Learning the Curriculum with Bayesian Optimization for Task-Specific Word Representation Learning

Yulia Tsvetkov[♠] **Manaal Faruqui**[♠] **Wang Ling**[♣] **Brian MacWhinney**[♠] **Chris Dyer**[♣♠]
[♠]Carnegie Mellon University [♣]Google DeepMind
{ytsvetko,mfaruqui,cdyer}@cs.cmu.edu,
lingwang@google.com, macw@cmu.edu

Abstract

We use Bayesian optimization to learn curricula for word representation learning, optimizing performance on downstream tasks that depend on the learned representations as features. The curricula are modeled by a linear ranking function which is the scalar product of a learned weight vector and an engineered feature vector that characterizes the different aspects of the complexity of each instance in the training corpus. We show that learning the curriculum improves performance on a variety of downstream tasks over random orders and in comparison to the natural corpus order.

1 Introduction

It is well established that in language acquisition, there are robust patterns in the order by which phenomena are acquired. For example, prototypical concepts are acquired earlier; concrete words tend to be learned before abstract ones (Rosch, 1978). The acquisition of lexical knowledge in artificial systems proceeds differently. In general, models will improve during the course of parameter learning, but the time course of acquisition is not generally studied beyond generalization error as a function of training time or data size. We revisit this issue of choosing the order of learning—**curriculum learning**—framing it as an optimization problem so that a rich array of factors—including nuanced measures of difficulty, as well as prototypicality and diversity—can be exploited.

Prior research focusing on curriculum strategies in NLP is scarce, and has conventionally been following a paradigm of "starting small" (Elman, 1993), i.e., initializing the learner with "simple" examples first, and then gradually increasing data complexity (Bengio et al., 2009; Spitkovsky et al., 2010). In language modeling, this preference for increasing complexity has been realized by curricula that increase the entropy of training data by growing the size of the training vocabulary from frequent to less frequent words (Bengio et al., 2009). In unsupervised grammar induction, an effective curriculum comes from increasing length of training sentences as training progresses (Spitkovsky et al., 2010). These case studies have demonstrated that carefully designed curricula can lead to better results. However, they have relied on heuristics in selecting curricula or have followed the intuitions of human and animal learning (Kail, 1990; Skinner, 1938). Had different heuristics been chosen, the results would have been different. In this paper, we use curriculum learning to create improved word representations. However, rather than testing a small number of curricula, we search for an optimal curriculum using Bayesian optimization. A curriculum is defined to be the ordering of the training instances, in our case it is the ordering of paragraphs in which the representation learning model reads the corpus. We use a linear ranking function to conduct a systematic exploration of interacting factors that affect curricula of representation learning models. We then analyze our findings, and compare them to human intuitions and learning principles.

We treat curriculum learning as an outer loop in the process of learning and evaluation of vector-space representations of words; the iterative procedure is (1) predict a curriculum; (2) train word embeddings; (3) evaluate the embeddings on tasks that use word embeddings as the sole features. Through this model we analyze the impact of curriculum on word representation models and on extrinsic tasks. To quantify curriculum properties, we define three groups of features aimed at analyzing statistical and linguistic content and structure

of training data: (1) diversity, (2) simplicity, and (3) prototypicality. A function of these features is computed to score each paragraph in the training data, and the curriculum is determined by sorting corpus paragraphs by the paragraph scores. We detail the model in §2. Word vectors are learned from the sorted corpus, and then evaluated on part-of-speech tagging, parsing, named entity recognition, and sentiment analysis (§3). Our experiments confirm that training data curriculum affects model performance, and that models with optimized curriculum consistently outperform baselines trained on shuffled corpora (§4). We analyze our findings in §5.

The contributions of this work are twofold. First, this is the first framework that formulates curriculum learning as an optimization problem, rather then shuffling data or relying on human intuitions. We experiment with optimizing the curriculum of word embeddings, but in principle the curriculum of other models can be optimized in a similar way. Second, to the best of our knowledge, this study is the first to analyze the impact of distributional and linguistic properties of training texts on the quality of task-specific word embeddings.

2 Curriculum Learning Model

We are considering the problem of maximizing a performance of an NLP task through sequentially optimizing the curriculum of training data of word vector representations that are used as features in the task.

Let $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$ be the training corpus with n lines (sentences or paragraphs). The curriculum of word representations is quantified by scoring each of the paragraphs according to the linear function $\mathbf{w}^\intercal \boldsymbol{\phi}(\mathcal{X})$, where $\boldsymbol{\phi}(\mathcal{X}) \in \mathbb{R}^{\ell \times 1}$ is a real-valued vector containing ℓ linguistic features extracted for each paragraph, and $\mathbf{w} \in \mathbb{R}^{\ell \times 1}$ denote the weights learned for these features. The feature values $\boldsymbol{\phi}(\mathcal{X})$ are z-normalized across all paragraphs. These scores are used to specify the order of the paragraphs in the corpus—the curriculum. We sort the paragraphs by their scores.

