Active Discriminative Word Embedding Learning

Ye Zhang
University of Texas at Austin
Dept. of Computer Science
yezhang@utexas.edu

Byron Wallace
University of Texas at Austin
School of Information
byron.wallace@utexas.edu

Abstract

We propose a new active learning (AL) method for text classification based on convolutional neural networks (CNNs). In AL, one selects the instances to be manually labeled with the aim of maximizing model performance with minimal effort. Neural models capitalize on word embeddings as features, tuning these to the task at hand. We argue that AL strategies for neural text classification should focus on selecting instances that most affect the embedding space (i.e., induce discriminative word representations). This is in contrast to traditional AL approaches (e.g., uncertainty sampling), which specify higher level objectives. We propose a simple approach that selects instances containing words whose embeddings are likely to be updated with the greatest magnitude, thereby rapidly learning discriminative, task-specific embeddings. Empirical results show that our method outperforms baseline AL approaches.

1 Introduction

In active learning (AL), the machine learning algorithm being trained is allowed to select the examples to be manually annotated (and in turn learned from) (Settles, 2010). The idea is that by selecting training data cleverly, rather than at i.i.d. random, better models can be learned at lower cost. This approach is particularly attractive in scenarios in which labels are expensive but unlabeled data is plentiful.

There has been a wealth of work on AL approaches for traditional machine learning methods generally (Settles, 2010), and for text classification in particular (Tong and Koller, 2002; McCallum and Nigam, 1998; Wallace et al., 2010). But practically no work has considered AL for text classification using modern neural models. We argue that the representation learning facet of these models motivates different AL approaches.

Here we propose a method for AL using convolutional neural networks (CNNs), which have recently achieved strong performance across a diverse set of text classification tasks (Kim, 2014; Zhang and Wallace, 2015; Johnson and Zhang, 2014; Zhang et al., 2016). It is natural to consider how to make the best use of CNNs when annotation resources are scarce; that is the question we address here. Word embedding estimation and tuning (for a specific text classification task) may be viewed as feature learning. It is reasonable to optimize feature vectors before expending effort to tune the parameters of a model that accepts these as input (note that adjusting the former will render updates to the latter potentially useless). Thus the objective in AL should be to select instances that result in better features.

More specifically, we propose a novel, simple AL approach in which we select instances that contain words likely to most affect the embeddings. We achieve this by calculating the expected gradient length (EGL) with respect to the embeddings for each word comprising the respective unlabeled instances. We show that this approach allows us to rapidly learn discriminative embeddings. For example, in the task of sentiment classification, selecting examples in this way quickly pushes the embeddings of ‘bad’ and ‘good’ apart. We show that this in turn improves performance over several baseline AL approaches.
2 Preliminaries

2.1 CNNs for Text Classification

Here we provide a brief review of CNNs for text classification, using the model proposed by Kim (2014). We will denote the word embedding matrix by \( \mathbf{E} \in \mathbb{R}^{V \times d} \), where \( V \) is the vocabulary size, and \( d \) is the dimension of the embedding layer. A specific instance (piece of text to be categorized) is then represented by stacking the vectors corresponding to the words it contains (stored in \( \mathbf{E} \)), preserving word order. This results in an instance matrix \( \mathbf{A} \in \mathbb{R}^{S \times d} \), where \( S \) is the text length.

We then apply convolution operations on this matrix, using multiple linear filters. Each filter matrix \( \mathbf{W}_i \in \mathbb{R}^{h \times d} \) performs a convolution operation on \( \mathbf{A} \), generating a feature map \( \mathbf{c}_i \in \mathbb{R}^{S-h+1} \). We then apply 1-max pooling to each \( \mathbf{c}_i \), obtaining a feature value \( o_i \) for this filter. (We note that we use multiple filter heights and redundant filters of each height.) Finally, we concatenate all \( o_i \) to compose a final feature vector \( o \) for each instance. This is run through a softmax layer to induce a probability distribution over the output space. Typically this model is trained by minimizing the cross-entropy loss via back-propagation (Rumelhart et al., 1988). We provide a schematic illustrating a toy realization of this model in Figure 1. For more details, we refer the reader to (Kim, 2014; Zhang and Wallace, 2015).

