mathematics

MDPI

Article

To Batch or Not to Batch? Comparing Batching and Curriculum Learning Strategies across Tasks and Datasets

Laura Burdick *,†, Jonathan K. Kummerfeld and Rada Mihalcea

Computer Science and Engineering, University of Michigan, Ann Arbor, MI 48109, USA; jkummerf@umich.edu (J.K.K.); mihalcea@umich.edu (R.M.)
* Correspondence: lburdick@umich.edu
† Current address: 2260 Hayward Street, Ann Arbor, MI 48109, USA.

Abstract: Many natural language processing architectures are greatly affected by seemingly small design decisions, such as batching and curriculum learning (how the training data are ordered during training). In order to better understand the impact of these decisions, we present a systematic analysis of different curriculum learning strategies and different batching strategies. We consider multiple datasets for three tasks: text classification, sentence and phrase similarity, and part-of-speech tagging. Our experiments demonstrate that certain curriculum learning and batching decisions do increase performance substantially for some tasks.

Keywords: natural language processing; word embeddings; batching; word2vec; curriculum learning; text classification; phrase similarity; part-of-speech tagging

Citation: Burdick, L.; Kummerfeld, J.K.; Mihalcea, R. To Batch or Not to Batch? Comparing Batching and Curriculum Learning Strategies across Tasks and Datasets. *Mathematics* **2021**, *9*, 2234. https://doi.org/10.3390/math9182234

Academic Editors: Florentina Hristea, Cornelia Caragea and David Pugalee

Received: 28 June 2021
Accepted: 4 September 2021
Published: 11 September 2021

1. Introduction

When designing architectures for tasks in natural language processing (NLP), relatively small methodological details can have a huge impact on the performance of the system. In this paper, we consider several methodological decisions that impact NLP systems that are based on word embeddings. Word embeddings are low-dimensional, dense vector representations that capture semantic and syntactic properties of words. They are often used in larger systems to accomplish downstream tasks.

We analyze two methodological decisions involved in creating word embeddings: batching and curriculum learning. For batching, we consider what batching method to use, and what batch size to use. We consider two batching methods, which we denote as *basic batching* and *cumulative batching*. Curriculum learning is the process of ordering the data during training. We consider three curriculum learning strategies: *ascending curriculum*, *descending curriculum*, and *default curriculum*.

Our batching and curriculum learning choices are evaluated using three downstream tasks: text classification, sentence and phrase similarity, and part-of-speech tagging. We consider a variety of datasets of different sizes in order to understand how these strategies work across diverse tasks and data.

We show that for some tasks, batching and curriculum learning decisions do not have a significant impact, but for other tasks, such as text classification on small datasets, these decisions are important considerations. This paper is an empirical study, and while we make observations about different batching and curriculum learning decisions, we do not explore the theoretical reasons for these observations.

To begin, we survey related work before introducing the methodology that we use to create the word embeddings and apply batching and curriculum learning. Next, we define architectures for our three downstream tasks. Finally, we present our results, and discuss future work and conclusions.

2. Related Work

Our work builds off previous work on word embeddings, batching, and curriculum learning.

2.1. Word Embeddings

Throughout this paper, we use a common word embedding algorithm, word2vec [1,2], which uses a shallow neural network to learn embeddings by predicting context words. We use the skip-gram word2vec model, a one-layer feed-forward neural network which tries to optimize the log probability of a target word, given its context words.

In many cases, word embeddings are used as features in a larger system architecture. The work of Collobert et al. [3] was an early paper that took this approach, incorporating word embeddings as inputs to a neural network that performed part-of-speech tagging, chunking, named entity recognition, and semantic role labeling. This line of work has been expanded to include many other tasks, including text similarity [4], sentiment analysis [5], and machine translation [6]. In this paper, we use word embeddings in systems for text classification, sentence and phrase similarity, and part-of-speech tagging.

We explore two methodological decisions in creating word embeddings: batching and curriculum learning.

2.2. Batching

We use two batching approaches (denoted as *basic batching* and *cumulative batching*). Basic batching was first introduced in Bengio et al. [7], and cumulative batching was first introduced in Spitkovsky et al. [8]. These approaches are described in more detail in Section 3.2.

While we use the batching approaches described in these papers, we analyze their performance on different tasks and datasets. Basic batching was originally proposed for synthetic vision and word representation learning tasks, while cumulative batching was applied to unsupervised dependency parsing. We use these batching techniques on NLP architectures built with word embeddings for the tasks of text classification, sentence and phrase similarity, and part-of-speech tagging.