After the paragraphs are curriculum-ordered, the reordered training corpus is used to generate word representations. These word representations are then used in a subsequent NLP task. We define the objective function $eval : \mathcal{X} \rightarrow \mathbb{R}$, which is the quality estimation metric for this task performed on a held-out dataset (e.g., correlation, accuracy, F_1 score, BLEU). Our goal is to define the features $\boldsymbol{\phi}(\mathcal{X})$ and to find the optimal weights $\mathbf{w}$ that maximize $eval$.

We optimize the feature weights using Bayesian optimization; we detail the model in §2.1. Distributional and linguistic features inspired by prior research in language acquisition and second language learning are described in §2.2. Figure 1 shows the computation flow diagram.

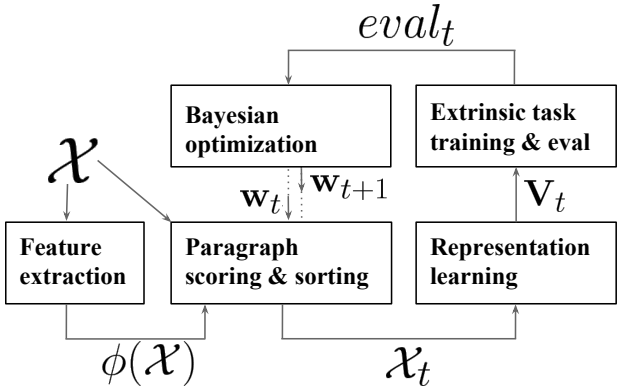

Figure 1: Curriculum optimization framework.

2.1 Bayesian Optimization for Curriculum Learning

As no assumptions are made regarding the form of $eval(\mathbf{w})$, gradient-based methods cannot be applied, and performing a grid search over parameterizations of $\mathbf{w}$ would require a exponentially growing number of parameterizations to be traversed. Thus, we propose to use Bayesian Optimization (BayesOpt) as the means to maximize $eval(\mathbf{w})$. BayesOpt is a methodology to globally optimize *expensive*, multimodal black-box functions (Shahriari et al., 2016; Bergstra et al., 2011; Snoek et al., 2012). It can be viewed as a sequential approach to performing a regression from high-level model parameters (e.g., learning rate, number of layers in a neural network, and in our model–curriculum weights $\mathbf{w}$) to the loss function or the performance measure ($eval$).

An arbitrary objective function, $eval$, is treated as a black-box, and BayesOpt uses Bayesian inference to characterize a posterior distribution over functions that approximate $eval$. This model of $eval$ is called the **surrogate model**. Then, the BayesOpt exploits this model to make decisions about $eval$, e.g., where is the expected maximum of the function, and what is the expected improvement that can be obtained over the best iteration so far. The strategy function, estimating the next set

of parameters to explore given the current beliefs about $eval$ is called the **acquisition function**. The surrogate model and the acquisition function are the two key components in the BayesOpt framework; their interaction is shown in Algorithm 1.

Popular choices for the surrogate model are Gaussian Processes (Rasmussen, 2006; Snoek et al., 2012, GP), providing convenient and powerful prior distribution on functions, and tree-structured Parzen estimators (Bergstra et al., 2011, TPE), tailored to handle conditional spaces.

The surrogate model allows us to cheaply approximate the quality of a set of parameters $\mathbf{w}$ without running $eval(\mathbf{w})$, and the acquisition function uses this surrogate to choose a new value of $\mathbf{w}$. However, a trade-off must be made: should the acquisition function move $\mathbf{w}$ into a region where the surrogate believes an optimal value will be found, or should it explore regions of the space that reveal more about how $eval$ behaves, perhaps discovering even better values? That is, acquisition functions balance a tradeoff between exploration—by selecting $\mathbf{w}$ in the regions where the uncertainty of the surrogate model is high, and exploitation—by querying the regions where the model prediction is high. Choices of the acquisition functions include probability of improvement (Kushner, 1964), expected improvement (EI) (Močkus et al., 1978; Jones, 2001), GP upper confidence bound (Srinivas et al., 2010), Thompson sampling (Thompson, 1933), entropy search (Hennig and Schuler, 2012), and dynamic combinations of the above functions (Hoffman et al., 2011); see Shahriari et al. (2016) for an extensive comparison. Yogatama et al. (2015) found that the combination of EI as the acquisition function and TPE as the surrogate model performed favorably in Bayesian optimization of text representations; we follow this choice in our model.

