Bayesian Optimization of Text Representations

Dani Yogatama  Lingpeng Kong
School of Computer Science  Carnegie Mellon University
Pittsburgh, PA 15213, USA
\{dyogatama,lingpenk\}@cs.cmu.edu

Noah A. Smith
Computer Science & Engineering  University of Washington
Seattle, WA 98195, USA
nasmith@cs.washington.edu

Abstract

When applying machine learning to problems in NLP, there are many choices to make about how to represent input texts. They can have a big effect on performance, but they are often uninteresting to researchers or practitioners who simply need a module that performs well. We apply sequential model-based optimization over this space of choices and show that it makes standard linear models competitive with more sophisticated, expensive state-of-the-art methods based on latent variables or neural networks on various topic classification and sentiment analysis problems. Our approach is a first step towards black-box NLP systems that work with raw text and do not require manual tuning.

1 Introduction

NLP researchers and practitioners spend a considerable amount of time comparing machine-learned models of text that differ in relatively uninteresting ways. For example, in categorizing texts, should the “bag of words” include bigrams, and is tf-idf weighting a good idea? In learning word embeddings, distributional similarity approaches have been shown to perform competitively with neural network models when the hyperparameters (e.g., context window, subsampling rate, smoothing constant) are carefully tuned (Levy et al., 2015). These choices matter experimentally, often leading to big differences in performance, with little consistency across tasks and datasets in which combination of choices works best. Unfortunately, these differences tell us little about language or the problems that machine learners are supposed to solve.

We propose that these decisions can be automated in a similar way to hyperparameter selection (e.g., choosing the strength of a ridge or lasso regularizer). Given a particular text dataset and classification task, we show a technique for optimizing over the space of representational choices, along with other “nuisances” that interact with these decisions, like hyperparameter selection. For example, using higher-order \(n\)-grams means more features and a need for stronger regularization and more training iterations. Generally, these decisions about instance representation are made by humans, heuristically; our work seeks to automate them, not unlike Daelemans et al. (2003), who proposed to use genetic algorithms to optimize representational choices.

Our technique instantiates sequential model-based optimization (SMBO; Hutter et al., 2011). SMBO and other Bayesian optimization approaches have been shown to work well for hyperparameter tuning (Bergstra et al., 2011; Hoffman et al., 2011; Snoek et al., 2012). Though popular in computer vision (Bergstra et al., 2013), these techniques have received little attention in NLP.

We apply it to logistic regression on a range of topic and sentiment classification tasks. Consistently, our method finds representational choices that perform better than linear baselines previously reported in the literature, and that, in some cases, are competitive with more sophisticated non-linear models trained using neural networks.

2 Problem Formulation and Notation

Let the training data consist of a collection of pairs \(d_{\text{train}} = \{(d.i_1,d.o_1), \ldots, (d.i_n,d.o_n)\}\), where each input \(d.i \in \mathcal{I}\) is a text document and each output \(d.o \in \mathcal{O}\), the output space. The overall training goal is to maximize a performance function \(f\) (e.g., classification accuracy, log-likelihood, \(F_1\) score, etc.) of a machine-learned model, on a held-out dataset, \(d_{\text{dev}} \in (\mathcal{I} \times \mathcal{O})^n\).

Classification proceeds in three steps: first, \(x : \mathcal{I} \rightarrow \mathbb{R}^N\) maps each input to a vector representation. Second, a predictive model (typically, its parameters) is learned from the inputs (now transformed into vectors) and outputs: \(L : (\mathbb{R}^N \times \mathcal{O})^n \rightarrow (\mathbb{R}^N \rightarrow \mathcal{O})\). Finally, the resulting classifier \(c : \mathcal{I} \rightarrow \mathcal{O}\) is fixed as \(L(d_{\text{train}}) \circ x\) (i.e., the composition of the representation function with
the learned mapping).

Here we consider linear classifiers of the form 
$c(d,i) = \arg \max_{\theta \in \Theta} w^T \theta$, where the parameters 
$w, \theta \in \mathbb{R}^N$, for each output $o$, are learned 
using logistic regression on the training data. We 
let $w$ denote the concatenation of all $w_o$. Hence 
the parameters can be understood as a function of 
the training data and the representation function 
$x$. The performance function $f$, in turn, is a function 
of the held-out data $d_{dev}$ and $x$—also $w$ and 
$d_{train}$, through $x$. For simplicity, we will write 
“$f(x)$” when the rest are clear from context.

