Linear Models for Text Understanding

Sheldon Benard
260618386

Elias Stengel-Eskin
260609642

Galen Bryant
260602804

***Abstract*—This paper explores the use of simple classifiers as an alternative to complex deep-learning models for text understanding. We compare our results to those of Zhang and LeCun's [1] character level temporal convolutional network, and show that logistic regression, when properly tuned, yields comparable or better performance and significantly lower training times.**

1. Introduction

Text understanding consists in recovering the intended meaning from an input text in some natural language; this problem is usually split into two sub-tasks: determining the structural properties of text (e.g. word boundaries, syntactic units, etc.) and making inferences based on the hierarchical structure of the text. Any text understanding system faces a multitude of challenges, such as ambiguity and determining the ground truth for semantic meaning.

In 2015, Xiang Zhang and Yann LeCun [1] published Text Understanding from Scratch, in which they applied a temporal Convolutional Neural Network (CNN) to several text understanding problems. In their investigation, [1] found that text understanding could be handled by a deep learning system, as the temporal CNN models consistently outperformed the baseline comparison models.

In our investigation, we sought to answer: *Can models of significantly less complexity perform similarly (or outperform) complex models?* We explore the power of simple baseline classifiers in the context of text understanding, as these models typically require significantly less computation power and time, juxtaposed against the temporal CNN models in [1]. Ultimately, we show that, given a thorough tuning of the hyper-parameters and proper utilization of optimization techniques, a logistic regression baseline classifier can match the performance of, and often outperform the CNN models.

1.1. Past work

[1] detail an approach to text understanding which deviates significantly from the traditional approach to the problem, wherein a long string of text is decomposed into constituents (e.g. words, clauses, syntactic chunks, dependencies, etc.) which are then represented in some vector format, either non-sequentially (e.g. in a bag-of-words or bag-of-centroids) or sequentially (e.g. in one-hot vectors or word embeddings). Some function between this representation and the desired output class is learned – these range from simple functions (as in logistic regression) to complex ones (such as long short-term memory networks, or LSTMs). [1] eschew these structural representations by working directly with input characters (rather than words or larger constituents). Using a temporal (2-dimensional) convolutional neural network over one-hot quantized 70-dimensional character vectors, their models attempt to tackle text understanding without any explicit knowledge of lexical, syntactic, or semantic structure. [1] apply a model with six convolutional layers (with 1-dimensional max pooling) and three fully connected layers to six different datasets, attempting to model various abstract qualities of the text. These datasets are described in detail in section 1.2.

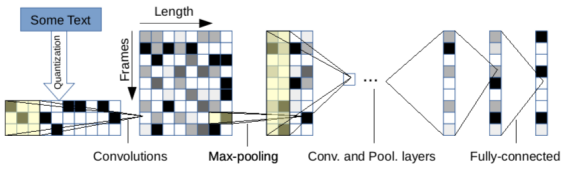

Figure 1. The CNN architecture used by [1]

In order to demonstrate the success of their model, [1] compared their model to two baseline models. The first was a logistic regression model trained on bag-of-words features. A bag-of-words model is a non-sequential vector representation of the input, where each word is encoded as a one-hot vector, and the entire sentence is encoded as the sum of the one-hot word vectors which it contains. The second was also a logistic regression model, but trained on bag-of-centroids features, where centroids were computed by running the k-means algorithm on pre-trained Google News word embeddings. Using centroids as word representations in a binary bag-of-words fashion, the representation of the sentence is a multi-dimensional vector where each dimension represents a centroid, and is assigned that centroids value if a word assigned to the cluster indexed by that centroid is present in the sentence.

1.2. Datasets

Six datasets were used in the original paper: DBpedia, AGNews, Amazon Full, Amazon Polarity, Yahoo! Answers, and the Sogou news corpus.