Algorithm 1 Bayesian optimization

$\mathcal{H} \leftarrow \emptyset$ ▷ Initialize observation history
$\mathcal{A} \leftarrow EI$ ▷ Initialize acquisition function
$\mathcal{S}_0 \leftarrow TPE$ ▷ Initialize surrogate model
for t $\leftarrow$ 1 to T **do**
  $\mathbf{w}_t \leftarrow \text{argmax}_{\mathbf{w}} \mathcal{A}(\mathbf{w}; \mathcal{S}_{t-1}, \mathcal{H})$ ▷ Predict $\mathbf{w}_t$ by optimizing acquisition function
  $eval(\mathbf{w}_t)$ ▷ Evaluate $\mathbf{w}_t$ on extrinsic task
  $\mathcal{H} \leftarrow \mathcal{H} \cup (\mathbf{w}_t, eval(\mathbf{w}_t))$ ▷ Update observation history
  Estimate $\mathcal{S}_t$ given $\mathcal{H}$
end for
return $\mathcal{H}$

2.2 Distributional and Linguistic Features

To characterize and quantify a curriculum, we define three categories of features, focusing on various distributional, syntactic, and semantic aspects of training data. We now detail the feature categories along with motivations for feature selection.

DIVERSITY. Diversity measures capture the distributions of types in data. Entropy is the best-known measure of diversity in statistical research, but there are many others (Tang et al., 2006; Gimpel et al., 2013). Common measures of diversity are used in many contrasting fields, from ecology and biology (Rosenzweig, 1995; Magurran, 2013), to economics and social studies (Stirling, 2007). Diversity has been shown effective in related research on curriculum learning in language modeling, vision, and multimedia analysis (Bengio et al., 2009; Jiang et al., 2014).

Let p_i and p_j correspond to empirical frequencies of word types t_i and t_j in the training data. Let d_{ij} correspond to their semantic similarity, calculated as the cosine similarity between embeddings of t_i and t_j learned from the training data. We annotate each paragraph with the following diversity features:

- Number of word types: $\#types$
- Type-token ratio: $\frac{\#types}{\#tokens}$
- Entropy: $-\sum_i p_i \ln(p_i)$
- Simpson's index (Simpson, 1949): $\sum_i {p_i}^2$
- Quadratic entropy (Rao, 1982):[1] $\sum_{i,j} d_{ij} p_i p_j$

SIMPLICITY. Spitkovsky et al. (2010) have validated the utility of syntactic simplicity in curriculum learning for unsupervised grammar induction by showing that training on sentences in order of increasing lengths outperformed other orderings. We explore the simplicity hypothesis, albeit without prior assumptions on specific ordering of data, and extend it to additional simplicity/complexity measures of training data. Our features are inspired by prior research in second language acquisition, text simplification, and readability assessment (Schwarm and Ostendorf, 2005; Heilman et al., 2007; Pitler and Nenkova, 2008; Vajjala and

[1] Intuitively, this feature promotes paragraphs that contain semantically similar high-probability words.

Meurers, 2012). We use an off-the-shelf syntactic parser[2] (Zhang and Clark, 2011) to parse our training corpus. Then, the following features are used to measure phonological, lexical, and syntactic complexity of training paragraphs:

- Language model score
- Character language model score
- Average sentence length
- Verb-token ratio
- Noun-token ratio
- Parse tree depth
- Number of noun phrases: $\#NPs$
- Number of verb phrases: $\#VBs$
- Number of prepositional phrases: $\#PPs$

PROTOTYPICALITY. This is a group of semantic features that use insights from cognitive linguistics and child language acquisition. The goal is to characterize the curriculum of representation learning in terms of the curriculum of human language learning. We resort to the Prototype theory (Rosch, 1978), which posits that semantic categories include more central (or prototypical) as well as less prototypical words. For example, in the ANIMAL category, *dog* is more prototypical than *sloth* (because *dog* is more frequent); *dog* is more prototypical than *canine* (because *dog* is more concrete); and *dog* is more prototypical than *bull terrier* (because *dog* is less specific). According to the theory, more prototypical words are acquired earlier. We use lexical semantic databases to operationalize insights from the prototype theory in the following semantic features; the features are computed on token level and averaged over paragraphs:

- Age of acquisition (AoA) of words was extracted from the crowd-sourced database, containing over 50 thousand English words (Kuperman et al., 2012). For example, the AoA of *run* is 4.47 (years), of *flee* is 8.33, and of *abscond* is 13.36. If a word was not found in the database it was assigned the maximal age of 25.
- Concreteness ratings on the scale of 1–5 (1 is most abstract) for 40 thousand English lemmas (Brysbaert et al., 2014). For example, *cookie* is rated as 5, and *spirituality* as 1.07.
- Imageability ratings are taken from the MRC psycholinguistic database (Wilson, 1988). Following Tsvetkov et al. (2014), we used the MRC annotations as seed, and propagated the ratings to all vocabulary words using the word embeddings as features in an ℓ_2-regularized logistic regression classifier.
- Conventionalization features count the number of "conventional" words and phrases in a paragraph. Assuming that a Wikipedia title is a proxy to a conventionalized concept, we counted the number of existing titles (from a database of over 4.5 million titles) in the paragraph.
- Number of syllables scores are also extracted from the AoA database; out-of-database words were annotated as 5-syllable words.
- Relative frequency in a supersense was computed by marginalizing the word frequencies in the training corpus over coarse semantic categories defined in the WordNet (Fellbaum, 1998; Ciaramita and Altun, 2006). There are 41 supersense types: 26 for nouns and 15 for verbs, e.g., NOUN.ANIMAL and VERB.MOTION. For example, in NOUN.ANIMAL the relative frequency of *human* is 0.06, of *dog* is 0.01, of *bird* is 0.01, of *cattle* is 0.009, and of *bumblebee* is 0.0002.
- Relative frequency in a synset was calculated similarly to the previous feature category, but word frequencies were marginalized over WordNet synsets (more fine-grained synonym sets). For example, in the synset {*vet, warhorse, veteran, oldtimer, seasoned stager*}, *veteran* is the most prototypical word, scoring 0.87.

