
Machine Learning with Annotator Rationales to Reduce Annotation Cost

Omar F. Zaidan

Department of Computer Science
Johns Hopkins University
Baltimore, MD 21218
ozaidan@cs.jhu.edu

Jason Eisner

Department of Computer Science
Johns Hopkins University
Baltimore, MD 21218
jason@cs.jhu.edu

Christine D. Piatko

Human Language Technology Center of Excellence
Johns Hopkins University
Baltimore, MD 21211
Christine.Piatko@jhuapl.edu

Abstract

We review two novel methods for text categorization, based on a new framework that utilizes richer annotations that we call *annotator rationales*. A human annotator provides hints to a machine learner by highlighting contextual “rationales” in support of each of his or her annotations. We have collected such rationales, in the form of substrings, for an existing document sentiment classification dataset [1]. We have developed two methods, one discriminative [2] and one generative [3], that use these rationales during training to obtain significant accuracy improvements over two strong baselines. Our generative model in particular could be adapted to help learn other kinds of probabilistic classifiers for quite different tasks. Based on a small study of annotation speed, we posit that for some tasks, providing rationales can be a more fruitful use of an annotator’s time than annotating more examples.

1 Introduction

Annotation cost is a bottleneck for many natural language processing applications. While supervised machine learning systems are effective, it is labor-intensive and expensive to construct the many training examples needed. Previous research has explored possible ways to lessen this burden, such as active [4] or semi-supervised [5] learning or adaptation from a different but related domain [6].

A rather different paradigm is to change the actual *task* that is given to annotators, giving them a greater hand in shaping the learned classifier. After all, human annotators themselves are more than just black-box classifiers to be run on training data. They possess some introspective knowledge about their own classification procedure. Our hope is to mine this knowledge rapidly via appropriate questions and use it to help train a machine classifier.

Annotators currently indicate *what* the correct answers are on training data. We propose that they should also provide coarse hints about *why*. Since annotators do not know what features a future machine learner will use, we propose that they should simply *highlight relevant portions of the example*, e.g., substrings or spatial regions that help to justify their annotations. We offer two learning approaches that benefit from these “rationales” at training time.

In some circumstances, rationales should not be too expensive or time-consuming to collect. As long as the annotator is spending the time to study example x_i and classify it, it may not require much extra effort to indicate some reasons along the way. We show this experimentally in one domain.

2 Rationale Annotation for Movie Reviews

In order to demonstrate that annotator rationales help machine learning, we chose a dataset that would be enjoyable to re-annotate: the movie review dataset of [1].¹ The dataset consists of 1000 positive and 1000 negative movie reviews obtained from the Internet Movie Database (IMDb) review archive, all written before 2002 by a total of 312 authors, with a cap of 20 reviews per author per category. Pang and Lee have divided the 2000 documents into 10 folds, each consisting of 100 positive reviews and 100 negative reviews. These gold-standard classifications were derived from the number of “stars” assigned by each review’s author.

The original dataset is arguably artificial in that it keeps only reviews where the reviewer provided a rather high or rather low numerical rating, allowing Pang and Lee to designate the review as positive or negative. Nonetheless, most reviews contain a difficult mix of praise, criticism, and factual description. In fact, it is possible for a mostly critical review to give a positive overall recommendation, or vice versa.

2.1 Annotation procedure

Rationale annotators were given guidelines² that instructed them to read reviews and justify *why* a review is positive or negative by highlighting rationales for the document’s class—for a positive review, phrases that would tell someone to see the movie; for a negative review, phrases that would tell someone *not* to see the movie. The instructions provided only light guidance on how *many* rationales to annotate, encouraging annotators to do their best to mark enough rationales to provide convincing support, but emphasizing that they need not go out of their way to mark everything.

Annotators were shown the following examples³ of positive and negative rationales, among others:

- **you will enjoy the hell out of** American Pie. (*Positive*)
- he is **one of the most exciting martial artists on the big screen**, continuing to perform his own stunts and **dazzling audiences** with his flashy kicks and punches. (*Positive*)
- the romance was **enchanting**. (*Positive*)
- A woman in peril. A confrontation. An explosion. The end. **Yawn. Yawn. Yawn.** (*Negative*)
- the movie is **so badly put together** that even the most casual viewer may notice the **miserable pacing and stray plot threads**. (*Negative*)
- **don’t go see** this movie. (*Negative*)

The annotation involves **boldfacing** the rationale phrases using an HTML editor. One of the authors (A0) annotated folds 0–8 of the dataset (1,800 documents) with rationales that supported the gold-standard classes. That was our main training/development set, with fold 9 serving as the test fold.

The process of annotating rationales does not require the annotator to think about the feature space, nor even to know anything about it. Arguably this makes annotation easier and more flexible. It also preserves the reusability of the annotated data. Anyone is free to reuse our collected rationales² to aid in learning a classifier with richer features, using our methods or methods yet to be developed.