Typically, $x$ is fixed by the model designer, perhaps 
after some experimentation, and learning focuses 
on selecting the parameters $w$. For logistic regression and 
many other linear models, this training step reduces to convex optimization in $\mathbb{R}^{|\Theta|}$ 
dimensions—a solvable problem that is costly for 
large datasets and/or large output spaces. In seeking to 
maximize $f$ with respect to $x$, we do not wish 
to carry out training any more times than necessary.

Choosing $x$ can be understood as a problem of 
selecting hyperparameter values. We therefore turn 
to Bayesian optimization, a family of techniques that 
can be used to select hyperparameter values intelligently when solving for parameters ($w$) is costly.

3 Bayesian Optimization

Our approach is based on sequential model-based 
optimization (SMBO; Hutter et al., 2011). It iteratively 
chooses representation functions $x$. On each 
round, it makes this choice through a probabilistic 
model of $f$, then evaluates $f$—we call this a “trial.” 
As in any iterative search algorithm, the goal is 
to balance exploration of options for $x$ with exploitation 
of previously-explored options, so that a good 
choice is found in a small number of trials.

More concretely, in the $t$th trial, $x_t$ is selected 
using an acquisition function $A$ and a “surrogate” 
probabilistic model $p_t$. Second, $f$ is evaluated 
given $x_t$—an expensive operation which involves 
training to learn parameters $w$ and assessing 
performance on the held-out data. Third, the surrogate 
model is updated. See Algorithm 1 details on $A$ 
and $p_t$ follow.

**Acquisition Function.** A good acquisition function 
returns high values for $x$ when either the value 
of $f(x)$ is predicted to be high, or the uncertainty 
about $f(x)$’s value is high; balancing between 
these is the classic tradeoff between exploitation 
and exploration. We use a criterion called Expected 
Improvement (EI; Jones, 2001), which is the expectation (under the current surrogate model $p_t$) 
that $f(x) = y$ will exceed $f(x^*) = y^*$:

$$A(x; p_t, y^*) = \int_{-\infty}^{\infty} \max(y - y^*, 0)p_t(y | x)dy$$

where $x^*$ is chosen depending on the surrogate 
model, discussed below. (For now, think of it as 
a strongly-performing “benchmark” discovered in 
earlier iterations.) Other options for the acquisition 
function include maximum probability of improvement 
(Jones, 2001), minimum conditional entropy 
(Villemonetix et al., 2009), Gaussian process upper 
confidence bound (Srinivas et al., 2010), or a combination of them (Hoffman et al., 2011).

**Surrogate Model.** As a surrogate model, we use 
a tree-structured Parzen estimator (TPE; Bergstra et al., 2011)\(^1\) This is a nonparametric approach to 
density estimation. We seek to estimate $p_t(y | x)$ 
where $y = f(x)$, the performance function that is 
expensive to compute exactly. The TPE approach 
seeks $p_t(y | x) \propto p_t(y) \cdot \left\{ \begin{array}{ll} p_t^c(x), & \text{if } y < y^* \\ p_t^f(x), & \text{if } y \geq y^* \end{array} \right\}$, where 
$p_t^c$ and $p_t^f$ are densities estimated using observations 
from previous trials that are less than and 
greater than $y^*$, respectively. In TPE, $y^*$ is defined 
as some quantile of the observed $y$ from previous 
trials; we use 15-quantiles.

As shown by Bergstra et al. (2011), the Expected 
Improvement in TPE can be written as:

\(^1\)EI is the most widely used acquisition function that has 
been shown to work well on a range of tasks.

\(^2\)Another common approach to the surrogate is the Gaussian 
process (Rasmussen and Williams, 2006; Hoffman et al., 2011; Snoek et al., 2012). Like Bergstra et al. (2011), our 
preliminary experiments found the TPE to perform favorably. 
Further TPE’s tree-structured configuration space is advanta-
geous, because it allows nested definitions of hyperparameters, 
which we exploit in our experiments (e.g., only allows bigrams 
to be chosen if unigrams are also chosen).

---

**Algorithm 1 SMBO algorithm**

Input: number of trials $T$, target function $f$

$p_1$ = initial surrogate model

Initialize $y^*$

for $t = 1$ to $T$ do

$x_t \leftarrow \arg\max_x A(x; p_t, y^*)$

$y_t \leftarrow$ evaluate $f(x_t)$

Update $y^*$

Estimate $p_t$ given $x_t$ and $y_t$

end for
Table 1: The set of hyperparameters considered in our experiments. The top half are hyperparameters related to text representation, while the bottom half are logistic regression hyperparameters, which also interact with the chosen representation.