3 Evaluation Benchmarks

We evaluate the utility of the pretrained word embeddings as features in downstream NLP tasks. We choose the following off-the-shelf models that utilize pretrained word embeddings as features:

Sentiment Analysis (Senti). Socher et al. (2013) created a treebank of sentences annotated with fine-grained sentiment labels on phrases and sentences from movie review excerpts. The coarse-grained treebank of positive and negative classes has been split into training, development, and test datasets containing 6,920, 872, and 1,821 sentences, respectively. We use the average of the word vectors of a given sentence as a feature vector for classification (Faruqui et al., 2015; Sedoc

[2] http://http://people.sutd.edu.sg/~yue_zhang/doc

et al., 2016). The ℓ_2-regularized logistic regression classifier is tuned on the dev set and accuracy is reported on the test set.

Named Entity Recognition (NER). Named entity recognition is the task of identifying proper names in a sentence, such as names of persons, locations etc. We use the recently proposed LSTM-CRF NER model (Lample et al., 2016) which trains a forward-backward LSTM on a given sequence of words (represented as word vectors), the hidden units of which are then used as (the only) features in a CRF model (Lafferty et al., 2001) to predict the output label sequence. We use the CoNLL 2003 English NER dataset (Tjong Kim Sang and De Meulder, 2003) to train our models and present results on the test set.

Part of Speech Tagging (POS). For POS tagging, we again use the LSTM-CRF model (Lample et al., 2016), but instead of predicting the named entity tag for every word in a sentence, we train the tagger to predict the POS tag of the word. The tagger is trained and evaluated with the standard Penn TreeBank (PTB) (Marcus et al., 1993) training, development and test set splits as described in Collins (2002).

Dependency Parsing (Parse). Dependency parsing is the task of identifying syntactic relations between the words of a sentence. For dependency parsing, we train the stack-LSTM parser of Dyer et al. (2015) for English on the universal dependencies v1.1 treebank (Agić et al., 2015) with the standard dev and test splits, reporting unlabeled attachment scores (UAS) on the test data. We remove all part-of-speech and morphology features from the data, and prevent the model from optimizing the word embeddings used to represent each word in the corpus, thereby forcing the parser to rely completely on the pretrained embeddings.

4 Experiments

Data. All models were trained on Wikipedia articles, split to paragraph-per-line. Texts were cleaned, tokenized, numbers were normalized by replacing each digit with "DG", all types that occur less than 10 times were replaces by the "UNK" token, the data was not lowercased. We list data sizes in table 1.

# paragraphs	# tokens	# types
2,532,361	100,872,713	156,663

Table 1: Training data sizes.

Setup. 100-dimensional word embeddings were trained using the `cbow` model implemented in the word2vec toolkit (Mikolov et al., 2013).[3] All training data was used, either shuffled or ordered by a curriculum. As described in §3, we modified the extrinsic tasks to learn solely from word embeddings, without additional features. All models were learned under same conditions, across curricula: in Parse, NER, and POS we limited the number of training iterations to 3, 3, and 1, respectively. This setup allowed us to evaluate the effect of curriculum without additional interacting factors.

Experiments. In all the experiments we first train word embedding models, then the word embeddings are used as features in four extrinsic tasks (§3). We tune the tasks on dev data, and report results on the test data. The only component that varies across the experiments is order of paragraphs in the training corpus—the curriculum. We compare the following experimental setups:

- **Shuffled** baselines: the curriculum is defined by random shuffling the training data. We shuffled the data 10 times, and trained 10 word embeddings models, each model was then evaluated on downstream tasks. Following Bengio et al. (2009), we report test results for the system that is closest to the median in dev scores. To evaluate variability and a range of scores that can be obtained from shuffling the data, we also report test results for systems that obtained the highest dev scores.
- **Sorted** baselines: the curriculum is defined by sorting the training data by sentence length in increasing/decreasing order, similarly to (Spitkovsky et al., 2010).
- **Coherent** baselines: the curriculum is defined by just concatenating Wikipedia articles. The goal of this experiment is to evaluate the importance of semantic coherence in training data. Our intuition is that a coherent curriculum

[3] To evaluate the impact of curriculum learning, we enforced sequential processing of data organized in a predefined order of training examples. To control for sequential processing, word embedding were learned by running the `cbow` using a single thread for one iteration.

can improve models, since words with similar meanings and similar contexts are grouped when presented to the learner.