2.2 Inter-annotator agreement and annotation time

Our work in [2] considered the questions of inter-annotator agreement and annotation time. We carried out a pilot study (two of the authors) and a later, more controlled study (two paid students), where each annotator was given 150 un-annotated documents, spread equally over three tasks:

- **Task 1:** Given the document, annotate only the class (positive/negative).
- **Task 2:** Given the document and its class, annotate some rationales for that class.
- **Task 3:** Given the document, annotate both the class and some rationales for it.

¹Polarity dataset version 2.0.

²Available at <http://cs.jhu.edu/~ozaidan/rationales>.

³For our controlled study of annotation time (section 2.2), different examples were given with full document context.

The annotators’ classification accuracies in Tasks 1 and 3 (against Pang & Lee’s labels) ranged from 92%–97%, with 4-way agreement on the class for 89% of the documents, and pairwise agreement also ranging from 92%–97%. Results in [2] show they provided an average of 5–11 rationales per document, with most rationales enjoying some degree of consensus, especially those marked by the least thorough rationale annotator.

As for annotation time, we found that rationales did not take too much extra time for most annotators to provide (see [2] for exact times). In general, providing rationales only took roughly twice the time (Task 3 vs. Task 1) Why this low overhead? Because marking the class already required the Task 1 annotator to read the document and mentally find some rationales. The only extra work in Task 3 is in making them explicit. This synergy was demonstrated by the fact that doing both annotations at once (Task 3) was faster than doing them separately (Tasks 1+2).

2.3 Other scenarios

We remark that the above task—binary classification on full documents—seems to be almost a worst-case scenario for the annotation of rationales. At a purely mechanical level, it was rather heroic of A0 to attach 8–9 new rationale phrases r_{ij} to every bit y_i of ordinary annotation.

Imagine by contrast a more local task of identifying entities or relations. Each lower-level annotation y_i will tend to have fewer rationales r_{ij} , while y_i itself will be more complex and hence more difficult to mark. We expect that the overhead of collecting the rationales will be even less in such scenarios than the factor of 2 we measured. For example, [2] briefly discusses how one might conduct rationale annotation for named entities, linguistic relations, or handwritten digits.

The overhead of rationale annotation could be further reduced by asking annotators to mark rationales for only a fraction of the annotations, or fewer rationales per annotation, since additional rationales may yield diminishing returns (see section 6.2). As a special case, for a multi-class problem like relation detection, one could ask the annotator to provide rationales *only* for the rarer classes. This small amount of extra time where the data is sparsest would provide extra guidance where it was most needed.

Another possibility is passive collection of rationales via eye tracking. Rationale annotations might also be collected for free via game play. In the visual domain, when classifying an image as containing a zoo, the annotator might circle some animals or cages and the sign reading “Zoo.” The Peekaboom game [7] was in fact built to elicit such approximate yet relevant regions of images.

3 A Discriminative Approach to Utilizing Rationales

We first suggest a modification to the training of a Support Vector Machine (SVM) to incorporate rationales. From the rationale annotations on a positive example \vec{x}_i , we construct several “not-quite-as-positive” *contrast examples* \vec{v}_{ij} . Each contrast document \vec{v}_{ij} is obtained by starting with the original and “masking out” one of the rationale substrings (r_{ij}). The intuition: the *correct* model should be less sure of a positive classification on the contrast example \vec{v}_{ij} than on the original example \vec{x}_i , because \vec{v}_{ij} lacks evidence the annotator found significant.

We translate this intuition into additional constraints on the correct model: in addition to the usual SVM constraint on positive examples that $\vec{w} \cdot \vec{x}_i \geq 1$, we also want (for each j) that $\vec{w} \cdot \vec{x}_i - \vec{w} \cdot \vec{v}_{ij} \geq \mu$, where $\mu \geq 0$ controls the size of the desired margin between original and contrast examples. In other words, an ordinary soft-margin SVM chooses \vec{w} and ξ to minimize

$$\frac{1}{2} \|\vec{w}\|^2 + C \left(\sum_i \xi_i \right) \tag{1}$$

subject to the constraints

$$(\forall i) \quad \vec{w} \cdot \vec{x}_i \cdot y_i \geq 1 - \xi_i \tag{2}$$

$$(\forall i) \quad \xi_i \geq 0 \tag{3}$$

where \vec{x}_i is a training example, $y_i \in \{-1, +1\}$ is its desired classification, and ξ_i is a slack variable that allows training example \vec{x}_i to miss satisfying the margin constraint if necessary. We add the *contrast constraints*

$$(\forall i, j) \quad \vec{w} \cdot (\vec{x}_i - \vec{v}_{ij}) \cdot y_i \geq \mu(1 - \xi_{ij}), \tag{4}$$

where \vec{v}_{ij} is one of the contrast examples constructed from example \vec{x}_i , and $\xi_{ij} \geq 0$ is an associated slack variable. Just as these extra constraints have their own margin μ , their slack variables have their own cost, so the objective function (1) becomes

$$\frac{1}{2} \|\vec{w}\|^2 + C \left(\sum_i \xi_i \right) + C_{contrast} \left(\sum_{i,j} \xi_{ij} \right) \quad (5)$$

The parameter $C_{contrast} \geq 0$ determines the importance of satisfying the contrast constraints. It should generally be less than C if the rationales are noisier than the training examples.