1.2.1. DBpedia.

DBPedia is an ontology classification corpus drawn from English Wikipedia. The DBPedia task is to classify Wikipedia articles into ontological (topical) classes based on their content. The authors chose 14 non-overlapping classes from the 2014 version of DBPedia, with 40,000 training samples and 5,000 testing samples per class, for a total of 560,000 training and 70,000 testing samples. The input to the model was the concatenation of article titles and abstracts.

1.2.2. AGNews.

This corpus is drawn from the AGs news corpus of categorized news articles. The authors chose the four largest news categories and built a categorization corpus, using article titles and descriptions as fields, with 30,000 training samples and 1,900 test samples from each class for a balanced corpus of 120,000 train and 7,600 test samples.

1.2.3. Amazon Reviews.

The authors base both AmazonFull and Amazon Polarity datasets from the Amazon corpus, produced by SNAP [2]. Drawn from product reviews on the online shopping platform Amazon, which allow customers to write a review of a purchased product, combined with a 1-to-5 scoring system (1 being the lowest score), the corpus contains review texts together with their corresponding ratings. Duplicates were removed. For Amazon Full, the prediction range was the full range of possible scores, while for polarity, reviews with 1 and 2 stars were rated as negative, those with 4 and 5 stars were rated as positive, and those with 3 stars were ignored. AmazonFull had 3,000,000 training and 650,000 testing datapoints, while AmazonPolarity had 3,600,000 training and 400,000 testing samples.

1.2.4. Yahoo.

On the Yahoo! Answers site, users ask and answer questions, which are are classified into different topic categories. The authors build a topic classification dataset using the 10 largest question categories. The corpus contains 1,400,000 training and 60,000 testing samples.

1.2.5. Sogou.

Based on the SogouCA and SogouCS corpora of Chinese news, the Sogou dataset is composed of matched pairs of a text input and news category output. The top five categories were used, and the input was formed from the concatenation of article title and content. The input was transformed into Latinized pinyin transcription format. This corpus contains 450,000 training and 60,000 testing examples.

2. Methods

The goal of this investigation was two-fold: the first aim was to replicate the baseline model results presented by [1], while the second was to improve upon these results using simple baseline models.

2.1. Models

In order to remain consistent with [1], logistic regression classifiers were implemented for both replication and improvement tracks. A logistic regression model learns a weight matrix W and bias vector b which minimize the loss (given by 2) between the true target (Y) and the predicted target $\hat{Y}$ (given by 1).

$$\hat{Y} = \sigma(W^T X + b) \tag{1}$$

$$L(w) = -y(log(\hat{y})) + (1-y)(log(1-\hat{y})) \tag{2}$$

Where $\sigma(x)$ is the sigmoid function. This model can be generalized to a multi-class case in two ways: the first is one-vs-rest classification, where, given k classes, $k-1$ classifiers are trained to distinguish between each class and all other classes. Alternatively, the logistic regression equations in 1 and 2 can be generalized to $k > 2$ classes (known as multinomial logistic regression), as seen in equations3 and 4.

$$\hat{Y} = softmax(W^T X + b) \tag{3}$$

$$L(w) = \sum_{c=1}^{M} y_{o,c} log(\hat{y}_{o,c}) \tag{4}$$

Logistic regression classifiers are linear, meaning that the decision surface they find is linear. Figure 2 shows example decision boundaries learned by logistic regression classifiers using one-vs-rest and multinomial generalization.

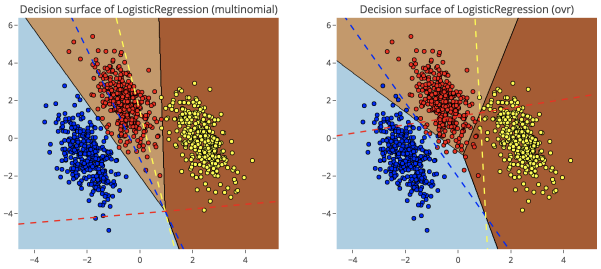

Figure 2. Decision surfaces learned by one-vs-rest and multinomial logistic regression classifiers [3]

2.2. Replication task

In order to replicate the results presented in the original paper, the same features were used. Since the paper does not reveal significant hyper-parameter settings, many of the settings for the bag of words replication models are default settings for the "Logistic Regression" classifier in the scikit-learn package. Any deviations from the default settings are detailed in Table 1.

