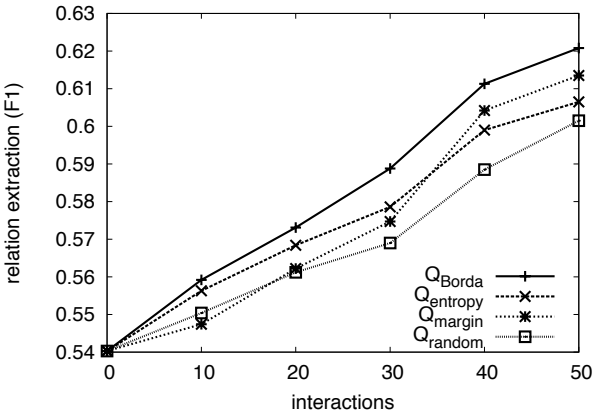

Figure 4: Relative performance of interactions generated through the respective querying functions. We see that $\mathcal{Q}_{entropy}$ performs well for a small number of interactions, $\mathcal{Q}_{margin}$ performs well as more interactions are performed and $\mathcal{Q}_{Borda}$ outperforms both consistently.

4.4 Robustness to Reduced Annotation

The third set of experiments consider the relative performance of the configurations from the first set of experiments as the amount of available training data is reduced. To study this scenario, we perform the same set of experiments with 50 interactions while varying the size of the training set (e.g. $|\mathcal{S}| = \{250, 500, 600, 675, 750, 1000\}$), summarizing the results in Figure 5. One observation is that the interactive feature space construction protocol outperforms all other configurations at all annotation levels. A second important observation is made when comparing these results to those presented in (Roth and Small, 2008), where this data is labeled using active learning. In (Roth and Small, 2008), once 65% of the labeled data is observed, a performance level is achieved comparable to training on the entire labeled dataset. In this work, an interpo-

lation of the performance at 600 and 675 labeled instances implies that we achieve a performance level comparable to training on all of the data of the baseline learner while about 55% of the labeled data is observed at random. Furthermore, as more labeled data is introduced, the performance continues to improve with only 50 interactions. This supports the hypothesis that a good representation is often more important than additional training data, even when the data is carefully selected.

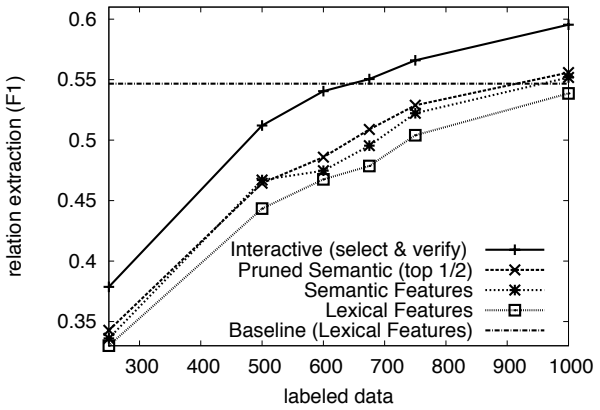

Figure 5: Relative performance of several baseline algorithm configurations and the interactive feature space construction protocol with variable labeled dataset sizes. The interactive protocol outperforms other baseline methods in all cases. Furthermore, the interactive protocol **(Interactive)** outperforms the baseline lexical system **(Baseline)** trained on all 1149 sentences even when trained with a significantly smaller subset of labeled data.

5 Related Work

There has been significant recent work on designing learning algorithms which attempt to reduce annotation requirements through a more sophisticated annotation method. These methods allow the annotator to directly specify information about the feature space in addition to providing labels, which is then incorporated into the learning algorithm (Huang and Mitchell, 2006; Raghavan and Allan, 2007; Zaidan et al., 2007; Druck et al., 2008; Zaidan and Eisner, 2008). Additionally, there has been recent work using explanation-based learning techniques to encode a more expressive feature space (Lim et al., 2007). Amongst these works, the only interactive learning protocol is (Raghavan and Allan, 2007) where instances are presented to an expert and features are labeled which are then emphasized by the learning algorithm. Thus, in this case, although additional information is provided the feature space itself remains static. To the best of our knowledge, this is the first work that interactively modifies the feature space by abstracting the FGFs.

6 Conclusions and Future Work

This work introduces the *interactive feature space construction* protocol, where the learning algorithm selects examples for which the feature space is believed to be deficient and uses existing semantic resources in coordination with a domain expert to abstract lexical features with their SRWL names. While the power of SRWL abstraction in terms of sample complexity is evident, incorporating this information is fraught with pitfalls regarding the introduction of additional ambiguity. This interactive protocol finds examples for which the domain expert will recognize promising semantic abstractions and for which those semantic abstraction will significantly improve the performance of the learner. We demonstrate the effectiveness of this protocol on a named entity and relation extraction system.