In practice, it is possible to solve this optimization using a standard soft-margin SVM learner. Dividing equation (4) through by μ , it becomes

$$(\forall i, j) \quad \vec{w} \cdot \frac{\vec{x}_i - \vec{v}_{ij}}{\mu} \cdot y_i \geq 1 - \xi_{ij}, \quad (6)$$

where $\frac{\vec{x}_i - \vec{v}_{ij}}{\mu} \stackrel{\text{def}}{=} \vec{x}_{ij}$. Since equation (6) takes the same form as equation (2), we simply add the pairs (\vec{x}_{ij}, y_i) to the training set as *pseudoexamples*, weighted by $C_{contrast}$ rather than C so that the learner will use the objective function (5). In our experiments, we optimize μ , C , and $C_{contrast}$ on held-out data (see subsection 5.1).

For our feature set, we exactly follow [1] in merely using binary unigram features, corresponding to the 17,744 unstemmed word or punctuation types with count ≥ 4 in the full 2000-document corpus. Thus, each document is reduced to a 0-1 vector with 17,744 dimensions (normalized to unit length).

4 A Generative Approach to Utilizing Rationales

Unlike our discriminative approach, our generative approach relies on an explicit, parametric model of the relationship between the rationales and the optimal classifier parameters θ (i.e., that we would learn on an infinite training set). This auxiliary model’s parameters, ϕ , capture what the annotator is doing when marking rationales. Most importantly, they capture how he or she is influenced by the true θ . Our learning method will prefer values for θ that would adequately explain the rationales r (as well as explaining class labels y). To that end, we jointly choose parameter vectors θ and ϕ to maximize the following regularized conditional likelihood:

$$\prod_{i=1}^n p(y_i, r_i | x_i, \theta, \phi) \cdot p_{\text{prior}}(\theta, \phi) \stackrel{\text{def}}{=} \prod_{i=1}^n p_{\theta}(y_i | x_i) \cdot p_{\phi}(r_i | x_i, y_i, \theta) \cdot p_{\text{prior}}(\theta, \phi) \quad (7)$$

Here we are trying to model all the annotations, both y_i and r_i . The first factor predicts y_i using an ordinary probabilistic classifier p_{θ} , while the novel second factor predicts r_i using some low-dimensional model p_{ϕ} of how a particular annotator generates the rationale annotations.

The crucial point is that the second factor depends on θ (since r_i is supposed to reflect the relation between x_i and y_i that is modeled by θ). As a result, the learner has an incentive to modify θ in a way that increases the second factor, even if this somewhat decreases the first factor on training data.⁴ After training (subsection 5.2), we simply use the first factor $p_{\theta}(y | x)$ to classify test documents x . The second factor is irrelevant for test documents, since they have not been annotated with rationales r . The second factor may likewise be omitted for any training documents i that have not been annotated with rationales, as there is no r_i to predict in those cases. In the extreme case where no documents are annotated with rationales, equation (1) reduces to the standard training procedure.

4.1 Main model: Modeling class annotations with p_{θ}

Let us first define the main classifier p_{θ} in equation (1) to be a standard conditional log-linear model:⁵

$$p_{\theta}(y | x) \stackrel{\text{def}}{=} \frac{\exp(\vec{\theta} \cdot \vec{f}(x, y))}{Z_{\theta}(x)} \stackrel{\text{def}}{=} \frac{u(x, y)}{Z_{\theta}(x)} \quad (8)$$

⁴Interestingly, even examples where the annotation y_i is wrong or unhelpful can provide useful information about θ via the pair (y_i, r_i) . Two annotators marking the same movie review might disagree on whether it is overall a positive or negative review—but the second factor still allows learning positive features from the first annotator’s positive rationales, and negative features from the second annotator’s negative rationales.

⁵In our binary classification setting ($y \in \{-1, +1\}$), equation (8) is equivalent to logistic regression. We nonetheless give the multi-class formulation (8), both for generality and for consistency with equation (9).