<table>
<thead>
<tr>
<th>Hyperparameter</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>( n_{\text{max}} )</td>
<td>([1, 2, 3])</td>
</tr>
<tr>
<td>weighting scheme</td>
<td>([n_{\text{min}}, \ldots, 3])</td>
</tr>
<tr>
<td>remove stop words?</td>
<td>([\text{tf}, \text{tf-idf}, \text{binary}])</td>
</tr>
<tr>
<td>regularization</td>
<td>([\ell_1, \ell_2])</td>
</tr>
<tr>
<td>regularization strength</td>
<td>([10^{-5}, 10^0])</td>
</tr>
<tr>
<td>convergence tolerance</td>
<td>([10^{-5}, 10^{-3}])</td>
</tr>
</tbody>
</table>

\[ A(x; p_t, y^*) \propto \left( \gamma + \frac{p_{\gamma}^T(x)}{p_t^T(x)} (1 - \gamma) \right)^{-1}, \]

where \( \gamma = p_t(y < y^*) \), fixed at 0.15 by definition of \( y^* \) (above). Here, we prefer \( x \) with high probability under \( p_{\gamma}^T(x) \) and low probability under \( p_t^T(x) \). To maximize this quantity, we draw many candidates according to \( p_{\gamma}^T(x) \) and evaluate them according to \( p_{\gamma}^T(x)/p_t^T(x) \). Note that \( p(y) \) does not need to be given an explicit form. To compute \( p_{\gamma}^T(x) \) and \( p_t^T(x) \), we associate each hyperparameter with a node in the graphical model and multiply individual probabilities at every node—see Bergstra et al. (2011) for details.

4 Experiments

We fix \( L \) to logistic regression. We optimize text representation based on the types of \( n \)-grams used, the type of weighting scheme, and the removal of stopwords; we also optimize the regularizer and training convergence criterion, which interact with the representation. See Table 1 for a complete list.

Note that even with this limited number of options, the number of possible combinations is huge so exhaustive search is computationally expensive. In all our experiments for all datasets, we limit ourselves to 30 trials per dataset. The only preprocessing we applied was downcasing.

We always use a development set to evaluate \( f(x) \) during learning and report the final result on an unseen test set. We summarize the hyperparameters selected by our method, and the accuracies achieved (on test data) in Table 1. We discuss comparisons to baselines for each dataset in turn. For each of our datasets, we select supervised, non-ensemble classification methods from previous literature as baselines. In each case, we emphasize comparisons with the best-published linear method (often an SVM with a linear kernel with representation selected by experts) and the best-published method overall. In the following, “SVM” always means “linear SVM.” All methods were trained and evaluated on the same training/testing splits as baselines; in cases where standard development sets were not available, we used a random 20% of the training data as a development set.

Stanford sentiment treebank (Socher et al., 2013)—Table 2 A sentence-level sentiment analysis dataset ofrottentomatoes.com movie reviews: http://nlp.stanford.edu/sentiment. We use the binary classification task where the goal is to predict whether a review is positive or negative (no neutral). Our logistic regression model outperforms the baseline SVM reported by Socher et al. (2013), who used only unigrams but did not specify the weighting scheme for their SVM baseline. While our result is still below the state-of-the-art based on the the recursive neural tensor networks (Socher et al., 2013) and the paragraph vector (Le and Mikolov, 2014), we show that logistic regression is comparable with recursive and matrix-vector neural networks (Socher et al., 2011; Socher et al., 2012).

<table>
<thead>
<tr>
<th>Method</th>
<th>Acc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Naive Bayes</td>
<td>81.8</td>
</tr>
<tr>
<td>SVM</td>
<td>79.4</td>
</tr>
<tr>
<td>Vector average</td>
<td>80.1</td>
</tr>
<tr>
<td>Recursive neural networks</td>
<td>82.4</td>
</tr>
<tr>
<td>LR (this work)</td>
<td>82.4</td>
</tr>
<tr>
<td>Matrix-vector RNN</td>
<td>82.9</td>
</tr>
<tr>
<td>Recursive neural tensor networks</td>
<td>85.4</td>
</tr>
<tr>
<td>Paragraph vector</td>
<td>87.8</td>
</tr>
</tbody>
</table>

Table 2: Comparisons on the Stanford sentiment treebank dataset. Scores are as reported by Socher et al. (2013) and Le and Mikolov (2014). Test size = 6,920.