2.2.1. Bag-of-words. In the paper, the authors constructed 5000-word vocabulary, consisting of the most frequent words for each dataset. This vocabulary was used to encode each data point as a 5000-dimensional vector. Thus, to replicate their results, we follow their approach. Our vocabulary consisted of the top 5000 most frequent words for each dataset, after the removal of stop words (words deemed irrelevant due to their relatively low semantic content).

2.2.2. Bag-of-centroids. For the word2vec model, the authors ran k-means (k = 5000) on word vectors learned from Google News corpus and used a bag of these centroids; we followed this approach for the Bag of Centroids feature selection, using minibatch k-means for speed improvements.

Feature type	Model	Solver	Max iterations
Bag-of-words	One-vs-rest	liblinear	100
	Multinomial	sag	1500
Bag-of-words	One-vs-rest	saga	100
	Multinomial	saga	100

TABLE 1. Fixed hyper-parameter settings for replication.

2.3. Improvement task

In order to improve on the baseline results presented in the original paper, feature selection options and hyper-parameter settings were thoroughly explored. The choices made in the construction of our baseline models are detailed in section 2.3.1.

2.3.1. Feature selection and pre-processing. [1] do not include any details about the data pre-processing decisions they made before the data was utilized by the classifiers, even though pre-processing has been shown to have a major impact on classification performance [4]. Thus, for each dataset, we performed several operations:

- Stop-word removal: for each data point, 1-character and 2-character word tokens were removed (ex. *Or,in,a,an,of*)
- Lowercasing: each word of the data point was converted to lowercase form; this prevents the same word generating multiple feature (*two* token is equivalent to *TWO*)
- Punctuation removal: punctuation was removed from each data point, to avoid the same word generating multiple features (*(dog)* token is equivalent to *dog*)

In addition, the following features were used:

- N-grams: bigrams have been shown to consistently improve the results on sentiment analysis tasks [5], and thus, we introduce n-gram features (with n as a hyper-parameter). In Bag of Words, unigrams (only solitary word tokens) are considered, meaning that contextual information about the data point is ignored. In our models, we explore unigrams, bigrams, and trigrams. Figure 3 highlights how the inclusion of bigrams and trigrams significantly influences the conveyed meaning.
- Skips: skip-grams are explored, as they have been shown to have value in modelling context [6]. For each of our models, either no-skip (i.e. no tokens are skipped) or 1-skips (i.e. 1 token is allowed to be skipped in the tokenization of the sentence) are utilized. The importance of skip-grams can be seen in Figure 4

Hyper-param	Values considered
Learning Rate	0.001,0.01,0.1,0.3,0.5,0.7,0.9
N-gram	1,2,3
Skip	None,1

TABLE 2. Hyper-parameter values considered for tuning

Due to the large number of features being used and the thoroughness of our hyper-parameter tuning, an extremely large number of iterations were run. In order to make training a large number of models feasible, the following optimization tricks were implemented:

- Feature Hashing: feature hashing is a method for vectorizing features, whereby the features in a data point are fed into a hash function and the hash values are directly utilized as indices. The datasets considered involve millions of unique tokens (when considering bigrams and trigrams) and keeping track of the most frequent tokens (as the authors did) is not scalable (i.e. if we want to use 100,000 most frequent tokens, we need to count the tokens, order the tokens by frequency, then keep track of the map between tokens and the indices of the feature vector). Feature hashing is fast and space-efficient, even for millions of features, and thus it is highly scalable [7].
- Online learning: to speed up training, rather than updating the parameters after considering all the data points, updates occur after each data point. Thus, loss minimization is continuous and fast; since the datasets being used contain hundreds of thousands of training points, full batch updates would be very slow.
- Early Termination: discussed in the upcoming subsection 2.3.4, early termination was utilized to prevent overfitting of the hyper parameters. Moreover, early termination returns the optimal number of passes for a given hyper parameter tuning. Utilizing the optimal number of epochs, we were able to retrain the model on the entire training set (i.e. no holdout set) and slightly boost the performance of the models.