In addition to using two batching techniques, we vary the number of the batches that we use. This was studied previously; Smith et al. [9] showed that choosing a good batch size can decrease the number of parameter updates to a network.

2.3. Curriculum Learning

This work also explores curriculum learning applied to word embeddings. Curriculum refers to the order that the data are presented to the embedding algorithm. Previous work has shown that the curriculum of the training data has some effect on the performance of the created embeddings [10]. Curriculum learning has also been explored for other tasks in NLP, including natural language understanding [11] and domain adaptation [12].

3. Materials and Methods

In order to evaluate the effectiveness of different batching and curriculum learning techniques, we choose a diverse set of tasks and architectures to explore. Specifically, we consider the downstream tasks of text classification, sentence and phrase similarity, and part-of-speech (POS) tagging. These were chosen because they have varying degrees of complexity. Sentence and phrase similarity, where word embeddings are compared using cosine similarity, has a very simple architecture, while part-of-speech tagging uses a more complex LSTM architecture. Text classification uses a linear classifier, which is more complex than a simple cosine similarity, but less complex than a neural network.

Figure 1 shows the experimental setup for these three tasks. For each task, we begin by training word2vec word embeddings, using Wikipedia training data. We apply batching and curriculum learning to the process of training word embeddings. These embeddings are then passed to a task-specific architecture.

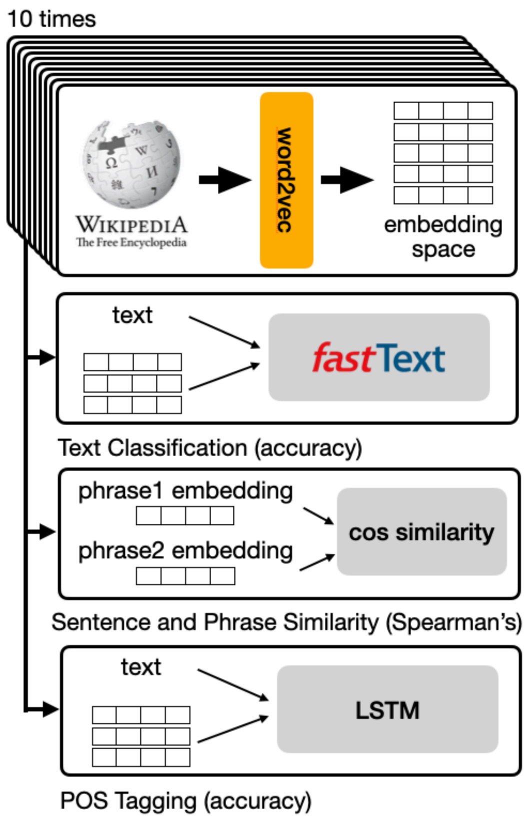

Figure 1. Experimental setup for text classification, sentence and phrase similarity, and POS tagging.

3.1. Initial Embedding Spaces

To obtain word embeddings, we begin with a dataset of sentences from Wikipedia (5,269,686 sentences; 100,003,406 words; 894,044 tokens) and create word2vec skip-gram embeddings [1]. We use 300 dimensions and a context window size of 5.

Because the word2vec embedding algorithm uses a shallow neural network, there is randomness inherent to it. This algorithmic randomness can cause variation in the word embeddings created [13,14]. In order to account for variability in embeddings, we train 10 embedding spaces with different random initializations for word2vec. (We randomize the algorithm by changing the random seed. We use the following ten random seeds: 2518, 2548, 2590, 29, 401, 481, 485, 533, 725, 777.) We use these 10 spaces to calculate the average performance and standard deviation, which allows us to characterize the variation that we see within the algorithm.

During the process of creating word embeddings, we use different batching and curriculum learning strategies.

3.2. Batching

We apply batching to the Wikipedia dataset input to the word embedding algorithm. We batch words (and their contexts) using two strategies: basic batching and cumulative batching, visualized in Figure 2. For each batching strategy, we consider different numbers of batches, ranging exponentially between 2 and 200.