Amazon electronics (McAuley and Leskovec, 2013)—Table 3 A binary sentiment analysis dataset of Amazon electronics product reviews: http://riejohnson.com/cnn_data.html. The best-performing methods on this dataset are based on convolutional neural networks (Johnson and Zhang, 2015). Our method is on par with the second-best of these, outperforming all of the reported feed-forward neural networks and SVM variants Johnson and Zhang used as baselines. They varied

\[4\] These are convolutional neural networks with a rectifier activation function, trained under \( \ell_2 \) regularization with stochastic gradient descent. The authors also consider an extension based on parallel CNN that we do not include here.
the representations, and used log term frequency and normalization to unit vectors as the weighting scheme, after finding that this outperformed term frequency. Our method achieved the best performance with binary weighting, which they did not consider.

**IMDB movie reviews (Maas et al., 2011)**—Table 3. A binary sentiment analysis dataset of highly polar IMDB movie reviews: http://ai.stanford.edu/~amaas/data/sentiment. The results parallel those for Amazon electronics; our method comes close to convolutional neural networks (Johnson and Zhang, 2015), which are state-of-the-art. It outperforms SVMs and feed-forward neural networks, the restricted Boltzmann machine approach presented by Dahl et al. (2012), and compressive feature learning (Paskov et al., 2013).

<table>
<thead>
<tr>
<th>Method</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Amazon</td>
</tr>
<tr>
<td>SVM-unigrams</td>
<td>88.29</td>
</tr>
<tr>
<td>RBM</td>
<td>89.23</td>
</tr>
<tr>
<td>SVM-{1, 2}-grams</td>
<td>90.95</td>
</tr>
<tr>
<td>Compressive feature learning</td>
<td>90.40</td>
</tr>
<tr>
<td>SVM-{1, 2, 3}-grams</td>
<td>91.29</td>
</tr>
<tr>
<td>LR-{1, 2, 3, 4, 5}-grams</td>
<td>91.52</td>
</tr>
<tr>
<td>NN-{1, 2, 3}-grams</td>
<td>91.52</td>
</tr>
<tr>
<td>LR (this work)</td>
<td>91.56</td>
</tr>
<tr>
<td>Bag of words CNN</td>
<td>91.61</td>
</tr>
<tr>
<td>Sequential CNN</td>
<td>92.52</td>
</tr>
</tbody>
</table>

Table 3: Comparisons on the Amazon electronics and IMDB reviews datasets. SVM results are from Wang and Manning (2012), the RBM (restricted Boltzmann machine) result is from Dahl et al. (2012), NN and CNN results are from Johnson and Zhang (2015), and LR-{1, 2, 3, 4, 5}-grams and compressive feature learning results are from Paskov et al. (2013). Test size = 20,000 for both datasets.

**Congressional vote (Thomas et al., 2006)**—Table 4. A dataset of transcripts from the U.S. Congressional debates: http://www.cs.cornell.edu/~ainur/sle-data.html. Similar to previous work (Thomas et al., 2006; Bansal et al., 2008; Yessenalina et al., 2010), we consider the task to predict the vote (“yea” or “nay”) for the speaker of each speech segment (speaker-based speech-segment classification). Our method outperforms the best results of Yessenalina et al. (2010), which use a multi-level structured model based on a latent-variable SVM. We show comparisons to two weaker baselines as well.

<table>
<thead>
<tr>
<th>Method</th>
<th>Acc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM-link</td>
<td>71.28</td>
</tr>
<tr>
<td>Min-cut</td>
<td>75.00</td>
</tr>
<tr>
<td>SVM-SLE</td>
<td>77.67</td>
</tr>
<tr>
<td>LR (this work)</td>
<td>78.59</td>
</tr>
</tbody>
</table>

Table 4: Comparisons on the congress vote dataset. SVM-link exploits link structures (Thomas et al., 2006); the min-cut result is from Bansal et al. (2008), and SVM-SLE result is reported by Yessenalina et al. (2010). Test size = 1,175.

**20 Newsgroups (Lang, 1995) all topics**—Table 6. 20 Newsgroups is a benchmark topic classification dataset: http://qwone.com/~jason/20Newsgroups. There are 20 topics in this dataset. Our method outperforms state-of-the-art methods including the distributed structured output model (Srikumar and Manning, 2014). The strong logistic regression baseline from Paskov et al. (2013) uses all 5-grams, heuristic normalization, and elastic net regularization; our method found that unigrams and bigrams, with binary weighting and $l_2$ penalty, achieved far better results.

