Building a robust Text Classifier on a Test-Time Budget

Md Rizwan Parvez  
University of California Los Angeles  
rizwan@cs.ucla.edu

Tolga Bolukbasi  
Boston University  
tolgab@bu.edu

Kai-Wei Chang  
University of California Los Angeles  
kwchang@cs.ucla.edu

Venkatesh Saligrama  
Boston University  
srv@bu.edu

Abstract
In this paper, we study a generic learning framework for building robust text classification model that achieves accuracy comparable to standard full models under test-time budget constraints. Our approach learns a selector to identify words that are relevant to the prediction tasks and only passes these words to the classifier for processing. The selector is trained jointly with the classifier and directly learns to incorporate with the classifier. We further propose a data aggregation scheme to improve the robustness of the classifier. Our learning framework is general and can be incorporated with any type of text classification model. On real-world data, we show that the proposed approach improves the performance of a given classifier and speeds up the model with a mere loss in accuracy performance.

Introduction
Recent advances in machine learning approaches have improved the accuracy performance of natural language processing tasks such as document classification, question answering, and sentiment analysis (Wu et al. 2017; Seo et al. 2016; Socher et al. 2011; Yu, Lee, and Le 2017). These approaches process the entire text and construct representations of words and phrases in order to perform the target task. While these models do realize high accuracy, the computation-time scales linearly with the size of the documents. Consequently, the inference process can be slow for documents consisting of several sentences.

Motivated by these issues, we focus on the problem of fast test-time prediction for long full text documents. We draw our inspiration from the work of (Lei, Barzilay, and Jaakkola 2016), who show that, on long documents, there is little degradation in prediction performance, even when the model predictions are primarily based on a suitably selected small subset of words in the document. In their method, Lei, Barzilay, and Jaakkola propose a selector to find snippets of input-text to serve as justification (rationales), which is then input into a neural network based classifier, which then outputs a decision. They propose to jointly train the selector and classifier modules. In their proposed embodiment, both of these modules have similar complexity and in turn, require similar processing times during run-time.

Analogously, our proposed approach also consists of a selector and a classifier. The selector identifies important words and eliminates the rest from the document. Therefore, it is obvious that the selector output is a list of important words (i.e., grammatically incorrect sentences with missing words). The selector output is passed as input to the classifier, which then performs the target task on the reduced text (See Figure 1). Although our method can be used in conjunction with any suitable selector, we find that it is more economical (i.e., faster) to use a selector based on simple word-embeddings. In contrast to (Lei, Barzilay, and Jaakkola 2016), our proposed selector is computationally inexpensive. Furthermore, we show that a classifier trained on words selected by the word-embedding-based-selector can outperform other conventional neural network based selectors leading to significant overall speedup with little loss in accuracy.

While we demonstrate speedup gains on word-embedding based selector, our proposed objective is to ensure that our classifier can seamlessly work with other conventional selectors. Our objective leads to a fundamental issue, in that a naively trained classifier on long texts with no missing words, is incompatible with inputs received by the classifier during the test phase. We mitigate this effect during training through blanking out input-text and sequentially aggregating data samples of blanked-out text. Our proposed blank-out process blanks-out text in various forms ranging from word to sentence level blank-outs. Our proposed approach
is related to DAGGER (Ross, Gordon, and Bagnell 2010), which sequentially augments data trajectories in a different reinforcement learning context during training to mitigate the discrepancy between training and testing phases. In summary, our contributions are two-folds.

1. We propose a modular data aggregation framework for training classifiers that can be deployed in conjunction with any suitable selector.
2. We propose a word-embedding based selector, which is computationally inexpensive, and when utilized in conjunction with our trained classifier leads to significant speedup with little degradation in accuracy.

