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 $\boldsymbol{d}_{train} = \langle\langle d.i_1, d.o_1\rangle, \ldots, \langle d.i_n, d.o_n\rangle\rangle$, 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, $\boldsymbol{d}_{dev} \in (\mathcal{I} \times \mathcal{O})^{n'}$.

Classification proceeds in three steps: first, $\mathbf{x} : \mathcal{I} \rightarrow \mathbb{R}^N$ 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 $L(\boldsymbol{d}_{train}) \circ \mathbf{x}$ (i.e., the composition of the representation function with

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2100–2105,
Lisbon, Portugal, 17-21 September 2015.

the learned mapping).

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

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

Choosing $\mathbf{x}$ 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 ($\mathbf{w}$) is costly.

3 Bayesian Optimization

Our approach is based on sequential model-based optimization (SMBO; Hutter et al., 2011). It iteratively chooses representation functions $\mathbf{x}$. 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 $\mathbf{x}$ with exploitation of previously-explored options, so that a good choice is found in a small number of trials.

More concretely, in the tth trial, $\mathbf{x}_t$ is selected using an acquisition function $\mathcal{A}$ and a "surrogate" probabilistic model p_t. Second, f is evaluated given $\mathbf{x}_t$—an expensive operation which involves training to learn parameters $\mathbf{w}$ and assessing performance on the held-out data. Third, the surrogate model is updated. See Algorithm 1; details on $\mathcal{A}$ and p_t follow.

Acquisition Function. A good acquisition function returns high values for $\mathbf{x}$ when either the value $f(\mathbf{x})$ is predicted to be high, or the uncertainty about $f(\mathbf{x})$'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),[1] which is the expectation (under the current surrogate model p_t) that $f(\mathbf{x}) = y$ will exceed $f(\mathbf{x}^*) = y^*$:

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

where $\mathbf{x}^*$ 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 (Villemonteix et al., 2009), Gaussian process upper confidence bound (Srinivas et al., 2010), or a combination of them (Hoffman et al., 2011).

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**
  $\mathbf{x}_t \leftarrow \arg\max_{\mathbf{x}} \mathcal{A}(\mathbf{x}; p_t, y^*)$
  $y_t \leftarrow$ evaluate $f(\mathbf{x}_t)$
  Update y^*
  Estimate p_t given $\mathbf{x}_{1:t}$ and $y_{1:t}$
end for

Surrogate Model. As a surrogate model, we use a tree-structured Parzen estimator (TPE; Bergstra et al., 2011).[2] This is a nonparametric approach to density estimation. We seek to estimate $p_t(y \mid \mathbf{x})$ where $y = f(\mathbf{x})$, the performance function that is expensive to compute exactly. The TPE approach seeks $p_t(y \mid \mathbf{x}) \propto p_t(y) \cdot \begin{cases} p_t^<(\mathbf{x}), & \text{if } y < y^* \\ p_t^\geq(\mathbf{x}), & \text{if } y \geq y^* \end{cases}$, where $p_t^<$ and $p_t^\geq$ 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 advantageous, 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).

Hyperparameter	Values
n_{min}	$\{1, 2, 3\}$
n_{max}	$\{n_{min}, \ldots, 3\}$
weighting scheme	{tf, tf-idf, binary}
remove stop words?	{True, False}
regularization	$\{\ell_1, \ell_2\}$
regularization strength	$[10^{-5}, 10^{5}]$
convergence tolerance	$[10^{-5}, 10^{-3}]$

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.

$\mathcal{A}(\mathbf{x}; p_t, y^*) \propto \left(\gamma + \frac{p_t^<(\mathbf{x})}{p_t^>(\mathbf{x})}(1-\gamma)\right)^{-1}$, where $\gamma = p_t(y < y^*)$, fixed at 0.15 by definition of y^* (above). Here, we prefer $\mathbf{x}$ with high probability under $p_t^>(\mathbf{x})$ and low probability under $p_t^<(\mathbf{x})$. To maximize this quantity, we draw many candidates according to $p_t^>(\mathbf{x})$ and evaluate them according to $p_t^<(\mathbf{x})/p_t^>(\mathbf{x})$. Note that $p(y)$ does not need to be given an explicit form. To compute $p_t^<(\mathbf{x})$ and $p_t^>(\mathbf{x})$, 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,[3] 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 $f(\mathbf{x})$ 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 5. 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 of rottentomatoes.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).