<table>
<thead>
<tr>
<th>Method</th>
<th>Acc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discriminative RBM</td>
<td>76.20</td>
</tr>
<tr>
<td>LR-{1, 2, 3, 4, 5}-grams</td>
<td>82.80</td>
</tr>
<tr>
<td>Compressive feature learning</td>
<td>83.00</td>
</tr>
<tr>
<td>Distributed structured output</td>
<td>84.00</td>
</tr>
<tr>
<td>LR (this work)</td>
<td>87.84</td>
</tr>
</tbody>
</table>

Table 6: Comparisons on the 20 Newsgroups dataset for classifying documents into all topics. The discriminative RBM result is from Larochelle and Bengio (2008); compressive feature learning and LR-5-grams results are from Paskov et al. (2013), and the distributed structured output result is from Srikumar and Manning (2014). Test size = 9,052.

20 Newsgroups: talk.religion.misc vs. alt.atheism and comp.graphics vs. comp.windows.x. We derived three additional topic classification tasks from the 20N dataset. The first and second tasks are talk.religion.misc vs. alt.atheism (test size = 686) and comp.graphics vs. comp.windows.x (test size = 942). Wang and Manning (2012) report a bigram naive Bayes model achieving 85.1% and 91.2% on these tasks, respectively (best single model results). Our methods were designed for structured prediction, but Srikumar and Manning (2014) also applied it to classification. It attempts to learn a distributed representation for features and for labels. The authors used unigrams and did not discuss the weighting scheme.

---

5As noted, semi-supervised and ensemble methods are excluded for a fair comparison.

6This approach is based on minimum description length, using unlabeled data to select a set of higher-order $n$-grams to use as features.
method achieves 86.3% and 92.1% using slightly different representations (see Table 5). The last task is to classify related science documents into four science topics (sci.crypt, sci.electronics, sci.space, sci.med; test size = 1,899). We were not able to find previous results that are comparable to ours on this task; we include our result (95.82%) to enable further comparisons in the future.

5 Discussion

**Optimized representations.** For each task, the chosen representation is different. Out of all possible choices in our experiments (Table 1), each of them is used by at least one of the datasets (Table 5). For example, on the Congress vote dataset, we only need to use bigrams, whereas on the Amazon electronics dataset we need to use \{1, 2, 3\}-grams. The binary weighting scheme works well for most of the datasets, except the sentence-level sentiment analysis task, where the tf-idf weighting scheme was selected. \(\ell_2\) regularization was best in all cases but one. We do not believe that an NLP expert would be likely to make these particular choices, except through the same kind of trial-and-error process our method automates efficiently.

**Number of trials.** We ran 30 trials for each dataset in our experiments. Figure 1 shows each trial accuracy and the best accuracy on development data as we increase the number of trials for two datasets. We can see that 30 trials are generally enough for the model to obtain good results, although the search space is large.

**Transfer learning and multitask setting.** We treat each dataset independently and create a separate model for each of them. It is also possible to learn from previous datasets (i.e., transfer learning) or to learn from all datasets simultaneously (i.e., multitask learning) to improve performance. This has the potential to reduce the number of trials required even further. See Bardenet et al. (2013), Swersky et al. (2013), and Yogatama and Mann (2014) for more about how to perform Bayesian optimization in these settings.

**Beyond supervised learning.** Our framework could also be extended to unsupervised and semi-supervised models. For example, in document clustering (e.g., \(k\)-means), we also need to construct representations for documents. Log-likelihood might serve as a performance function. A range of random initializations might be considered. Investigation of this approach for nonconvex problems is an exciting area for future work.

6 Conclusion

We used Bayesian optimization to optimize choices about text representations for various categorization problems. Our technique identifies settings for a standard linear model (logistic regression) that are competitive with far more sophisticated methods on topic classification and sentiment analysis.

Acknowledgments

We thank several reviewers for their helpful feedback. This work was supported by the Defense Advanced Research Projects Agency through grant FA87501420244 and computing resources provided by Amazon. This research was completed while NAS was at CMU.
References


Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In Proc. of ICML.


Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In Proc. of ACL.


Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Chris Manning, Andrew Ng, and Chris Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In Proc. of EMNLP.

Vivek Srikumar and Christopher D. Manning. 2014. Learning distributed representations for structured output prediction. In NIPS.


Matt Thomas, Bo Pang, and Lillian Lee. 2006. Get out the vote: Determining support or opposition from congressional floor-debate transcripts. In Proc. of EMNLP.