Our framework makes the baseline classifier robust to the missing words, grammar errors in the input text and enables to capture the necessary features from it. A relatively small fraction of text (i.e., selector output) is enough to achieve nearly the baseline classifier performance. Our framework achieves even better accuracy than the baseline classifier while the selector selects all the text (i.e., no selector). Hence being robust, even with a simple and fast selector which does not select the important words very precisely, the classifier trained with our framework can predict much faster with a negligible loss in accuracy performance in compare to the original baseline.

Related Work

**Fast Reading Text:** Neural networks used in many real-life applications, such as reading text, are often computationally expensive. Many recent works have focused on speeding up computation at test-time. Wu et al. proposes a CNN based approach and demonstrates a 2X speedup for question answering. Choi et al. proposes a GPU efficient CNN classifier for question answering. Learning to skim method (Yu, Lee, and Le 2017) and skim-RNN (Seo et al. 2017) are of particular relevance to our paper are two recent methods. Learning to skim method learns to completely skip words deemed to be irrelevant and a variant skim-RNN that skims words rather than skipping and suitably updates hidden state in an RNN based on the words importance and achieves the state-of-art speedup performance. Both of these two approaches use LSTM classifier model and cannot be incorporated with any other state-of-art text classifier model. In contrast, ours is generic and classifier invariant. We can leverage the state-of-art text classifiers (e.g., BCN classifier which provides a strong baseline performance on SST-5 and SNLI (McCann et al. 2017a)).

In contrast, while neither of these methods are interpretable by a human, we propose an interpretable modular approach for reading text with impressive speedup gains.

**Interpretable Architecture:** Lei, Barzilay, and Jaakkola develops a selector to find words in the text as justifications (rationales) for the decision of a neural network. While our approach is similar in concept, their approach is impractical from a computational perspective. In particular, they train a complex selector, which is as complex as the classifier leading to significant increase in overall computation time.

**Budgeted Learning:** The trade-off between computational cost and accuracy has drawn a considerable interest in recent years. In standard multi-class classification case (Viola and Jones 2001; Karayev, Fritz, and Darrell 2013; Xu et al. 2013; Trapeznikov and Saligrama 2013; Kusner et al. 2014; Wang et al. 2014) propose frameworks that are cost-aware. (Strubell et al. 2015; Weiss and Taskar 2013; He, Daumé III, and Eisner 2013; Bolukbasi et al. 2017a) focus on different instances of structured prediction and sparsity input features or eliminate unnecessary edge computations for graphical models to achieve better computation. Finally, (Bengio et al. 2015; Leroux et al. 2017; Lin et al. 2017; Bolukbasi et al. 2017b) build on DNN architectures to allow avoiding costly computations.

**Feature Selection in Text Classification:** Removing stop words in text classification has been considered as a standard process in text classification. However, these stop words are often predefined based on word frequencies and are not learned along with the targeted task. Various feature selection approaches (Chandrashekar and Sahin 2014) have been discussed in the literature. The most relevant one is to employ lasso (Tibshirani 1996) or group lasso (Faruqui et al. 2015) for learning sparse features. Different from these approaches, we directly learn a feature selector along with the classifier. Our selector chooses salient words of an instance (long sentence). These words serve as input to a classifier (e.g., LSTM, PCA or dimension reduction methods map an instance (long sentence) into low dimension space but this representation is not aligned with required LSTM input. Logistic Regression model over unigrams and bigrams with heavy feature selection or L1-regularized classifiers (Zou and Hastie 2005; Ng 2004; Yuan et al. 2010) are reasonable methods but the drawback is that it does not allow generalization to novel words/phrases at test-time which are account for by word embeddings.

A Data Aggregation Framework

In the following, we present a data aggregation framework to train a robust classification model that performs well under different test-time budgets. The approach is general and can be incorporated with various types of text classification model.

**Motivation and Overview:**

The motivation behinds the data aggregation framework is that not all words contribute equally to the targeted classification task. For example, to predict the sentiment of the example: “We ordered the beef kabob. It was very delicious. All of us really enjoyed it.”. Words such as “delicious” and “enjoyed” are strong indicators for classifying the sentence as a positive review, while words such as “We” and “order” are less important as they appear often. By filtering out non-important words, we can reduce unnecessary computation as well as make the model robust and interpretable.