Original text: (from a $y = +1$ review)		"51 weeks into '98, a champ has emerged. The Prince of Egypt succeeds where other movies failed."									
\vec{r}		○	○	○	○	○	I	I	I	I	○
\vec{x}		51	weeks	into	'98	,	a	champ	has	emerged	.
$y \cdot \theta_{x_m}$		-0.01	+0.05	-0.09	0.00	-0.05	+0.04	-0.01	+0.08	+0.02	-0.01
$B_{10}(y \cdot \theta_{x_m})$		0.00	+0.03	-0.03	0.00	-0.02	<u>+0.02</u>	<u>0.00</u>	<u>+0.05</u>	<u>+0.01</u>	0.00
\vec{r}		○	○	○	○	I	I	I	I	I	○
\vec{x}		the	prince	of	egypt	succeeds	where	other	movies	failed	.
$y \cdot \theta_{x_m}$		0.00	-0.03	-0.06	+0.02	+0.13	-0.08	+0.06	+0.11	-0.15	-0.01
$B_{10}(y \cdot \theta_{x_m})$		0.00	-0.02	-0.02	+0.01	<u>+0.09</u>	<u>-0.03</u>	<u>+0.03</u>	<u>+0.07</u>	<u>-0.05</u>	0.00

$g_{O-I} = 2$	$g_{O(.)-I} = 0$	$g_{O-I(.)} = 0$	$g_{O(,) - I} = 1$	$g_{O-I(,)} = 0$
$g_{I-O} = 2$	$g_{I(.)-O} = 0$	$g_{I-O(.)} = 2$	$g_{I(,) - O} = 0$	$g_{I-O(,)} = 0$
$g_{O-O} = 8$				
$g_{I-I} = 7$				
$g_{rel} = 0.19$				

Figure 1: Rationales as sequence annotation: the annotator highlighted two textual segments as rationales for a positive class. Highlighted words in \vec{x} are tagged I in \vec{r} , and other words are tagged O. The figure also shows some ϕ -features. For instance, $g_{O(.)-I}$ is a count of O-I transitions that occur with a comma as the left word. Notice also that g_{rel} is the sum of the underlined values.

where $\vec{f}(\cdot)$ extracts a feature vector from a classified document, $\vec{\theta}$ are the corresponding weights of those features, and $Z_\theta(x) \stackrel{\text{def}}{=} \sum_y u(x, y)$ is a normalizer. We use the same set of binary unigram features as in section 3. That is, define $f_w(x, y)$ to be y if word type w appears at least once in x , and 0 otherwise. Thus $\theta \in \mathbb{R}^{17744}$, and positive weights in θ favor class label $y = +1$ and equally discourage $y = -1$, while negative weights do the opposite.

4.2 Rationales as a noisy channel

The interesting part of our model is $p_\phi(r | x, y, \theta)$, which models the rationale annotation process. *The rationales r reflect θ , but in noisy ways.* $p_\phi(r | x, y, \theta)$ should consider two questions when assessing whether r is a plausible set of rationales given θ . First, it needs a “language model” of rationales: does r consist of rationales that are well-formed *a priori*, i.e., before θ is considered? Second, it needs a “channel model”: does r faithfully signal the features of θ that strongly support classifying x as y ?

If a feature contributes heavily to the classification of document x as class y , then the “channel model” should tell us which *parts* of document x tend to be highlighted as a result. The channel model must know about the particular kinds of features that are extracted by f and scored by θ . The “language model,” however, is independent of the feature set θ . It models what rationales tend to look like. In our document task, ϕ should describe things like: How frequent and how long are typical rationales? Do their edges tend to align with punctuation or major syntactic boundaries in x ?

4.3 Auxiliary model: Modeling rationale annotations with p_ϕ

The rationales collected in this task are textual segments of a document to be classified. The document itself is a word token sequence $\vec{x} = x_1, \dots, x_M$. We encode its rationales as a corresponding tag sequence $\vec{r} = r_1, \dots, r_M$, as illustrated in Figure 1. Here $r_m \in \{I, O\}$ according to whether the token x_m is **in** a rationale (i.e., x_m was at least partly highlighted) or **outside** all rationales. x_1 and x_M are special boundary symbols, tagged with O. We predict the full tag sequence \vec{r} at once using a conditional random field [8]. A CRF is just another conditional log-linear model:

$$p_\phi(r | x, y, \vec{\theta}) \stackrel{\text{def}}{=} \frac{\exp(\vec{\phi} \cdot \vec{g}(r, x, y, \vec{\theta}))}{Z_\phi(x, y, \vec{\theta})} \stackrel{\text{def}}{=} \frac{u(r, x, y, \vec{\theta})}{Z_\phi(x, y, \vec{\theta})} \quad (9)$$

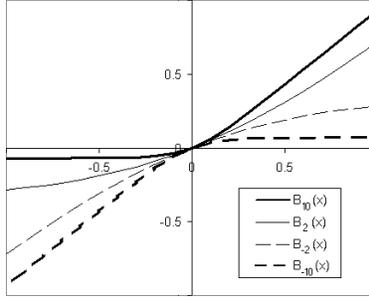


Figure 2: The function family B_s in equation (12), shown for $s \in \{10, 2, -2, -10\}$.

where $\vec{g}(\cdot)$ extracts a feature vector, $\vec{\phi}$ are the corresponding weights of those features, and $Z_\phi(x, y, \vec{\theta}) \stackrel{\text{def}}{=} \sum_r u(r, x, y, \vec{\theta})$ is a normalizer.