Method	Acc.
Naïve Bayes	81.8
SVM	79.4
Vector average	80.1
Recursive neural networks	82.4
LR (this work)	82.4
Matrix-vector RNN	82.9
Recursive neural tensor networks	85.4
Paragraph vector	87.8

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).[4] 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

[3] It is actually infinite since the reg. strength and conv. tolerance are continuous values, but we could discretize them.

[4] These are convolutional neural networks with a rectifier activation function, trained under ℓ_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.[5] 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).[6]

Method	Accuracy	
	Amazon	IMDB
SVM-unigrams	88.29	88.64
RBM		89.23
SVM-$\{1,2\}$-grams	90.95	90.26
Compressive feature learning		90.40
SVM-$\{1,2,3\}$-grams	91.29	90.58
LR-$\{1,2,3,4,5\}$-grams		90.60
NN-$\{1,2,3\}$-grams	91.52	90.83
LR (this work)	91.56	90.85
Bag of words CNN	91.61	91.34
Sequential CNN	92.52	91.61

Table 3: Comparisons on the Amazon electronics and IMDB reviews datasets. SVM results are from Wang and Manning (2012), the RBM (restricted Bolzmann 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.

Method	Acc.
SVM-link	71.28
Min-cut	75.00
SVM-SLE	77.67
LR (this work)	78.59

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).[7] 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.

Method	Acc.
Discriminative RBM	76.20
LR-$\{1,2,3,4,5\}$-grams	82.80
Compressive feature learning	83.00
Distributed structured output	84.00
LR (this work)	87.84

Table 6: Comparisons on the 20 Newsgroups dataset for classifying documents into all topics. The disriminative 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 naïve Bayes model achieving 85.1% and 91.2% on these tasks, respectively (best single model results).[8] Our

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

[6] This approach is based on minimum description length, using unlabeled data to select a set of higher-order n-grams to use as features.

[7] This method was 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.

[8] They also report a naïve Bayes/SVM ensemble achieving 87.9% and 91.2%.

Dataset	Acc.	n_{min}	n_{max}	Weighting	Stopword removal?	Reg.	Strength	Conv.
Stanford sentiment	82.43	1	2	tf-idf	F	ℓ_2	10	0.098
Amazon electronics	91.56	1	3	binary	F	ℓ_2	120	0.022
IMDB reviews	90.85	1	2	binary	F	ℓ_2	147	0.019
Congress vote	78.59	2	2	binary	F	ℓ_2	121	0.012
20N all topics	87.84	1	2	binary	F	ℓ_2	16	0.008
20N all science	95.82	1	2	binary	F	ℓ_2	142	0.007
20N atheist/religion	86.32	1	2	binary	T	ℓ_1	41	0.011
20N x/graphics	92.09	1	1	binary	T	ℓ_2	91	0.014

Table 5: Classification accuracies and the best hyperparameters for each of the datasets in our experiments. "Acc" shows accuracies for our logistic regression model. "Min" and "Max" correspond to the min n-grams and max n-grams respectively. "Reg." is the regularization type, "Strength" is the regularization strength, and "Conv." is the convergence tolerance. For regularization strength, we round it to the nearest integer for readability.

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 datsets (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. ℓ_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.