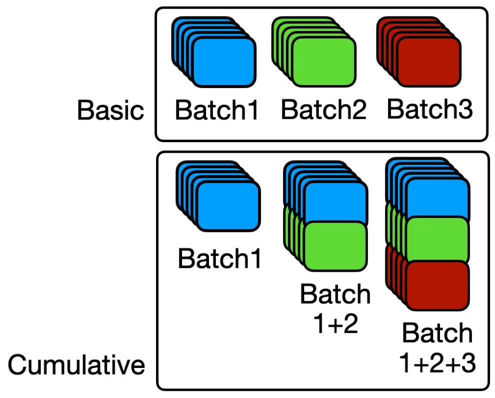

Figure 2. Basic vs. cumulative batching. Rectangles represent chunks of the training data, with different colors representing different sections of the data.

3.2.1. Basic Batching

As described in Bengio et al. [7], we split the data up into X disjoint batches. Each batch is processed sequentially, and each batch is run for n epochs (Batch 1 runs for n epochs, then Batch 2 runs for n epochs, etc.). Once a batch is finished processing, it is discarded and never returned to. Both X and n are hyperparameters. We try different values of X between 2 and 200; we set $n = 5$ for all experiments.

3.2.2. Cumulative Batching

Our second batching strategy [8] begins in the same way, with the data split up into X disjoint batches. In this strategy, the batches are processed cumulatively (Batch 1 is run for n epochs, then Batches 1 and 2 combined are run for n epochs, etc.). We try different values of X between 2 and 200; we set $n = 5$ for all experiments.

3.3. Curriculum Learning

In addition to batching, we apply different curriculum learning strategies to the Wikipedia dataset input to the word embedding algorithm.

We consider three different curricula for the data: the *default* order of Wikipedia sentences, *descending* order by sentence length (longest to shortest), and *ascending* order by sentence length (shortest to longest). Note that curriculum learning only applies to the Wikipedia dataset used to create the word embeddings, rather than the task-specific datasets used for training.

Qualitatively looking at Wikipedia sentences ordered by length, both the shortest and the longest sentences tend to be unnatural sounding. The shortest sentences are only a single token, such as a single word or a single punctuation mark. Some of these are most likely the result of incorrect sentence tokenization. The longest sentences tend to be either run-on sentences, or lists of a large number of items. For instance, the longest sentence is 725 tokens long, and it lists numerical statistics for different countries. This unnaturalness may adversely affect the embedding algorithm when using either the ascending or descending curriculum. It is possible that a more complex ordering of the data would achieve better performance; we leave this exploration to future work.

When evaluating curriculum learning and batching strategies, our baseline strategy is a default curriculum with basic batching.

Once we have created word embeddings using a specific batching and curriculum learning strategy, the embeddings are input into task-specific architectures for each of our three downstream tasks.

3.4. Task 1: Text Classification

The first task we consider is text classification—deciding what category a particular document falls into. We evaluate 11 datasets, shown in Table 1. (For all tasks, sentences are tokenized using NLTK's Tokenizer.) These datasets span a wide range of sizes (from 96 sentences to 3.6 million training sentences), as well as number of classes to be categorized (from 2 to 14).

Table 1. Data statistics (number of training sentences, number of test sentences, number of classes) for text classification. The first eight datasets are from Zhang et al. [15]. Two datasets have both a polarity (pol.) version with two classes and a full version with more classes.

Dataset	# Sent. (Train)	# Sent. (Test)	# Classes
Amazon Review (pol.)	3.6*e*6	4*e*5	2
Amazon Review (full)	3*e*6	6.5*e*5	5
Yahoo! Answers	1.4*e*6	6*e*4	10
Yelp Review (full)	6.5*e*5	5*e*4	5
Yelp Review (pol.)	5.6*e*5	3.8*e*4	2
DBPedia	5.6*e*5	7*e*4	14
Sogou News	4.5*e*5	6*e*4	5
AG News	1.2*e*5	7600	4
Open Domain Deception	5733	1435	2
Personal Email	260	89	2
Real Life Deception	96	25	2

Of particular note are three datasets that are at least an order of magnitude smaller than the other datasets. These are the Open Domain Deception Dataset [16] and the Real Life Deception Dataset [17], both of which classify statements as truthful or deceptive, as well as the Personal Email Dataset [18], which classifies e-mail messages as personal or non-personal.

After creating embedding spaces, we use fastText [19] for text classification. (Available online at https://fasttext.cc/. (accessed on 7 September 2021.) FastText represents sentences as a bag of words and trains a linear classifier to classify the sentences. The performance is measured using accuracy.