As usual for linear-chain CRFs, $\vec{g}(\cdot)$ extracts two kinds of features: first-order “emission” features that relate the tag r_m to (x_m, y, θ) , and second-order “transition” features that relate the tag r_m to r_{m-1} (although some of these also look at x). These two kinds of features respectively capture the “channel model” and “language model” of section 4.2. The former says r_m is \mathbb{I} because x_m is associated with a relevant θ -feature.⁶ The latter says r_m is \mathbb{I} simply because it is next to another \mathbb{I} .

4.3.1 Emission ϕ -features (“channel model”)

Recall that our θ -features (at present) correspond to unigrams. Given $(\vec{x}, y, \vec{\theta})$, let us say that a unigram $w \in \vec{x}$ is **relevant**, **irrelevant**, or **anti-relevant** if $y \cdot \theta_w$ is respectively $\gg 0$, ≈ 0 , or $\ll 0$. That is, w is relevant if its presence in x strongly supports the annotated class y , and anti-relevant if its presence strongly supports the opposite class $-y$. We would like to learn the extent ϕ_{rel} to which annotators try to *include* relevant unigrams in their rationales, and the (perhaps lesser) extent ϕ_{antirel} to which they try to *exclude* anti-relevant unigrams. This will help us infer $\vec{\theta}$ from the rationales. The details are as follows. ϕ_{rel} and ϕ_{antirel} are the weights of two emission features extracted by \vec{g} :

$$g_{\text{rel}}(\vec{x}, y, \vec{r}, \vec{\theta}) \stackrel{\text{def}}{=} \sum_{m=1}^M I(r_m = \mathbb{I}) \cdot B_{10}(y \cdot \theta_{x_m}) \quad (10)$$

$$g_{\text{antirel}}(\vec{x}, y, \vec{r}, \vec{\theta}) \stackrel{\text{def}}{=} \sum_{m=1}^M I(r_m = \mathbb{I}) \cdot B_{-10}(y \cdot \theta_{x_m}) \quad (11)$$

Here $I(\cdot)$ denotes the indicator function, returning 1 or 0 according to whether its argument is true or false. Relevance and negated anti-relevance are respectively measured by the differentiable nonlinear functions B_{10} and B_{-10} , shown in Figure 2, which are defined by

$$B_s(a) = (\log(1 + \exp(a \cdot s)) - \log(2))/s \quad (12)$$

How does this work? The g_{rel} feature is a sum over all unigrams in the document \vec{x} . It does not fire strongly on the irrelevant or anti-relevant unigrams, since B_{10} is close to zero there. But it fires positively on relevant unigrams w if they are tagged with \mathbb{I} , and the strength of such firing increases approximately linearly with θ_w . Since the weight $\phi_{\text{rel}} > 0$ in practice, this means that raising a relevant unigram’s θ_w (if $y = +1$) will proportionately raise its log-odds of being tagged with \mathbb{I} . Symmetrically, since $\phi_{\text{antirel}} > 0$ in practice, lowering an anti-relevant unigram’s θ_w (if $y = +1$) will proportionately lower its log-odds of being tagged with \mathbb{I} . See [3] for further discussion.

4.3.2 Transition ϕ -features (“language model”)

Annotators highlight more than just relevant unigrams. (After all, they aren’t aware of the feature set.) They tend to mark full phrases, and ϕ models these phrases’ shape, via weights for several “language model” features, such as the 4 traditional CRF tag transition features $g_{\mathbb{O}-\mathbb{O}}, g_{\mathbb{O}-\mathbb{I}}, g_{\mathbb{I}-\mathbb{I}}, g_{\mathbb{I}-\mathbb{O}}$.

⁶[3] sketches how to model a channel that transmits θ -features more complex than our present unigrams.

For example, $g_{\text{O}-\text{I}}$ counts the number of O-to-I transitions in \vec{r} (see Figure 1). Other things equal, an annotator with high $\phi_{\text{O}-\text{I}}$ is predicted to have many rationales. And if $\phi_{\text{I}-\text{I}}$ is high, rationales are predicted to be long phrases (including more irrelevant unigrams around relevant ones).