- **Optimized curriculum** models: the curriculum is optimized using the BayesOpt. We evaluate and compare models optimized using features from one of the three feature groups (§2.2). As in the shuffled baselines, we fix the number of trials (here, BayesOpt iterations) to 10, and we report test results of systems that obtained best dev scores.

Results. Experimental results are listed in table 2. Most systems trained with curriculum substantially outperform the strongest of all baselines. These results are encouraging, given that all word embedding models were trained on the same set of examples, only in different order, and display the indirect influence of the data curriculum on downstream tasks. These results support our assumption that curriculum matters. Albeit not as pronounced as with optimized curriculum, sorting paragraphs by length can also lead to substantial improvements over random baselines, but there is no clear recipe on whether the models prefer curricula sorted in an increasing or decreasing order. These results also support the advantage of a task-specific optimization framework over a general, intuition-guided recipe. An interesting result, also, that shuffling is not essential: systems trained on coherent data are on par (or better) than the shuffled systems.[4] In the next section, we analyze these results qualitatively.

5 Analysis

What are task-specific curriculum preferences? We manually inspect learned features and curriculum-sorted corpora, and find that best systems are obtained when their embeddings are learned from curricula appropriate to the downstream tasks. We discuss below several examples.

[4]Note that in the shuffled NER baselines, best dev results yield lower performance on the test data. This implies that in the standard dev/test splits the dev and test sets are not fully compatible or not large enough. We also observe this problem in the curriculum-optimized Parse-prototypicality and Senti-diversity systems. The dev scores for the Parse systems are 76.99, 76.47, 76.47 for diversity, prototypicality, and simplicity, respectively, but the prototypicality-sorted parser performs poorly on test data. Similarly in the sentiment analysis task, the dev scores are 69.15, 69.04, 69.49 for diversity, prototypicality, and simplicity feature groups. Senti-diversity scores, however, are lower on the test data, although the dev results are better than in Senti-simplicity. This limitation of the standard dev/test splits is beyond the scope of this paper.

POS and Parse systems converge to the same set of weights, when trained on features that provide various measures of syntactic simplicity. The features with highest coefficients (and thus the most important features in sorting) are $\#NPs$, Parse tree depth, $\#VPs$, and $\#PPs$ (in this order). The sign in the $\#NPs$ feature weight, however, is the opposite from the other three feature weights (i.e., sorted in different order). $\#NPs$ is sorted in the increasing order of the number of noun phrases in a paragraph, and the other features are sorted in the decreasing order. Since Wikipedia corpus contains a lot of partial phrases (titles and headings), such curriculum promotes more complex, full sentences, and demotes partial sentences.

Best Senti system is sorted by prototypicality features. Most important features (with the highest coefficients) are Concreteness, Relative frequency in a supersense, and the Number of syllables. First two are sorted in decreasing order (i.e. paragraphs are sorted from more to less concrete, and from more to less prototypical words), and the Number of syllables is sorted in increasing order (this also promotes simpler, shorter words which are more prototypical). We hypothesize that this soring reflects the type of data that Sentiment analysis task is trained on: it is trained on movie reviews, that are usually written in a simple, colloquial language.

Unlike POS, Parse, and Senti systems, all NER systems prefer curricula in which texts are sorted from short to long paragraphs. The most important features in the best (simplicity-sorted) system are $\#PPs$ and Verb-token ratio, both sorted from less to more occurrences of prepositional and verb phrases. Interestingly, most of the top lines in the NER system curricula contain named entities, although none of our features mark named entities explicitly. We show top lines in the simplicity-optimized system in figure 2.

Finally, in all systems sorted by prototypicality, the last line is indeed not a prototypical word *Donaudampfschiffahrtselektrizitätenhauptbetriebswerkbauunterbeamtengesellschaft*, which is an actual word in German, frequently used as an example of compounding in synthetic languages, but rarely (or never?) used by German speakers.

Weighting examples according to curriculum. Another way to integrate curriculum in word embedding training is to weight training examples according to curriculum during word represen-

		Senti	NER	POS	Parse
Shuffled	median	66.01	85.88	96.35	75.08
	best	66.61	85.50	96.38	76.40
Sorted	long→short	66.78	85.22	96.47	75.85
	short→long	66.12	85.49	96.20	75.31
Coherent	original order	66.23	85.99	96.47	76.08
Optimized curriculum	diversity	66.06	86.09	96.59	**76.63**
	prototypicality	**67.44**	85.96	96.53	75.81
	simplicity	67.11	**86.42**	**96.62**	76.54

Table 2: Evaluation of the impact of the curriculum of word embeddings on the downstream tasks.