3.5. Task 2: Sentence and Phrase Similarity

The second task that we consider is sentence and phrase similarity: determining how similar two sentences or phrases are. We consider three evaluation datasets, shown in Table 2. The Human Activity Dataset [20] consists of pairs of human activities with four annotated relations each (similarity, relatedness, motivational alignment [MA], and perceived actor congruence [PAC]). The STS Benchmark [21] has pairs of sentences with semantic similarity scores, and the SICK dataset [22] has pairs of sentences with relatedness scores.

Table 2. Data statistics for sentence and phrase similarity.

Dataset	# Pairs (Train)	# Pairs (Test)	# Tokens (Train)
Human Activity	1373	1000	1446
STS Benchmark	5749	1379	14,546
SICK	4439	4906	2251

For each pair of phrases or sentences in our evaluation set, we average the embeddings for each word, and take the cosine similarity between the averaged word vectors from both phrases or sentences. We compare this with the ground truth using Spearman's correlation [23]. Spearman's correlation is a measure of rank correlation. The values of two variables are ranked in order, and then Pearson's correlation [24] (a measure of linear correlation) is taken between the ranked values. Spearman's correlation assesses monotonic relationships (are values strictly not decreasing, or strictly not increasing?), while Pearson's correlation assesses linear relationships.

3.6. *Task 3: Part-of-Speech Tagging*

Our final task is part-of-speech (POS) tagging—determining the correct part-of-speech for a given word in a sentence. For evaluation, we use two datasets: the email and answers datasets from the English Universal Dependencies Corpus (UD) [25], shown in Table 3.

Table 3. Data statistics for POS tagging.

Dataset	# Sentences (Train)	# Sentences (Test)
UD Answers	2631	438
UD Email	3770	606

After creating embedding spaces, we use a bi-directional LSTM implemented using DyNet [26] to perform POS tagging. An LSTM is a type of recurrent neural network that is able to process sequential data [27]. This is an appropriate architecture for part-of-speech tagging because we want to process words sequentially, in the order that they appear in the sentence. Our LSTM has 1 layer with a hidden dimension size of 50, and a multi-layer perceptron on the output. Performance is measured using accuracy.

4. Results

We apply the different curriculum learning and batching strategies to each task, and we consider the results to determine which strategies are most effective.

4.1. *Task 1: Text Classification*

For the larger text classification datasets (>120,000 training sentences), there are no substantial differences between different curriculum and batching strategies. However, we do see differences for the three smallest datasets, shown in Figure 3. To compare across datasets of different sizes, we show the number of sentences per batch, rather than the number of batches. Because these graphs show many combinations of curriculum and batching strategies, we report numbers on each dataset's dev set.

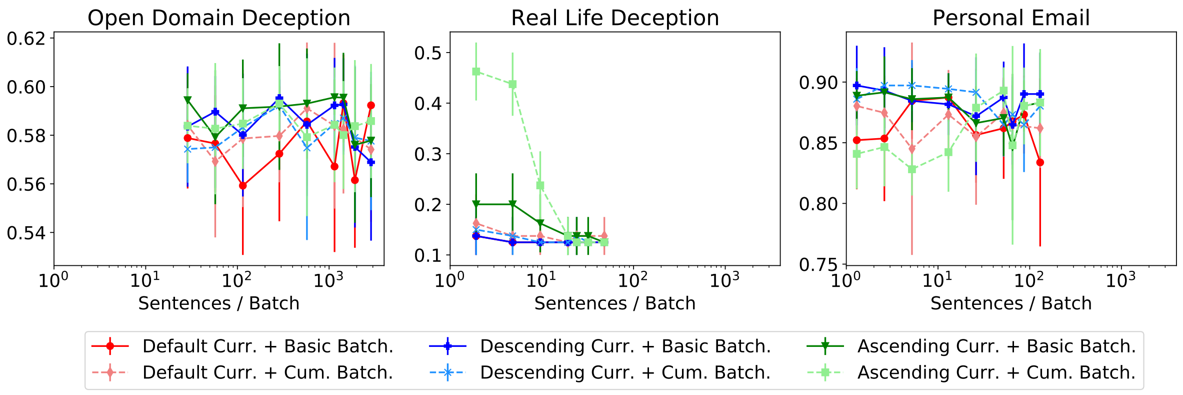

Figure 3. Accuracy scores on the development set for three text classification datasets. Different lines indicate models trained with different curriculum and batching strategies (basic, cumulative). Datasets span different ranges of the x-axis because they are different sizes. Error bars show the standard deviation over ten word2vec embedding spaces, trained using different random seeds.