2.2 Active Learning

We consider the pool-based AL scenario, in which it is assumed that there is a small set of labeled data \( \mathcal{L} \) and a large pool of available unlabeled data \( \mathcal{U} \). The task for the learner is to draw examples selectively from this pool to be labeled. These selections, or queries, are typically made in a greedy fashion; an informativeness measure is used to evaluate all instances in the pool, and the instance maximizing this measure is selected.

The key to developing AL strategies is designing a good informativeness measure. Let \( x^* \) be the most informative instance according to a query strategy \( \phi(x_i; \theta) \), which is a function used to evaluate each instance \( x_i \) in the unlabeled pool \( \mathcal{U} \) conditioned on the current set of parameter estimates \( \theta \). Then

\[
x^* = \arg\max_{x_i \in \mathcal{U}} \phi(x_i; \theta)
\]

In the case of CNNs, \( \theta \) includes word embedding parameters \( \mathbf{E} \), convolution layer parameters \( \mathbf{C} \), and softmax layer parameters \( \mathbf{W} \).

Many querying strategies have been proposed in the literature (Settles, 2010). In this work we consider three general strategies.

Random sampling. This strategy is equivalent to standard (or ‘passive’) learning; here the training data is simply an i.i.d. sample from \( \mathcal{U} \).

Uncertainty sampling. One of the simplest and perhaps the most commonly used query framework is uncertainty sampling (Lewis and Gale, 1994; Long and Koller, 2002; Zhu et al., 2008). In this approach, the learner requests labels for instances about which it is least certain, with respect to categorization. A general uncertainty sampling variant uses entropy (Shannon, 2001) as an uncertainty measure, defining \( \phi(x_i; \theta) \) as:

\[
- \sum_k P(y_i = k|x_i; \theta) \log P(y_i = k|x_i; \theta)
\]

where \( k \) ranges over all possible labels.

Entropy-based uncertainty sampling has been observed to achieve strong empirical performance across many tasks (Settles, 2010).

Expected Gradient Length (EGL). This general AL strategy aims to select instances expected to result in the greatest change to the current model parameter estimates when their labels are revealed (or provided) (Settles and Craven, 2008). If the true class for a given instance were known, the gradient could be directly calculated under this assignment. But in practice this is unknown, and so the expectation is taken by marginalizing over the gradients calculated under possible class assignments, scaled by current model estimates of the posterior probabilities of said assignments.
We now introduce our proposed AL strategy for text classification with embeddings. This is based on the EGL method described above. In past work on EGL – which involved linear models – the expected change to model parameters was evaluated over the entire set of parameters $\theta$. By contrast, we propose explicitly selecting examples that are likely to affect the feature-level parameters (i.e., the embeddings).

In gradient-based optimization, the training gradient back-propagated to a set of model parameters given label $y_i$ for instance $x_i$ may be viewed as a measure of change imparted by example $i$. Thus the learner should request the label for an instance expected to produce a large magnitude training gradient. If this gradient is taken w.r.t. all model parameters, then this is a straight-forward instantiation of EGL. Formally, let $\nabla J(L; \theta)$ be the gradient of the objective function $J$ w.r.t. the model parameters $\theta$, where $J$ is the cost function. Further, let $\nabla J(L \cup \langle x_i, y_i \rangle; \theta)$ be the new gradient that would be obtained by adding the training tuple $(x_i, y_i)$ to $L$. Because the true label $y_i$ will be unknown, we take an expectation over possible class assignments $k$. More precisely, we can calculate $\phi(x_i; \theta)$ as:

$$
\sum_k P(y_i = k|x_i; \theta) \| \nabla J(L \cup \langle x_i, y_i = k \rangle; \theta) \|
$$

(3)

where $\| \mathbf{r} \|$ denotes the Euclidean norm of $\mathbf{r}$. Note that at query time $\| \nabla J(L; \theta) \|$ should be near zero, assuming $J$ converged during the previous iteration. Thus, we can approximate $\nabla J(L \cup \langle x_i, y_i = k \rangle; \theta)$ by $\nabla J(\langle x_i, y_i = k \rangle; \theta)$ for efficiency.