We also learn weights for more refined versions of these 4 basic transition features. For instance, a feature of the form $g_{t_1(v)-t_2}$ counts the number of times an t_1-t_2 transition occurs in \vec{r} **conditioned on** v appearing as the first of the two word tokens where the transition occurs. We limit v to a few frequent punctuation marks and function words. Other features condition transitions on syntactic boundaries in the text. Further motivation and full details of all the ϕ -features can be found in [3].

5 Experimental Details

5.1 Discriminative Approach

We transformed this problem to an SVM problem (as described in section 3) and used the SVM^{light} software [9] for training and testing, using the default linear kernel.

For any given value of T and any given training method, we chose hyperparameters $\theta^* = (C, \mu, C_{\text{contrast}})$ to maximize the following cross-validation performance:

$$\theta^* = \operatorname{argmax}_{\theta} \sum_{i=0}^8 \operatorname{acc}(F_i \mid \theta, F_{i+1} \cup \dots \cup F_{i+T}) \quad (13)$$

where F_j denotes the fold numbered $j \bmod 9$, and $\operatorname{acc}(Z \mid \theta, Y)$ means classification accuracy on the set Z after training on Y with hyperparameters θ . We used a simple alternating optimization procedure that begins at $\theta_0 = (1.0, 1.0, 1.0)$ and cycles repeatedly through the three dimensions, optimizing along each dimension by a local grid search with resolution 0.1. Of course, when training without rationales, we did not have to optimize μ or C_{contrast} .

5.2 Generative Approach

To train our model, we used L-BFGS to locally maximize the log of the objective function (1):⁷

$$\sum_{i=1}^n \log p_{\theta}(y_i \mid x_i) - \frac{1}{2\sigma_{\theta}^2} \|\theta\|^2 + C \left(\sum_{i=1}^n \log p_{\phi}(r_i \mid x_i, y_i, \theta) \right) - \frac{1}{2\sigma_{\phi}^2} \|\phi\|^2 \quad (14)$$

This defines p_{prior} from (1) to be a standard diagonal Gaussian prior, with variances σ_{θ}^2 and σ_{ϕ}^2 for the two sets of parameters. We optimized σ_{θ}^2 on held-out data, but always took $\sigma_{\phi}^2 = 1$. (Different values of σ_{ϕ}^2 did not affect the results, since the prior distribution over the relatively few ϕ weights was overwhelmed by the large number of observed $\{\text{I}, \text{O}\}$ rationale tags available to train them.)

Note the new C factor in equation (14). Initial experiments showed that optimizing equation (14) without C led to an increase in the likelihood of the rationale data at the expense of classification accuracy, which degraded noticeably. This is because the second sum in (14) has a much larger magnitude than the first: in a set of 100 documents, it predicts around 74,000 binary $\{\text{I}, \text{O}\}$ tags, versus the one hundred binary class labels. While we are willing to reduce the log-likelihood of the training classifications (the first sum) to a certain extent, focusing too much on modeling rationales (the second sum) is clearly not our ultimate goal, so we optimized C on development data to achieve some balance between the two terms of equation (14).⁸ Typical values of C ranged from $\frac{1}{300}$ to $\frac{1}{50}$.

We perform alternating optimization on θ and ϕ :

1. Initialize θ to maximize equation (14) but with $C = 0$ (i.e. based only on class data).

⁷One might expect this function to be convex because p_{θ} and p_{ϕ} are both log-linear models with no hidden variables. However, $\log p_{\phi}(r_i \mid x_i, y_i, \theta)$ is not necessarily convex in θ .

⁸ C also balances our confidence in the classification data y against our confidence in the rationale data r ; either may be noisy. One may regard the second half of equation (14) as a kind of regularizer that biases θ toward a solution compatible with the rationale annotations, and C controls the strength of this regularizer.

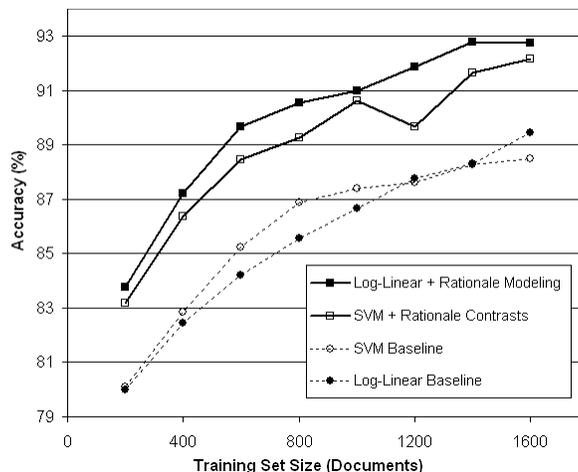


Figure 3: Classification accuracy curves for the 4 methods: the two baseline learners that only utilize class data, and the two learners that also utilize rationale annotations provided by annotator A0.

2. Fix θ , and find ϕ that maximizes equation (14).
3. Fix ϕ , and find θ that maximizes equation (14).
4. Repeat 2 and 3 until convergence.