On the smallest dataset, Real Life Deception (96 training sentences), we see that above approximately ten batches, ascending curriculum with cumulative batching outperforms the other methods. On the test set, we compare our best strategy (ascending curriculum with cumulative batching) with the baseline setting (default curriculum with basic batching), both with 100 batches, and we see no significant difference. This is most likely because the test set is so small (25 sentences).

4.2. Task 2: Sentence and Phrase Similarity

Next, we consider results from the sentence and phrase similarity task, shown in Figure 4. Because these graphs show many combinations of curriculum and batching strategies, we report numbers on each dataset's train set (we use the train sets rather than the dev sets because there is no training or hyperparemeter tuning).

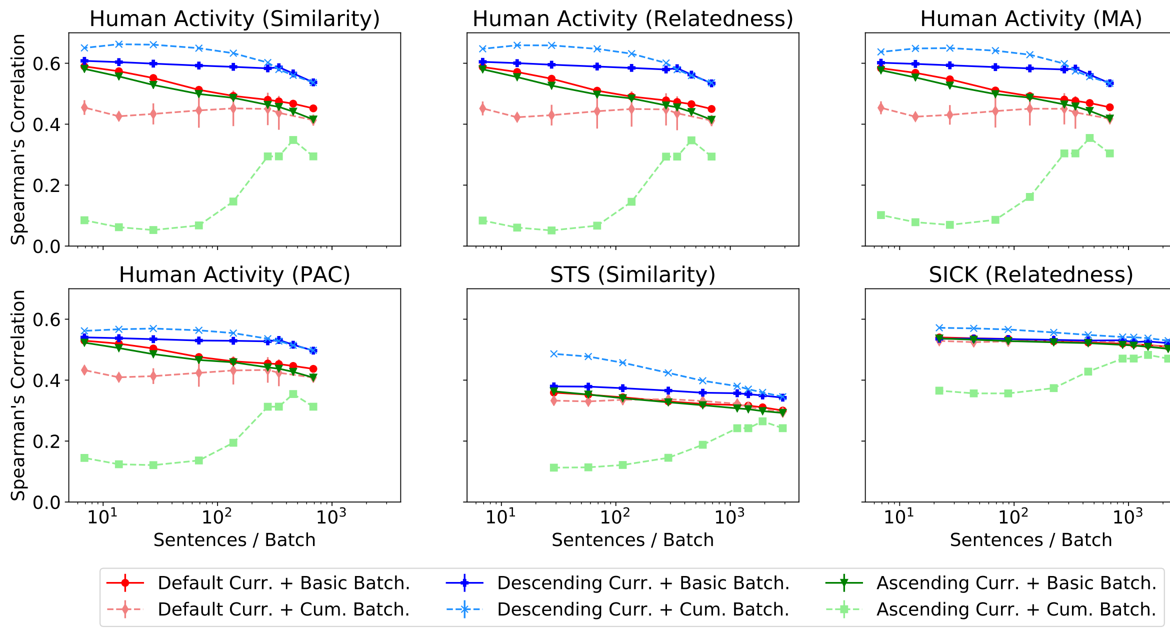

Figure 4. Spearman's correlation scores on the train set for sentence and phrase similarity tasks. Different lines indicate models trained with different curriculum and batching strategies (basic, cumulative). Datasets span different ranges of the x-axis because they are different sizes. Error bars show the standard deviation over ten word2vec embedding spaces trained using different random seeds.

First, we note that the relative performance of different strategies remains consistent across all three datasets and across all six measures of similarity. An ascending curriculum with cumulative batching performs the worst by a substantial amount, while a descending curriculum with cumulative batching performs the best by a small amount. As the number of sentences per batch increases, the margin between the different strategies decreases. On the test set, we compare our best strategy (descending curriculum with cumulative batching) with the baseline setting (default curriculum with basic batching), and we see in Table 4 that the best strategy significantly outperforms the baseline with five batches.

Table 4. Spearman's correlation on the test set for similarity tasks (all have a standard deviation of 0.0).

	Human Activity					
Dataset	**Sim.**	**Rel.**	**MA**	**PAC**	**STS**	**SICK**
Baseline	0.36	0.33	0.33	0.22	0.27	0.51
Best	0.43	0.41	0.41	0.29	0.32	0.53

For all six measures, we observe a time vs. performance trade-off: the fewer sentences are in a batch, the better the performance is, but the more computational power and time it takes to run.