As a relatively new direction, there are many possibilities for future work. The most immediate task is effectively quantifying interaction costs with a user study, including the impact of including users with varying levels of expertise. Recent work on modeling the costs of the active learning protocol (Settles et al., 2009; Haertel et al., 2009) provides some insight on modeling costs associated with interactive learning protocols. A second potentially interesting direction would be to incorporate other semantic resources such as lexical patterns (Hearst, 1992) or Wikipedia-generated gazetteers (Toral and Muñoz, 2006).

Acknowledgments

The authors would like to thank Ming-Wei Chang, Margaret Fleck, Julia Hockenmaier, Alex Klementiev, Ivan Titov, and the anonymous reviewers for their valuable suggestions. This work is supported by DARPA funding under the Bootstrap Learning Program and by MIAS, a DHS-IDS Center for Multimodal Information Access and Synthesis at UIUC.

References

Avrim Blum. 1992. Learning boolean functions in an infinite attribute space. *Machine Learning*, 9(4):373–386.

Lois Boggess, Rajeev Agarwal, and Ron Davis. 1991. Disambiguation of prepositional phrases in automatically labelled technical text. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 155–159.

Thorsten Brandts and Alex Franz. 2006. Web 1T 5-gram Version 1.

David Cohn, Les Atlas, and Richard Ladner. 1994. Improving generalization with active learning. *Machine Learning*, 15(2):201–222.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proc. of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*, pages 1–8.

Gregory Druck, Gideon Mann, and Andrew McCallum. 2008. Learning from labeled features using generalization expectation criteria. In *Proc. of International Conference on Research and Development in Information Retrieval (SIGIR)*, pages 595–602.

Christiane Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. MIT Press.

Robbie Haertel, Kevin D. Seppi, Eric K. Ringger, and James L. Carroll. 2009. Return on investment for active learning. In *NIPS Workshop on Cost Sensitive Learning*.

Marti A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proc. the International Conference on Computational Linguistics (COLING)*, pages 539–545.

Yifen Huang and Tom M. Mitchell. 2006. Text clustering with extended user feedback. In *Proc. of International Conference on Research and Development in Information Retrieval (SIGIR)*, pages 413–420.

Xin Li and Dan Roth. 2005. Learning question classifiers: The role of semantic information. *Journal of Natural Language Engineering*, 11(4).

Siau Hong Lim, Li-Lun Wang, and Gerald DeJong. 2007. Explanation-based feature construction. In *Proc. of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 931–936.

Scott Miller, Jethran Guinness, and Alex Zamanian. 2004. Name tagging with word clusters and discriminative training. In *Proc. of the Annual Meeting of the North American Association of Computational Linguistics (NAACL)*, pages 337–342.

Patrick Pantel and Dekang Lin. 2002. Discovering word senses from text. In *Proc. of the International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 613–619.

Hema Raghavan and James Allan. 2007. An interactive algorithm for asking and incorporating feature feedback into support vector machines. In *Proc. of International Conference on Research and Development in Information Retrieval (SIGIR)*, pages 79–86.

Dan Roth and Kevin Small. 2008. Active learning for pipeline models. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 683–688.

Dan Roth and Wen-Tau Yih. 2004. A linear programming formulation for global inference in natural language tasks. In *Proc. of the Annual Conference on Computational Natural Language Learning (CoNLL)*, pages 1–8.

Burr Settles, Mark Craven, and Lewis Friedland. 2009. Active learning with real annotation costs. In *NIPS Workshop on Cost Sensitive Learning*.

Simon Tong and Daphne Koller. 2001. Support vector machine active learning with applications to text classification. *Journal of Machine Learning Research*, 2:45–66.

Antonio Toral and Rafael Muñoz. 2006. A proposal to automatically build and maintain gazetteers using wikipedia. In *Proc. of the Annual Meeting of the European Association of Computational Linguistics (EACL)*, pages 56–61.

H. Peyton Young. 1974. An axiomatization of borda's rule. *Journal of Economic Theory*, 9(1):43–52.

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

Omar Zaidan, Jason Eisner, and Christine Piatko. 2007. Using "annotator rationales" to improve machine learning for text categorization. In *Proc. of the Annual Meeting of the North American Association of Computational Linguistics (NAACL)*, pages 260–267.