Trimingham " Golf " ball .
Adélie penguin
" **Atriplex** " leaf UNK UNK
Hồng Lĩnh mountain
Anneli Jäätteenmäki UNK cabinet
Gävle goat
Early telescope observations .
Scioptric ball
Matryoshka doll
Luxembourgian passport
Australian Cashmere goat
Plumbeous water redstart
Dagebüll lighthouse
Vecom FollowUs . tv
Syracuse Junction railroad .
San Clemente Island goat
Tychonoff plank

Figure 2: Most of the top lines in best-scoring NER system contain named entities, although our features do not annotate named entities explicitly.

tation training. We modify the `cbow` objective $\sum_{t=1}^{T} \log p(w_t|w_{t-c}..w_{t+c})$ as follows:[5]

$$\sum_{t=1}^{T}(\frac{1}{1+e^{-weight(w_t)}} + \lambda) \log p(w_t|w_{t-c}..w_{t+c})$$

Here, $weight(w_t)$ denotes the score attributed to the token w_t, which is the z-normalized score of the paragraph they appear in (equation **??**); λ=0.5 is determined empirically. $\log p(w_t)|w_{t-c}..w_{t+c})$ computes the probability of predicting word w_t, using the context of c words to the left and right of w_t. Notice that this quantity is no longer a proper probability, as we are not normalizing over the weights $weight(w_t)$ over all tokens. However, the optimization in word2vec is performed using stochastic gradient descent, optimizing for a single token at each iteration. This yields a normalizer of 1 for each iteration, yielding the same gradient as the original `cbow` model.

We retrain our best curriculum-sorted systems with the modified objective, also controlling for curriculum. The results are shown in table 3. We find that the benefit of integrating curriculum in training objective of word representations is not evident across tasks: Senti and NER systems trained on vectors with the modified objective substantially outperform best results in table 2; POS and Parse perform better than the baselines but worse than the systems with the original objective.

	Senti	NER	POS	Parse
curriculum	67.44	86.42	**96.62**	**76.63**
`cbow`+curric	**68.26**	**86.49**	96.48	76.54

Table 3: Evaluation of the impact of curriculum integrated in the `cbow` objective.

Are we learning task-specific curricula? One way to assess whether we learn meaningful task-specific curriculum preferences is to compare curricula learned by one downstream task across different feature groups. If learned curricula are similar in, say, NER system, despite being optimized once using diversity features and once using prototypicality features—two disjoint feature sets—we can infer that the NER task prefers word embeddings learned from examples presented in a certain order, regardless of specific optimization features. For each downstream task, we thus measure Spearman's rank correlation between the curricula optimized using diversity (D), or prototypicality (P), or simplicity (S) feature sets. Prior to measuring correlations, we remove duplicate lines from

[5]The modified word2vec tool is located at `https://github.com/wlin12/wang2vec`.

the training corpora. Correlation results across tasks and across feature sets are shown in table 4.

The general pattern of results is that if two systems score higher than baselines, training sentences of their feature embeddings have similar curricula (i.e., the Spearman's ρ is positive), and if two systems disagree (one is above and one is below the baseline), then their curricula also disagree (i.e., the Spearman's ρ is negative or close to zero). NER systems all outperform the baselines and their curricula have high correlations. Moreover, NER sorted by diversity and simplicity have better scores than NER sorted by prototypicality, and in line with these results $\rho(\text{S,D})_{NER} > \rho(\text{P,S})_{NER}$ and $\rho(\text{S,D})_{NER} > \rho(\text{D,P})_{NER}$. Similar pattern of results is in POS correlations. In Parse systems, also, diversity and simplicity features yielded best parsing results, and $\rho(\text{S,D})_{Parse}$ has high positive correlation. The prototypicality-optimized parser performed poorly, and its correlations with better systems are negative. The best parser was trained using the diversity-optimized curriculum, and thus $\rho(\text{D,P})_{Parse}$ is the lowest. Senti results follow similar pattern of curricula correlations.

	Senti	**NER**	**POS**	**Parse**
ρ(D, P)	-0.68	0.76	0.66	-0.76
ρ(P, S)	0.33	0.75	0.75	-0.45
ρ(S, D)	-0.16	0.81	0.51	0.67

Table 4: Curricula correlations across feature groups.

Curriculum learning vs. data selection. We compare the task of curriculum learning to the task of data selection (reducing the set of training instances to more important or cleaner examples). We reduce the training data to the subset of 10% of tokens, and train downstream tasks on the reduced training sets. We compare system performance trained using the top 10% of tokens in the best curriculum-sorted systems (Senti-prototypicality, NER-implicity, POS-simplicity, Parse-diversity) to the systems trained using the top 10% of tokens in a corpus with randomly shuffled paragraphs.[6] The results are listed in table 5.

The curriculum-based systems are better in POS and in Parse systems, mainly because these tasks prefer vectors trained on curricula that promote well-formed sentences (as discussed above). Conversely, NER prefers vectors trained on corpora that begin with named entities, so most of the tokens in the reduced training data are constituents in short noun phrases. These results suggest that the tasks of data selection and curriculum learning are different. Curriculum is about strong initialization of the models and time-course learning, which is not necessarily sufficient for data reduction.

	Senti	**NER**	**POS**	**Parse**
random	63.97	**82.35**	96.22	69.11
curriculum	**64.47**	76.96	**96.55**	**72.93**

Table 5: Data selection results.