4.3. Task 3: Part-of-Speech Tagging

Finally, there are no significant differences in POS tagging between batching and curriculum learning strategies. For the previous two tasks, we have seen the largest changes in performance on the smallest dataset. Both of the datasets that we use to evaluate POS tagging are relatively large (>2500 sentences in the training data), which may explain why we do not see significant performance differences here.

5. Conclusions

One strategy does not perform equally well on all tasks. On some tasks, such as POS tagging, the curriculum and batching strategies that we tried have no effect at all. Simpler tasks that rely most heavily on word embeddings, such as sentence and phrase similarity and text classification with very small datasets, benefit the most from fine-tuned curriculum learning and batching. We have shown that making relatively small changes to curriculum learning and batching can have an impact on the results; this may be true in other tasks with small data as well.

In general, cumulative batching outperforms basic batching. This is intuitive because cumulative batching sees the same training example more times than basic batching, and overall sees the training data more times. As the number of sentences per batch increases, the differences between cumulative and basic batching shrink. Even though cumulative batching has higher performance, it takes more computational time and power than basic batching. We see a trade-off here between computational resources and performance.

It is inconclusive what the best curriculum is. For text classification, the ascending curriculum works best, while for sentence and phrase similarity, the descending curriculum works best. One hypothesis for why we see this is that for text classification, the individual words are more important than the overall structure of the sentence. The individual words are able to determine which class the sentence is a part of. Therefore, the algorithm does better when it looks at smaller sentences first, before building to larger sentences (an ascending curriculum). With the smaller sentences, the algorithm can focus more on the words and less on the overall structure. For sentence and phrase similarity, it is possible that the overall structure of the sentence is more important than the individual words because the algorithm is looking for overall similarity between two phrases. Thus, a descending curriculum, where an algorithm is exposed to longer sentences first, works better for this task.

We have explored different combinations of curriculum learning and batching strategies across three different downstream tasks. We have shown that for different tasks, different strategies are appropriate, but that overall, cumulative batching performs better than basic batching.

Since our experiments demonstrate that certain curriculum learning and batching decisions do increase the performance substantially for some tasks, for future experiments, we recommend that practitioners experiment with different strategies, particularly when the task at hand relies heavily on word embeddings.

6. Future Work

There are many tasks and NLP architectures that we have not explored in this work, and our direct results are limited to the tasks and datasets presented here. However, our work implies that curriculum learning and batching may have similar effects on other tasks and architectures. Future work is needed here.

Additionally, the three curricula that we experimented with in this paper (default, ascending, and descending) are relatively simple ways to order data; future work is needed to investigate more complex orderings. Taking into account such properties as the readability of a sentence, the difficulty level of words, and the frequency of certain part-of-speech combinations could create a better curriculum that consistently works well on a large variety of tasks. Additionally, artificially simplifying sentences (e.g., substituting simpler words or removing unnecessary clauses) at the beginning of the curriculum, and then gradually increasing the difficulty of the sentences could be helpful for "teaching" the embedding algorithm to recognize increasingly complex sentences.

There are many other word embedding algorithms (e.g., BERT [28], GloVe [29]); batching and curriculum learning may affect these algorithms differently than they affect word2vec. Different embedding dimensions and context window sizes may also make a difference. More work is needed to explore this.

Finally, our paper is an empirical study, with our observations indicating that the observed batching variation is something that researchers should consider, even though we do not explore the theoretical reasons for this.

The code used in the experiments is publicly available at https://lit.eecs.umich.edu/downloads.html (accessed on 7 September 2021).

Author Contributions: Conceptualization, L.B., J.K.K. and R.M.; methodology, L.B., J.K.K. and R.M.; software, L.B. and J.K.K.; investigation, L.B. and J.K.K.; writing—original draft preparation, L.B.; writing—review and editing, L.B., J.K.K. and R.M.; supervision, J.K.K. and R.M.; project administration, R.M.; funding acquisition, R.M. All authors have read and agreed to the published version of the manuscript.