This approach selects instances that are likely to most perturb the model parameters. Thus far we have treated all parameters $\theta$ as being equally important. However, ‘deep’ neural architectures are notable for their hierarchical structure, which corresponds to a large set of features distributed across different layers in the architecture; this makes calculating the EGL computationally expensive. More importantly, it is arguably incoherent to jointly consider the expected change at different layers in the model. If we view lower levels in the model as learning features, it makes little sense to jointly maximize expected change in these features and the parameters of the final softmax layer that accepts these as input. Changes to the former will immediately change the implications of perturbing the latter.

Instead, we want to select unlabeled instances that can most improve the features learned by the model. Intuitively, it is paramount that the model learn good (discriminative) representations; these will feed forward through the network, in turn improving classification. Thus for each text instance in $\mathcal{U}$, we take the expected gradient with respect to only the embeddings of its constituent words, and we select the example that maximizes this max-expected embedding gradient as our measure of informativeness. Intuitively, we use a max-over-words approach because we aim to adjust particular word embeddings that are discriminative for the task at hand. Formally, we define our $\phi(x_i; \theta)$ as:

$$
\max_{j \in x_i} \sum_k P(y_i = k|x_i; \theta) \| \nabla_j \phi(\langle x_i, y_i = k \rangle; \theta) \|
$$

(4)

Where we denote by $\nabla_j \phi$ the gradient of $J$ w.r.t. only the embedding of word $j$ ($j$ ranges over the words in $x_i$). Note that the gradient is only taken for each word in the instance $x_i$; the gradients on embeddings corresponding to words not in instance are 0 and can thus be ignored, which is a computational boon. We refer to this strategy as EGL-word.

EGL-word focuses on parameters associated with the lowest level in the model (Figure 1). We also consider the other extreme: taking the gradient w.r.t. only the final softmax layer parameters $W$. In this case $\phi(x_i; \theta)$ becomes:

$$
\sum_k P(y_i = k|x_i; \theta) \| \nabla_j \phi(\langle x_i, y_i = k \rangle; \theta) \|
$$

(5)

where $J_W$ denotes the gradient w.r.t. the softmax layer. We refer to this strategy as EGL-sm.

4 Experimental Setup and Results

4.1 Datasets

**CR** Customer reviews of products categorized as ‘positive’ or ‘negative’ [Hu and Liu, 2004]\(^1\)

**MR** Movie reviews labeled as ‘positive’ or ‘negative’ (Pang and Lee, 2005).

**Subj** Sentences deemed ‘subjective’ or ‘objective’ (Pang and Lee, 2004)\(^2\)

\(^1\)http://www.cs.uic.edu/liub/FBS/sentiment-analysis.html

\(^2\)Both the MR and Subj datasets are available at:
4.2 Model Configuration

For CNNs, we used pre-trained word2vec-induced vectors\(^3\) to initialize \(E\). We used three filter heights (3, 4, 5) and 50 filters for each. We did not tune these hyperparameters.

We performed 20 rounds of batch active learning. All learners began with the same 25 instances. In subsequent rounds, each learner selected 25 instances from \(U\) according to their respective querying strategies. These examples were added to \(L\), and the models were retrained. Performance was evaluated by calculating accuracy on a held-out test set after each round. We repeated the entire AL process 10 times via 10-fold CV, replicating the experiment 5 times per fold to account for variation. Reported results are means of these averages.

We used Adadelta (Zeiler, 2012) for parameter estimation, and we tuned \(E\) during back-propagation, thus inducing discriminative embeddings.