6 Related Work

Two prior studies on curriculum learning in NLP are discussed in the paper (Bengio et al., 2009; Spitkovsky et al., 2010). Curriculum learning and related research on self-paced learning has been explored more deeply in computer vision (Bengio et al., 2009; Kumar et al., 2010; Lee and Grauman, 2011) and in multimedia analysis (Jiang et al., 2015). Bayesian optimization has also received little attention in NLP. GPs were used in the task of machine translation quality estimation (Cohn and Specia, 2013) and in temporal analysis of social media texts (Preotiuc-Pietro and Cohn, 2013); TPEs were used by Yogatama et al. (2015) for optimizing choices of feature representations—n-gram size, regularization choice, etc.—in supervised classifiers.

7 Conclusion

We used Bayesian optimization to optimize curricula for training dense distributed word representations, which, in turn, were used as the sole features in NLP tasks. Our experiments confirmed that better curricula yield stronger models. We also conducted an extensive analysis, which sheds better light on understanding of text properties that are beneficial for model initialization. The proposed novel technique for finding an optimal curriculum is general, and can be used with other datasets and models.

[6] Top $n\%$ tokens are used rather than top $n\%$ paragraphs because in all tasks except NER curriculum-sorted corpora begin with longer paragraphs. Thus, with top $n\%$ paragraphs our systems would have an advantage over random systems due to larger vocabulary sizes and not necessarily due to a better subset of data.

Acknowledgments

This work was supported by the National Science Foundation through award IIS-1526745. We are grateful to Nathan Schneider, Guillaume Lample, Waleed Ammar, Austin Matthews, and the anonymous reviewers for their insightful comments.

References

Željko Agić, Maria Jesus Aranzabe, Aitziber Atutxa, Cristina Bosco, Jinho Choi, Marie-Catherine de Marneffe, Timothy Dozat, Richárd Farkas, Jennifer Foster, Filip Ginter, Iakes Goenaga, Koldo Gojenola, Yoav Goldberg, Jan Hajič, Anders Trærup Johannsen, Jenna Kanerva, Juha Kuokkala, Veronika Laippala, Alessandro Lenci, Krister Lindén, Nikola Ljubešić, Teresa Lynn, Christopher Manning, Héctor Alonso Martínez, Ryan McDonald, Anna Missilä, Simonetta Montemagni, Joakim Nivre, Hanna Nurmi, Petya Osenova, Slav Petrov, Jussi Piitulainen, Barbara Plank, Prokopis Prokopidis, Sampo Pyysalo, Wolfgang Seeker, Mojgan Seraji, Natalia Silveira, Maria Simi, Kiril Simov, Aaron Smith, Reut Tsarfaty, Veronika Vincze, and Daniel Zeman. 2015. Universal dependencies 1.1. LINDAT/CLARIN digital library at Institute of Formal and Applied Linguistics, Charles University in Prague.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proc. ICML*, pages 41–48.

James S Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. 2011. Algorithms for hyper-parameter optimization. In *Proc. NIPS*, pages 2546–2554.

Marc Brysbaert, Amy Beth Warriner, and Victor Kuperman. 2014. Concreteness ratings for 40 thousand generally known english word lemmas. *Behavior research methods*, 46(3):904–911.

Massimiliano Ciaramita and Yasemin Altun. 2006. Broad-coverage sense disambiguation and information extraction with a supersense sequence tagger. In *Proc. EMNLP*, pages 594–602.

Trevor Cohn and Lucia Specia. 2013. Modelling annotator bias with multi-task Gaussian processes: An application to machine translation quality estimation. In *Proc. ACL*, pages 32–42.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proc. EMNLP*, pages 1–8.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proc. ACL*.

Jeffrey L Elman. 1993. Learning and development in neural networks: The importance of starting small. *Cognition*, 48(1):71–99.

Manaal Faruqui, Jesse Dodge, Sujay K. Jauhar, Chris Dyer, Eduard Hovy, and Noah A. Smith. 2015. Retrofitting word vectors to semantic lexicons. In *Proc. NAACL*.

Christiane Fellbaum, editor. 1998. *WordNet: an electronic lexical database*. MIT Press.

Kevin Gimpel, Dhruv Batra, Chris Dyer, and Gregory Shakhnarovich. 2013. A systematic exploration of diversity in machine translation. In *Proc. EMNLP*, pages 1100–1111.

Michael J. Heilman, Kevyn Collins-Thompson, Jamie Callan, and Maxine Eskenazi. 2007. Combining lexical and grammatical features to improve readability measures for first and second language texts. In *Proc. NAACL*, pages 460–467.

Philipp Hennig and Christian J Schuler. 2012. Entropy search for information-efficient global optimization. *The Journal of Machine Learning Research*, 13(1):1809–1837.

Matthew D Hoffman, Eric Brochu, and Nando de Freitas. 2011. Portfolio allocation for Bayesian optimization. In *Proc. UAI*, pages 327–336.