The L-BFGS method requires calculating the gradient of the objective function (14). The partial derivatives with respect to components of θ and ϕ involve calculating expectations of the feature functions, which can be computed in linear time (with respect to the size of the training set) using the forward-backward algorithm for CRFs. The partial derivatives also involve the derivative of (12), to determine how changing θ will affect the firing strength of the emission features g_{rel} and g_{antirel} .

6 Experimental Results

Figure 3 shows learning curves for training set sizes up to 1600 documents for four methods: a log-linear baseline, an SVM baseline, the discriminative method of section 3, and the generative method of section 4. A data point in the figure reports an averaged accuracy over 9 experiments with different subsets of the training set (see [2, 3]). The top two curves indicate significant⁹ improvements in accuracy over the respective baselines when rationales are introduced using our two proposed approaches. Further, our generative method outperforms¹⁰ our masking SVM method, which starts with a slightly better baseline classifier (an SVM) but incorporates the rationales more crudely.

To show that the results generalized beyond annotator A0, we performed the same comparison for three additional annotators, A3–A5, each of whom provided class and rationale annotations on a small 100-document training set.¹¹ Experiments reported in [3] again show improvements for both methods, usually significant, over the two baselines. [2] further analyze the masking SVM’s benefit.

6.1 Analysis for the generative method

For the generative approach, examining the learned weights $\vec{\phi}$ gives insight into annotator behavior. High weights include I–O and O–I transitions conditioned on punctuation, e.g., $\phi_{\text{I}(\cdot)\text{-O}} = 3.55$,¹². The large emission feature weights, e.g., $\phi_{\text{rel}} = 14.68$ and $\phi_{\text{antirel}} = 15.30$, tie rationales closely to θ values, as hoped. For example, in Figure 1, the word $w = \text{succeeds}$, with $\theta_w = 0.13$, drives up

⁹We call a difference significant at $p < 0.05$, under a paired permutation test that is designed to be appropriate for comparing the averaged accuracies that we report (see [2] for details).

¹⁰Differences are not significant at sizes 200, 1000, and 1600.

¹¹A0 and A3–A5 were blind test data—we developed our method (e.g., ϕ -features) using annotators A6–A8.

¹²When trained on folds F_4 – F_8 with A0’s rationales.

$p(\top)/p(\circ)$ by a factor of 7 (in a positive document) relative to a word with $\theta_w = 0$. We also found that annotators’ styles differ enough that it helps to tune ϕ to the “target” annotator A who gave the rationales. Results reported in [3] show that a ϕ model trained on A ’s *own* rationales does best at predicting new rationales from A , and similarly, classification performance is usually best if it was A ’s *own* ϕ that was used to help learn θ from A ’s rationales.

Feature ablation experiments in [3] showed that almost all the classification benefit from rationales can be obtained by using only the 2 emission ϕ -features and the 4 unconditioned transition ϕ -features. Our full ϕ (115 features) merely improved our ability to predict the rationales.¹³

6.2 Using fewer rationales

In [2], we investigated improvements over the baseline if one has time to annotate rationales for only *some* documents. The key discovery is that much of the benefit can actually be obtained with relatively few rationales. For example, with 800 training documents, annotating (0%, 50%, 100%) of them with rationales gives accuracies of (86.9%, 89.2%, 89.3%), and with 1600 training documents, annotating (0%, 50%, 100%) with rationales gives (88.5%, 91.7%, 92.2%).

We also experimented with keeping a randomly chosen subset of rationales but spread out over *all* training documents. We found that, for a fixed number of rationales, it did not matter whether they were spread out or came from a subset of the documents. Spreading out the rationales simulates a “lazy annotator” less assiduous than A0. Such annotators may in fact be more desirable, as they should be able to add rationales at a higher rate (some rationales are simply more noticeable than others, and a lazy annotator will probably be marking those).¹⁴

Results in [2] also suggest that rationales and class labels may be somewhat orthogonal in their benefit. With many documents and few rationales, the learning curve starts to flatten out, but adding more rationales provides a fresh benefit: rationales have not yet reached *their* point of diminishing returns. In practice, we suggest dynamically adjusting the fraction of documents annotated with rationales. For a given annotator, should the next half-hour of annotation include rationales or not? This is a quantifiable cost-benefit tradeoff. One knows the time cost of the annotator’s most recent rationales, and one can determine their marginal benefit on accuracy using cross-validation.

7 Related Work

Our rationales resemble “side information” in machine learning—supplementary information about the target function that is available at training time. Past work generates such information, by *automatically* transforming the training examples in ways that are expected to preserve or alter the classification [10], sometimes having to manually annotate the extra examples [11]. Our approach differs because a human helps to *generate* the virtual examples. Our two methods for utilizing the virtual examples also appear new (e.g., enforcing a margin between ordinary and contrast examples).