Funding: This material is based in part upon work supported by the National Science Foundation (NSF #1344257), the Defense Advanced Research Projects Agency (DARPA) AIDA program under grant #FA8750-18-2-0019, and the Michigan Institute for Data Science (MIDAS). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF, DARPA, or MIDAS.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: *Wikipedia:* The Wikipedia dataset used to create the initial embedding spaces was used in Tsvetkov et al. [10] and is available by contacting the authors of that paper. *Text classification:* Amazon Review (pol.), Amazon Review (full), Yahoo! Answers, Yelp Review (full), Yelp Review (pol.), DBPedia, Sogou News, and AG News are available at https://course.fast.ai/datasets#nlp. (accessed on 7 September 2021). The Open Domain Deception Dataset is available at https://lit.eecs.umich.edu/downloads.html. (accessed on 7 September 2021) under “Open-Domain Deception." The Real Life Deception Dataset is available at https://lit.eecs.umich.edu/downloads.html (accessed on 7 September 2021) under “Real-life Deception”. The Personal Email Dataset is available at https://lit.eecs.umich.edu/downloads.html (accessed on 7 September 2021) under “Summarization and Keyword Extraction from Emails”. *Sentence and phrase similarity:* The Human Activity Dataset is available at https://lit.eecs.umich.edu/downloads.html (accessed on 7 September 2021) under “Human Activity Phrase Data”. The STS Benchmark is available at https://ixa2.si.ehu.es/stswiki/index.php/STSbenchmark (accessed on 7 September 2021). The SICK dataset is available at https://wiki.cimec.unitn.it/tiki-index.php?page=CLIC (accessed on 7 September 2021). *Part-of-speech tagging:* The English Universal Dependencies Corpus is available at https://universaldependencies.org/ (accessed on 7 September 2021).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G.S.; Dean, J. Distributed representations of words and phrases and their compositionality. In Proceedings of the Advances in Neural Information Processing Systems, Lake Tahoe, NA, USA, 5–10 December 2013; pp. 3111–3119.
2. Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. Efficient estimation of word representations in vector space. *arXiv* **2013**, arXiv:1301.3781.
3. Collobert, R.; Weston, J.; Bottou, L.; Karlen, M.; Kavukcuoglu, K.; Kuksa, P. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.* **2011**, *12*, 2493–2537.
4. Kenter, T.; de Rijke, M. Short Text Similarity with Word Embeddings. In Proceedings of the 24th ACM International on Conference on Information and Knowledge Management, Melbourne, Australia, 19–23 October 2015; Association for Computing Machinery: New York, NY, USA, 2015; CIKM ’15; pp. 1411–1420. [CrossRef]
5. Faruqui, M.; Dodge, J.; Jauhar, S.K.; Dyer, C.; Hovy, E.; Smith, N.A. Retrofitting Word Vectors to Semantic Lexicons. In Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Denver, CO, USA, 31 May–5 June 2015; Association for Computational Linguistics: Stroudsburg, PA, USA, 2019; pp. 1606–1615. [CrossRef]