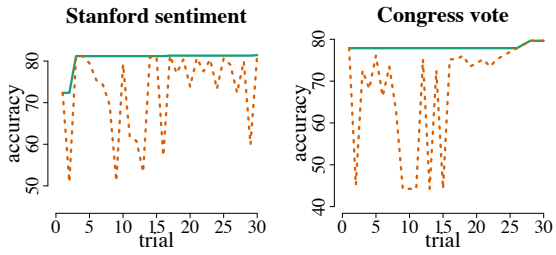

Figure 1: Classification accuracies on development data for Stanford sentiment treebank (left) and congressional vote (right) datasets. In each plot, the green solid line indicates the best accuracy found so far, while the dotted orange line shows accuracy at each trial. We can see that in general the model is able to obtain reasonably good representation in 30 trials.

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

Mohit Bansal, Clair Cardie, and Lillian Lee. 2008. The power of negative thinking: Exploiting label disagreement in the min-cut classification framework. In *Proc. of COLING*.

Remi Bardenet, Matyas Brendel, Balazs Kegl, and Michele Sebag. 2013. Collaborative hyperparameter tuning. In *Proc. of ICML*.

James Bergstra, Remi Bardenet, Yoshua Bengio, and Balazs Kegl. 2011. Algorithms for hyper-parameter optimization. In *NIPS*.

James Bergstra, Daniel Yamins, and David Cox. 2013. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *Proc. of ICML*.

Walter Daelemans, Veronique Hoste, Fien De Meulder, and Bart Naudts. 2003. Combined optimization of feature selection and algorithm parameters in machine learning of language. In *Proc. of ECML*.

George E. Dahl, Ryan P. Adams, and Hugo Larochelle. 2012. Training restricted Boltzmann machines on word observations. In *Proc. of ICML*.

Matthew Hoffman, Eric Brochu, and Nando de Freitas. 2011. Portfolio allocation for Bayesian optimization. In *Proc. of UAI*.

Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. 2011. Sequential model-based optimization for general algorithm configuration. In *Proc. of LION*.

Rie Johnson and Tong Zhang. 2015. Effective use of word order for text categorization with convolutional neural networks. In *Proc. of NAACL*.

Donald R. Jones. 2001. A taxonomy of global optimization methods based on response surfaces. *Journal of Global Optimization*, 21:345–385.

Ken Lang. 1995. Newsweeder: Learning to filter netnews. In *Proc. of ICML*.

Hugo Larochelle and Yoshua Bengio. 2008. Classification using discriminative restricted Boltzmann machines. In *Proc. of ICML*.

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

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225.

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*.

Julian McAuley and Jure Leskovec. 2013. Hidden factors and hidden topics: understanding rating dimensions with review text. In *Proc. of RecSys*.

Hristo S. Paskov, Robert West, John C. Mitchell, and Trevor J. Hastie. 2013. Compressive feature learning. In *Proc of NIPS*.

Carl Edward Rasmussen and Christopher K. I. Williams. 2006. *Gaussian Processes for Machine Learning*. The MIT Press.

Jasper Snoek, Hugo Larrochelle, and Ryan P. Adams. 2012. Practical Bayesian optimization of machine learning algorithms. In *NIPS*.

Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proc. of EMNLP*.

Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proc. of EMNLP*.

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*.

Niranjan Srinivas, Andreas Krause, Sham Kakade, and Matthias Seeger. 2010. Gaussian process optimization in the bandit setting: No regret and experimental design. In *Proc. of ICML*.

Kevin Swersky, Jasper Snoek, and Ryan P. Adams. 2013. Multi-task Bayesian optimization. In *NIPS*.

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

Julien Villemonteix, Emmanuel Vazquez, and Eric Walter. 2009. An informational approach to the global optimization of expensive-to-evaluate functions. *Journal of Global Optimization*, 44(4):509–534.

Sida Wang and Christopher D. Manning. 2012. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proc. of ACL*.

Ainur Yessenalina, Yisong Yue, and Claire Cardie. 2010. Multi-level structured models for document sentiment classification. In *Proc. of EMNLP*.

Dani Yogatama and Gideon Mann. 2014. Efficient transfer learning method for automatic hyperparameter tuning. In *Proc. of AISTATS*.