One could instead ask annotators to examine or propose some *features* instead of rationales. In document classification, Raghavan et al. [12] show that feature selection by an oracle could be helpful, and that humans are both rapid and reasonably good at distinguishing highly useful n -gram features from randomly chosen ones. Druck et al. [13] show annotators some features f from a fixed set, and ask them to choose a class label y such that $p(y | f)$ is as high as possible. Haghghi and Klein [14] do the reverse: for each class label y , they ask the annotators to propose a few “prototypical” features f such that $p(y | f)$ is as high as possible.

However, we have several concerns about asking annotators to identify globally relevant features. First, by committing to a particular feature set at annotation time, it restricts subsequent research on the costly annotated dataset. Second, it is not clear how an annotator would easily view and highlight features in context, except for the simplest feature sets. In the phrase **Apple** shares up 3%, there may be several features that fire on the substring **Apple**—responding to the string **Apple**, its case-invariant form **apple**, its part of speech **noun**, etc. How does the annotator indicate which of these features are relevant? Third, annotating features is only appropriate when the feature set

¹³Their likelihood does increase significantly with more features. A good predictive model of rationales, p_ϕ , could be independently useful for selecting “snippets” that explain a machine’s classification to a human.

¹⁴The “most noticeable” rationales might also be the most effective ones for learning.

can be easily understood by a human. It would be hard for annotators to read, write, or evaluate a description of a convolution filter in machine vision, for instance.

8 Conclusions

We have presented two machine learning algorithms that exploit the cleverness of annotators, by considering enriched annotations that they provide. Our simple discriminative training method incorporated “annotator rationales”—on some or all of the training set—to learn a significantly better linear classifier for positive vs. negative movie reviews. Our generative approach achieved small but significant further improvements by explicitly modeling each annotator’s rationale-marking process.

Most annotators provided several rationales per classification without taking too much extra time—even in our text classification scenario, where the rationales greatly outweigh the classifications in number and complexity. To lower annotation cost further while preserving most of the empirical benefit, one should solicit rationales for only a subset of the examples. Greater speed might also be possible through an improved user interface or passive feedback (e.g., eye tracking).

In principle, many machine learning methods might be modified to exploit rationale data, as in our modified SVM. And our generative approach, being essentially Bayesian inference, is potentially extensible to many situations—other tasks, other classifier architectures, and more complex features.

Acknowledgments

This work was supported by the JHU WSE/APL Partnership Fund, by National Science Foundation grant No. 0347822 to the second author, and by a JHU APL Hafstad Fellowship.

References

- [1] B. Pang and L. Lee. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of ACL*, pages 271–278, 2004.
- [2] O. Zaidan, J. Eisner, and C. Piatko. Using “annotator rationales” to improve machine learning for text categorization. In *Proceedings of NAACL HLT*, pages 260–267, April 2007.
- [3] O. Zaidan and J. Eisner. Modeling annotators: A generative approach to learning from annotator rationales. In *Proceedings of EMNLP*, pages 31–40, October 2008.
- [4] D. Lewis and W. Gale. A sequential algorithm for training text classifiers. In *Proceedings of ACM SIGIR*, pages 3–12, 1994.
- [5] O. Chapelle, A. Zien, and B. Schölkopf, editors. *Semi-Supervised Learning*. MIT Press, 2006.
- [6] H. Daumé III and D. Marcu. Domain adaptation for statistical classifiers. *Journal of Artificial Intelligence Research (JAIR)*, 26:101–126, 2006.
- [7] L. von Ahn, R. Liu, and M. Blum. Peekaboom: A game for locating objects. In *CHI’06: Proc. of the SIGCHI Conference on Human Factors in Computing Systems*, pages 55–64, 2006.
- [8] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML*, pages 282–289, 2001.
- [9] T. Joachims. Making large-scale SVM learning practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods—Support Vector Learning*, pages 169–185. MIT Press, 1999. Software available at svmlight.joachims.org.
- [10] Y. Abu-Mostafa. Hints. *Neural Computation*, 7:639–671, July 1995.
- [11] P. Kuusela and D. Ocone. Learning with side information: PAC learning bounds. *J. of Computer and System Sciences*, 68(3):521–545, May 2004.
- [12] H. Raghavan, O. Madani, and R. Jones. Active learning on both features and instances. *Journal of Machine Learning Research*, 7:1655–1686, Aug 2006.
- [13] G. Druck, G. Mann, and A. McCallum. Learning from labeled features using generalized expectation criteria. In *Proceedings of ACM SIGIR*, 2008.
- [14] A. Haghighi and D. Klein. Prototype-driven learning for sequence models. In *Proceedings of NAACL HLT*, pages 320–327, New York City, USA, 2006.