6. Mikolov, T.; Le, Q.V.; Sutskever, I. Exploiting similarities among languages for machine translation. *arXiv* **2013**, arXiv:1309.4168.
7. Bengio, Y.; Louradour, J.; Collobert, R.; Weston, J. Curriculum learning. In Proceedings of the International Conference on Machine Learning, Montreal, QC, Canada, 14–18 June 2009; pp. 41–48.
8. Spitkovsky, V.I.; Alshawi, H.; Jurafsky, D. From Baby Steps to Leapfrog: How "Less is More" in Unsupervised Dependency Parsing. In Proceedings of the Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Los Angeles, CA, USA, 2–4 June 2010; Association for Computational Linguistics: Stroudsburg, PA, USA, 2019; pp. 751–759.
9. Smith, S.L.; Kindermans, P.J.; Ying, C.; Le, Q.V. Don't decay the learning rate, increase the batch size. In Proceedings of the International Conference on Learning Representations, Vancouver, BC, Canada, 30 April–3 May 2018.
10. Tsvetkov, Y.; Faruqui, M.; Ling, W.; MacWhinney, B.; Dyer, C. Learning the Curriculum with Bayesian Optimization for Task-Specific Word Representation Learning. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Berlin, Germany, 7–12 August 2016; pp. 130–139.
11. Xu, B.; Zhang, L.; Mao, Z.; Wang, Q.; Xie, H.; Zhang, Y. Curriculum Learning for Natural Language Understanding. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics; Association for Computational Linguistics, Online, 5–10 July 2020; pp. 6095–6104. [CrossRef]
12. Zhang, X.; Shapiro, P.; Kumar, G.; McNamee, P.; Carpuat, M.; Duh, K. Curriculum Learning for Domain Adaptation in Neural Machine Translation. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Minneapolis, MN, USA, 2–9 June 2019; Association for Computational Linguistics: Stroudsburg, PA, USA, 2019; pp. 1903–1915. [CrossRef]
13. Antoniak, M.; Mimno, D. Evaluating the Stability of Embedding-based Word Similarities. *Tran. Assoc. Comput. Linguist.* **2018**, *6*, 107–119. [CrossRef]
14. Wendlandt, L.; Kummerfeld, J.K.; Mihalcea, R. Factors Influencing the Surprising Instability of Word Embeddings. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, New Orleans, LA, USA, 1–6 June 2018; Assocation for Computational Linguistics: Stroudsburg, PA, USA, 2019; pp. 2092–2102. [CrossRef]
15. Zhang, X.; Zhao, J.; LeCun, Y. Character-level convolutional networks for text classification. In Proceedings of the Advances in Neural Information Processing Systems, Montreal, QC, Canada, 7–12 December 2015; pp. 649–657.
16. Pérez-Rosas, V.; Abouelenien, M.; Mihalcea, R.; Burzo, M. Deception detection using real-life trial data. In Proceedings of the 2015 ACM on International Conference on Multimodal Interaction, Seattle, WA, USA, 9–13 November 2015; ACM: Seattle, WA, USA, 2015; pp. 59–66.
17. Pérez-Rosas, V.; Mihalcea, R. Experiments in open domain deception detection. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, Lisbon, Portugal, 17–21 September 2015; pp. 1120–1125.
18. Loza, V.; Lahiri, S.; Mihalcea, R.; Lai, P.H. Building a Dataset for Summarization and Keyword Extraction from Emails. In Proceedings of the Ninth International Conference on Language Resources and Evaluation, Reykjavik, Iceland, 26–31 May 2014; European Languages Resources Association: Paris, France, 2014; pp. 2441–2446.
19. Joulin, A.; Grave, E.; Bojanowski, P.; Mikolov, T. Bag of Tricks for Efficient Text Classification. In Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, Valencia, Spain, 3–7 April 2017; Association for Computational Linguistics: Stroudsburg, PA, USA, 2017; pp. 427–431.
20. Wilson, S.; Mihalcea, R. Measuring Semantic Relations between Human Activities. In Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers), Asian Federation of Natural Language Processing, Taipei, Taiwan, 27 November–1 December 2017; pp. 664–673.
21. Cer, D.; Diab, M.; Agirre, E.; Lopez-Gazpio, I.; Specia, L. SemEval-2017 Task 1: Semantic Textual Similarity Multilingual and Crosslingual Focused Evaluation. In Proceedings of the 11th International Workshop on Semantic Evaluation, Vancouver, BC, Canada, 3–4 August 2017; Association for Computational Linguistics: Stroudsburg, PA, USA, 2017; pp. 1–14. [CrossRef]
22. Bentivogli, L.; Bernardi, R.; Marelli, M.; Menini, S.; Baroni, M.; Zamparelli, R. SICK through the SemEval glasses. Lesson learned from the evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. *Lang. Resour. Eval.* **2016**, *50*, 95–124. [CrossRef]
23. Spearman, C. Correlation calculated from faulty data. *Br. J. Psychol. 1904–1920* **1910**, *3*, 271–295. [CrossRef]
24. Sedgwick, P. Pearson's correlation coefficient. *Bmj* **2012**, *345*, e4483 . [CrossRef]
25. Nivre, J.; de Marneffe, M.C.; Ginter, F.; Goldberg, Y.; Hajič, J.; Manning, C.D.; McDonald, R.; Petrov, S.; Pyysalo, S.; Silveira, N.; et al. Universal Dependencies v1: A Multilingual Treebank Collection, Language Resources and Evaluation. In Proceedings of the Tenth International Conference on Language Resources and Evaluation, Portorož, Slovenia, 23–28 May 2016; European Languages Resources Association: Paris, France, 2016; pp. 1659–1666.
26. Neubig, G.; Dyer, C.; Goldberg, Y.; Matthews, A.; Ammar, W.; Anastasopoulos, A.; Ballesteros, M.; Chiang, D.; Clothiaux, D.; Cohn, T.; et al. DyNet: The Dynamic Neural Network Toolkit. *arXiv* **2017**, arXiv:1701.03980.
27. Hochreiter, S.; Schmidhuber, J. Long Short-Term Memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef] [PubMed]

28. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Minneapolis, MN, USA, 2–9 June 2019; Association for Computational Linguistics: Stroudsburg, PA, USA, 2019; pp. 4171–4186.
29. Pennington, J.; Socher, R.; Manning, C. GloVe: Global Vectors for Word Representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Doha, Qatar, 25–29 October 2014; pp. 1532–1543. [CrossRef]