2.3.2. Hyper-parameters. The following hyper-parameters were tuned for this experiment: learning rate, n-grams, and skips. Note that an n-gram of value of 2 includes unigrams and bigrams, and 3 includes unigrams, bigrams, and trigrams. The values we considered are outlined in Table 2.

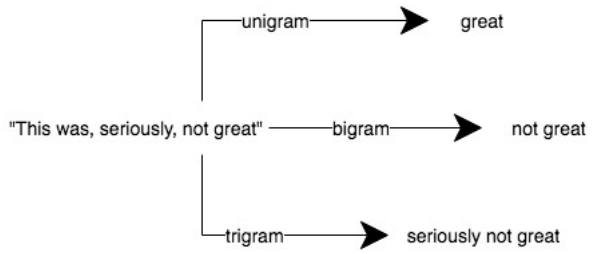

Figure 3. The importance of n-grams for conveying meaning

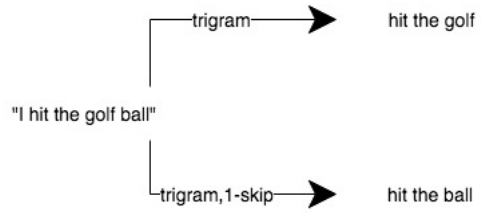

Figure 4. The importance of skip-grams for conveying meaning.

2.3.3. Validation split. For our experiment, we used a holdout of 3, meaning that, the training set was broken up into 2/3 training and 1/3 validation. For reasonably sized datasets ($n \geq 100$), this split has been shown to be near optimal [8].

2.3.4. Over-fitting. To prevent over-fitting, early termination was utilized. Thus, after each pass of the stochastic gradient descent (each epoch), the holdout validation error was calculated. If, after consecutive epochs, the validation error increased, the experiment was terminated early (to avoid over-fitting the training data).

2.4. Tools

Vowpal Wabbit (VW), an open-source out-of-core library developed by Microsoft Research [9], was utilized for this experiment. VW is scalable and adept at handling large datasets by using several techniques:

- Out-of-core online learning, to minimize memory usage
- Feature Hashing
- Parallelization

Further, VW is able to incorporate all of the considerations we wished to implement (ngrams, skips, holdout set, early termination, logistic loss function). In Table 3, we outline the settings we used.

3. Results

Table 4 shows the test accuracy (percentage) for our models replicating [1]'s bag-of-words logistic regression baseline. All experiments achieved within 3.17% of the original paper. Our results are reported in blue.

Setting	Value
Bits (used for hashing)	25
Holdout	3
Early terminate	3
Multiclass	One-vs-all
Passes	15
Loss Function	Logistic

TABLE 3. Settings used for VW

Dataset	Z+L BOW	OVR	Multi
DBpedia	96.19	97.50	97.54
AmazonFull	54.17	52.12	52.36
AmazonPolarity	89.86	88.81	88.81
Yahoo	66.62	68.04	65.48
AG	86.69	90.47	89.86
Sogou	91.35	92.98	92.21

TABLE 4. Bag of Word result comparison

Table 5 similarly shows the test accuracy (percentage) for our models replicating [1]'s bag-of-centroids logistic regression baseline. All experiments except AG achieved within 5.03% of the original paper. It was unclear why AG outperformed the original baseline by so much, but it is worth remarking that it also outperformed the BOW baseline without any tuning. Our results are reported in blue.