Lu Jiang, Deyu Meng, Shoou-I Yu, Zhenzhong Lan, Shiguang Shan, and Alexander Hauptmann. 2014. Self-paced learning with diversity. In *Proc. NIPS*, pages 2078–2086.

Lu Jiang, Deyu Meng, Qian Zhao, Shiguang Shan, and Alexander G Hauptmann. 2015. Self-paced curriculum learning. In *Proc. AAAI*, volume 2, page 6.

Donald R Jones. 2001. A taxonomy of global optimization methods based on response surfaces. *Journal of global optimization*, 21(4):345–383.

Robert Kail. 1990. *The development of memory in children*. W. H. Freeman and Company, 3rd edition.

M. P. Kumar, Benjamin Packer, and Daphne Koller. 2010. Self-paced learning for latent variable models. In *Proc. NIPS*, pages 1189–1197.

Victor Kuperman, Hans Stadthagen-Gonzalez, and Marc Brysbaert. 2012. Age-of-acquisition ratings for 30,000 english words. *Behavior Research Methods*, 44(4):978–990.

Harold J Kushner. 1964. A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. *Journal of Basic Engineering*, 86(1):97–106.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. pages 282–289.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proc. NAACL*.

Yong Jae Lee and Kristen Grauman. 2011. Learning the easy things first: Self-paced visual category discovery. In *Proc. CVPR*, pages 1721–1728.

Anne E Magurran. 2013. *Measuring biological diversity*. John Wiley & Sons.

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *Proc. ICLR*.

Jonas Močkus, Vytautas Tiesis, and Antanas Žilinskas. 1978. On Bayesian methods for seeking the extremum. *Towards global optimization*, 2(117-129):2.

Emily Pitler and Ani Nenkova. 2008. Revisiting readability: A unified framework for predicting text quality. In *Proc. EMNLP*, pages 186–195.

Daniel Preotiuc-Pietro and Trevor Cohn. 2013. A temporal model of text periodicities using gaussian processes. In *Proc. EMNLP*, pages 977–988.

C Radhakrishna Rao. 1982. Diversity and dissimilarity coefficients: a unified approach. *Theoretical population biology*, 21(1):24–43.

Carl Edward Rasmussen. 2006. Gaussian Processes for machine learning.

Eleanor Rosch. 1978. Principles of categorization. In Eleanor Rosch and Barbara B. Lloyd, editors, *Cognition and categorization*, pages 28–71.

Michael L Rosenzweig. 1995. *Species diversity in space and time*. Cambridge University Press.

Sarah E. Schwarm and Mari Ostendorf. 2005. Reading level assessment using support vector machines and statistical language models. In *Proc. ACL*, pages 523–530.

João Sedoc, Jean Gallier, Lyle Ungar, and Dean Foster. 2016. Semantic word clusters using signed normalized graph cuts. *arXiv preprint arXiv:1601.05403*.

Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando de Freitas. 2016. Taking the human out of the loop: A review of Bayesian optimization. *Proc. IEEE*, 104(1):148–175.

Edward H Simpson. 1949. Measurement of diversity. *Nature*.

Burrhus Frederic Skinner. 1938. The behavior of organisms: an experimental analysis. *An Experimental Analysis*.

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. 2012. Practical Bayesian optimization of machine learning algorithms. In *Proc. NIPS*, pages 2951–2959.

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

Valentin I Spitkovsky, Hiyan Alshawi, and Dan Jurafsky. 2010. From baby steps to leapfrog: How less is more in unsupervised dependency parsing. In *Proc. NAACL*, pages 751–759.

Niranjan Srinivas, Andreas Krause, Sham M Kakade, and Matthias Seeger. 2010. Gaussian process optimization in the bandit setting: No regret and experimental design. In *Proc. ICML*, pages 1015–1022.

Andy Stirling. 2007. A general framework for analysing diversity in science, technology and society. *Journal of the Royal Society Interface*, 4(15):707–719.

E Ke Tang, Ponnuthurai N Suganthan, and Xin Yao. 2006. An analysis of diversity measures. *Machine Learning*, 65(1):247–271.

William R Thompson. 1933. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294.

Erik F Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proc. CoNLL*.

Yulia Tsvetkov, Leonid Boytsov, Anatole Gershman, Eric Nyberg, and Chris Dyer. 2014. Metaphor detection with cross-lingual model transfer. In *Proc. ACL*.

Sowmya Vajjala and Detmar Meurers. 2012. On improving the accuracy of readability classification using insights from second language acquisition. In *Proc. BEA*, pages 163–173.

Michael Wilson. 1988. MRC psycholinguistic database: Machine-usable dictionary, version 2.00. *Behavior Research Methods, Instruments, & Computers*, 20(1):6–10.

Dani Yogatama, Lingpeng Kong, and Noah A Smith. 2015. Bayesian optimization of text representations. In *Proc. EMNLP*, pages 2100–2105.

Yue Zhang and Stephen Clark. 2011. Syntactic processing using the generalized perceptron and beam search. *Computational Linguistics*, 37(1):105–151.