4.3 Results and Discussion

We show learning curves in Figure 2. Our proposed EGL-word active learning method outperforms all of the baselines. The method performs especially well on sentiment analysis tasks (MR and CR). We believe this is due to our model rapidly learning more discriminative representations of words with opposing polarities.

To further illustrate this point, we also provide plots displaying the Euclidean distances between selected pairs of word embeddings induced using different AL strategies (Figure 2). In the case of the customer review dataset, for example, we see that EGL-word quickly pushes the embeddings corresponding to ‘good’ and ‘bad’ apart. Similarly, on the movie review dataset, ‘fun’ and ‘boring’ are rapidly separated in embedding space. The subjectivity detection task is less clear-cut. Here we picked words ‘amusing’ and ‘their’, because ‘amusing’ strongly indicates subjectivity, while ‘their’ is plainly neutral. In the Figure 2, we observe that EGL-word separates these, as expected, though less rapidly than in the sentiment case.

5 Conclusions

We have proposed a new active learning strategy for CNNs that is specifically designed to quickly induce discriminative word embeddings, and thus improve text classification. We have shown that this approach outperforms baseline AL strategies on three text classification datasets, and that it rapidly finds discriminative embeddings for words.
References

[Hu and Liu2004] Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining, pages 168–177. ACM.

[Johnson and Zhang2014] Rie Johnson and Tong Zhang. 2014. Effective use of word order for text categorization with convolutional neural networks. arXiv preprint arXiv:1412.1058.

[Kim2014] Yoon Kim. 2014. Convolutional neural networks for sentence classification. arXiv preprint arXiv:1408.5882.

[Lewis and Gale1994] David D Lewis and William A Gale. 1994. A sequential algorithm for training text classifiers. In Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval, pages 3–12. Springer-Verlag New York, Inc.

[McCallumzy and Nigamy1998] Andrew Kachites McCallumzy and Kamal Nigamy. 1998. Employing em and pool-based active learning for text classification. In Proc. International Conference on Machine Learning (ICML), pages 359–367. Citeseer.

[Pang and Lee2004] Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In Proceedings of the 42nd annual meeting on Association for Computational Linguistics, page 271. Association for Computational Linguistics.

[Pang and Lee2005] Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics, pages 115–124. Association for Computational Linguistics.

[Rumelhart et al.1988] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1988. Learning representations by back-propagating errors. Cognitive modeling, 5(3):1.

[Settles and Craven2008] Burr Settles and Mark Craven. 2008. An analysis of active learning strategies for sequence labeling tasks. In Proceedings of the conference on empirical methods in natural language processing, pages 1070–1079. Association for Computational Linguistics.

[Settles2010] Burr Settles. 2010. Active learning literature survey. University of Wisconsin, Madison, 52(55-66):11.

[Shannon2001] Claude Elwood Shannon. 2001. A mathematical theory of communication. ACM SIGMOBILE Mobile Computing and Communications Review, 5(1):3–55.

[Tong and Koller2002] Simon Tong and Daphne Koller. 2002. Support vector machine active learning with applications to text classification. The Journal of Machine Learning Research, 2:45–66.

[Wallace et al.2010] Byron C Wallace, Kevin Small, Carla E Brodley, and Thomas A Trikalinos. 2010. Active learning for biomedical citation screening. In Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 173–182. ACM.

[Zeiler2012] Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. arXiv preprint arXiv:1212.5701.

[Zhang and Wallace2015] Ye Zhang and Byron Wallace. 2015. A sensitivity analysis of (and practitioners’ guide to) convolutional neural networks for sentence classification. arXiv preprint arXiv:1510.03820.

[Zhang et al.2016] Ye Zhang, Stephen Roller, and Byron Wallace. 2016. Mnc-cnn: A simple approach to exploiting multiple word embeddings for sentence classification. arXiv preprint arXiv:1603.00968.

[Zhu et al.2008] Jingbo Zhu, Huizhen Wang, Tianshun Yao, and Benjamin K Tsou. 2008. Active learning with sampling by uncertainty and density for word sense disambiguation and text classification. In Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1, pages 1137–1144. Association for Computational Linguistics.