Table 6 shows a comparison of [1]'s best CNN and our best logistic regression model (with optimal passes), for each dataset. Best results are reported in bold. Our models outperform the best CNN-based model for four of the six datasets.

Table 7 shows the optimal hyper-parameter settings (number of n-grams, number of skips, learn rate, and optimal number of passes) for the best performing model on each corpus, along with the corresponding validation loss. Note that since AG outperformed the baseline model with default hyper-parameters, it was not included in the tuning.

Table 8 shows a comparison of the time to train the CNN models *per epoch* versus time to train the best-performing logistic regression models (all iterations). Note that Sogou was not reported by [1]. This highlights the extreme speed differences between the complex CNN-based approaches and the far simpler logistic regression baseline.

Dataset	Z+L BOC	OVR	Multi
DBpedia	89.09	93.66	94.12
AmazonFull	36.50	37.00	36.99
AmazonPolarity	72.86	73.71	73.72
Yahoo	56.47	57.50	58.27
AG	76.73	88.99	88.71

TABLE 5. Bag of Centroids result comparison

Dataset	Z+L best	Our best
DBpedia	98.40	**98.53**
AmazonFull	**59.57**	58.45
AmazonPolarity	**95.07**	94.31
Yahoo	71.10	**71.97**
AG	87.18	**90.47**
Sogou	95.12	**96.93**

TABLE 6. Best result comparison

Dataset	ngrams	skips	learn rate	opt # pass	valid loss
DBpedia	2	1	0.7	11	0.0151
AmazonFull	2	1	0.1	9	0.4190
AmazonPol	2	1	0.3	10	0.0587
Yahoo	2	0	0.1	15	0.2824
Sogou	3	0	0.9	15	0.0322

TABLE 7. Best hyper-parameter tuning and corresponding validation loss

Datasets	Large CNN	Small CNN	Ours
DBPedia	5 hours	2 hours	2m49s
AmazonFull	5 days	2 days	9m12s
AmazonPolarity	5 days	2 days	6m57s
Yahoo	1 day	8 hours	8m24s
AG	3 hours	1 hour	0m52s

TABLE 8. Execution time comparison

4. Discussion

On the whole, our experiment shows that simple models are powerful and efficient in understanding textual data; furthermore, we showed that these models benefit greatly from thorough tuning. Rigorous hyper-parameter tuning and feature selection on a sufficiently large holdout validation set ensured optimal or near-optimal settings for these considerations were found. Online learning and utilizing Vowpal Wabbit allowed us to drastically reduce the training and testing time of the model. Early termination ensured overfitting did not occur and provided the optimal number of passes for each model. Altogether, our models were quickly trained and performed similarly to the more complex models they were compared to, in many cases actually outperforming the more complex models.

However, in our methodology, we assumed that a simple model meant a linear model. Thus, one possible improvement in our methodology would be to explore non-linear classifiers (ex. SVM with a non-linear kernel) and simple neural networks, comparing the results of these different baselines with each other and the temporal Convolutional Neural Network.

The high performance of linear classifiers for this problem calls into question the utility of using CNNs for text understanding, especially in settings where word segmentations are available. Since most text content is designed to be understood by human readers, it is unclear when unsegmented text would be encountered (outside of languages which do not explicitly segment text, e.g. Mandarin). Thus word features are almost always available at a negligible computation cost (the cost of splitting the input of space characters). Additionally, while in the spatial context CNN architecture choices (especially pooling layers) have clear interpretations, the reasoning which underlies them becomes far less robust in the temporal context, when considering text – particularly with regards to the absence of pixel values.

4.1. Future Work

According to a paper released by Facebook AI Research, their tool, FastText, which utilizes character n-grams (visualized in Figure 5), is able to outperform the temporal CNN on the Amazon Full dataset and outperform our VW tuned model for Amazon Polarity [10]. Tuning of this tool may lead to better overall results, and thus, this is an interesting area of consideration.

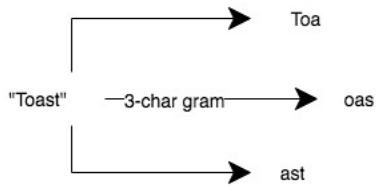

Figure 5. Character n-grams for text understanding

5. Conclusion

By tuning the learning rate, n-gram, and skip-gram hyper-parameters of an online logistic regression baseline model, we were able to match and beat [1]'s performance on several large text understanding problems.

When compared to complex temporal Convolutional Neural Network models, as proposed by [1], we were able to outperform these models in 4 of the 6 datasets (AG, Yahoo, Sogou, and DBPedia), and we were within 1.119% of the complex models for the Amazon Full dataset and 0.7625% for the Amazon Polarity dataset. Crucially, all of our models could be trained in less than 10 minutes (on a MacBook Pro 2015 laptop), as compared to several hours to several days for the temporal CNNs (utilizing 2 Tesla K40 GPUs from NVIDIA).

To answer the question: models of significantly less complexity can certainly perform similarly to complex models for text understanding, and these simple models are computationally less intensive and much faster than their complex counterparts.

6. Statement of Contributions

Elias Stengel-Eskin: Coded the Bag of Centroids replication, report and executive summary writing

Sheldon Benard: Coded the Bag of Words replication, report and executive summary writing, Vowpal Wabbit coding and execution

Galen Bryant: Research (optimization, FastText), FastText implementation investigation

We hereby state that all the work presented in this report is that of the authors.

References

[1] X. Zhang and Y. LeCun, "Text understanding from scratch," *arXiv preprint arXiv:1502.01710*, 2015.

[2] J. McAuley and J. Leskovec, "Hidden factors and hidden topics: understanding rating dimensions with review text," in *Proceedings of the 7th ACM conference on Recommender systems*. ACM, 2013, pp. 165–172.

[3] T. Dupre la Tour, "Multinomial and one-vs-rest logistic regression." [Online]. Available: https://plot.ly/scikit-learn/plot-logistic-multinomial/

[4] V. Srividhya and R. Anitha, "Evaluating preprocessing techniques in text categorization," *International journal of computer science and application*, vol. 47, no. 11, pp. 49–51, 2010.

[5] S. Wang and C. D. Manning, "Baselines and bigrams: Simple, good sentiment and topic classification," in *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*. Association for Computational Linguistics, 2012, pp. 90–94.

[6] D. Guthrie, B. Allison, W. Liu, L. Guthrie, and Y. Wilks, "A closer look at skip-gram modelling," in *Proceedings of the 5th international Conference on Language Resources and Evaluation (LREC-2006)*. sn, 2006, pp. 1–4.

[7] K. Weinberger, A. Dasgupta, J. Langford, A. Smola, and J. Attenberg, "Feature hashing for large scale multitask learning," in *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, 2009, pp. 1113–1120.

[8] K. K. Dobbin and R. M. Simon, "Optimally splitting cases for training and testing high dimensional classifiers," *BMC medical genomics*, vol. 4, no. 1, p. 31, 2011.

[9] A. Agarwal, O. Chapelle, M. Dudík, and J. Langford, "A reliable effective terascale linear learning system," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1111–1133, 2014.

[10] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, "Bag of tricks for efficient text classification," *arXiv preprint arXiv:1607.01759*, 2016.

Executive Summary: Linear Models for Text Understanding

Sheldon Benard *260618386*
Elias Stengel-Eskin *260609642*
Galen Bryant *260602804*

1. Introduction

In our investigation, Linear Models for Text Understanding, we sought to answer: *Can models of significantly less complexity perform similarly (or outperform) complex models?* We explored the power of simple baseline classifiers in the context of text understanding, as these models typically require significantly less computation power and time, juxtaposed against the temporal CNN models in [1]. Ultimately, we showed that, given a thorough tuning of the hyper-parameters and proper utilization of optimization techniques, a logistic regression baseline classifier can match the performance of, and often outperform the CNN models.

2. Methodology

[1] utilized multinomial Logistic Regression for their Bag of Words and Bag of Centroids baseline models, and thus, we chose to explore the Logistic Regression classifier (one-vs-rest and multinomial) in our investigation.

For the replication task, our feature selection procedure mimicked that of [1]. For Bag of Words, the 5000 most frequent words were utilized to encode each data point as a 5000-dimensional vector. For Bag of Centroids, centroids were computed, by running the k-means algorithm on pre-trained Google News word embeddings with k = 5000, and used to encode each data point. The paper did not specify any additional pre-processing, and thus, we did no pre-processing for this task.

For the improvement task, we focussed on improving both the quality of the features selected and the efficiency and optimization of execution.

For feature selection, we introduced pre-processing, n-grams, and skips. Pre-processing, which included punctuation and stop-word removal and lower-casing the input strings, eliminated redundancy in the tokenization of the data points. N-grams and skips allowed for structural and contextual information to be taken into consideration by the model.

To increase efficiency and optimization, feature hashing, online learning, and early termination were explored. Feature hashing is a scalable solution to increasing the quantity of features which we used to expand the total number of features that our model could explore. Online learning ensured that gradient descent updates would take significantly less time, as single data points were used rather than the whole dataset. Lastly, early termination prevented overfitting and provided us with the optimal number of passes to use.

Vowpal Wabbit, of Microsoft Research [2], was utilized to combine all these considerations in one solution.

3. Results

Table 1 shows the test accuracy (percentage) for our models replicating [1]'s bag-of-words logistic regression baseline. All experiments achieved within 3.17% of the original paper. Our results are reported in blue.

Dataset	Z+L BOW	OVR	Multi
DBpedia	96.19	97.50	97.54
AmazonFull	54.17	52.12	52.36
AmazonPolarity	89.86	88.81	88.81
Yahoo	66.62	68.04	65.48
AG	86.69	90.47	89.86
Sogou	91.35	92.98	92.21

TABLE 1.

Dataset	Z+L BOC	OVR	Multi
DBpedia	89.09	93.66	94.12
AmazonFull	36.50	37.00	36.99
AmazonPolarity	72.86	73.71	73.72
Yahoo	56.47	57.50	58.27
AG	76.73	88.99	88.71

TABLE 2.

Table 2 similarly shows the test accuracy (percentage) for our models replicating [1]'s bag-of-centroids logistic regression baseline. All experiments except AG achieved within 5.03% of the original paper. It was unclear why AG outperformed the original baseline by so much, but it is worth remarking that it also outperformed the BOW baseline without any tuning. Our results are reported in blue.

Table 3 shows a comparison of [1]'s best CNN and our best logistic regression model (with optimal passes), for each dataset. Best results are reported in bold. Our models outperform the best CNN-based model for four of the six datasets.

4. Conclusion

By tuning the hyper-parameters of an online logistic regression baseline model, we were able to match and beat [1]'s performance. We were able to outperform temporal CNNs in 4 of the 6 datasets, and we were within 1.119% and 0.7625% for those that we did not beat. Crucially, all of our models could be trained in under 10 minutes (on a laptop), versus several hours to several days for the temporal CNNs (utilizing Tesla K40 GPUs). We find that models of significantly less complexity can outperform complex models for text understanding at a much smaller computational cost.

Dataset	Z+L best	Our best
DBpedia	98.40	**98.53**
AmazonFull	**59.57**	58.45
AmazonPolarity	**95.07**	94.31
Yahoo	71.10	**71.97**
AG	87.18	**90.47**
Sogou	95.12	**96.93**

TABLE 3.

References

[1] X. Zhang and Y. LeCun, "Text understanding from scratch," *arXiv preprint arXiv:1502.01710*, 2015.

[2] A. Agarwal, O. Chapelle, M. Dudík, and J. Langford, "A reliable effective terascale linear learning system," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1111–1133, 2014.