Inspired by (Lei, Barzilay, and Jaakkola 2016), we design a selector to decide if a word is worthwhile or not for passing to the classifier. The selector is trained along with the classifier; therefore, it learns to pick words to optimize the performance of the classifier while minimizing the number of...
words it selects. In contrast to (Lei, Barzilay, and Jaakkola 2016), we consider a simple selector such that the overall test-time can be reduced. However, a simple selector may make mistakes when selecting words to pass, causing significant performance drops of the classifier. To overcome this, we introduce a data aggregation approaches for learning a robust classifier.

**Learning a Selector:**

We train the classifier module to perform the task of reading text on outputs received from any suitable selector and as described above, the function of a selector is to eliminate words that are deemed to be unimportant. This results in a scheme wherein the test-time classifier cost scales in proportion to the size of the reduced text. Note that, there is no supervision for training the selector rather we train the selector jointly. We extend the parameter controlled joint training of (Lei, Barzilay, and Jaakkola 2016) for cross entropy loss function in order to incorporate any state-of-art classifier. In this section, we summarize the joint training.

Formally, let us denote a training instance \((x, y)\) where the input sentence \(x = \{x_1, x_2, \ldots, x_n\}\) and the true target vector is \(y\). We denote the corresponding selector output by \(s(x) = \{s_1, s_2, \ldots, s_n\}\) where \(s_t \in \{0,1\}\). We assume that the rationales (i.e., important words) can be identified independently by the word embeddings. Hence,

\[
P(s(x)|x) = \prod_{t=1}^{n} P(z_t|x_t)
\]

To control/regularize the selection budget (i.e., what fraction of the given text should be selected) and to enforce the selector to select only a few words which are meaningful phrase (i.e., consecutive selection instead of scattered ones), two parameters \((\lambda_1, \lambda_2)\) are introduced:

\[
\phi(s(x)) = \lambda_1||s(x)|| + \lambda_2 \sum_{t=1}^{n} |s_t - s_{t-1}|
\]

where the first term is for sparsity and the second term is for coherent (i.e., continuous selection).

The words selected by the selector are input into the classifier. We denote the selector output, classifier input, classifier output by \(s(x), (s(x), x)\) and \(c(s(x), x)\) respectively. For the cross entropy loss,

\[
\ell(c(s(x), x), y) = -P(y) \log(P'(y))
\]

where \(P(y)\) is the true probability and \(P'(y)\) is the predicted probability.

The final objective function

\[
\min_{\theta_s, \theta_c} \mathbb{E}_{x, y \sim X, Y} [\ell(c(s(x), x), y) + \phi(s(x))]
\]

is minimized by the doubly stochastic gradient described in (Lei, Barzilay, and Jaakkola 2016).

---

**Algorithm 1: Data Aggregated Training Schema**

**Input:** Training corpus \(X\), budget level set \(B\), classes of selectors \(\{S_1\}\), classifier class \(C\)

**Output:** Trained selector, and classifier

1. \(C_1 \leftarrow \text{Train a classifier } C \text{ on } X\)
2. \(\text{Initialize aggregated corpus: } D \leftarrow X\)
3. for \(s_i \in \{S_i\}\) do
   4. for \(b \in B\) do
      5. Warm-start a classifier \(C\) with model parameters in \(C_1\)
      6. Jointly train \(s_i\) and \(C\) with budget level \(b\) on \(X\)
      7. Generate dataset \(D_{i,b}\) by applying \(s_i\) on \(X\)
      8. under budget level \(b\)
      9. Aggregate datasets: \(D \leftarrow D \cup D_{i,b}\)
   10. end
11. \(C_2 \leftarrow \text{Train another classifier } C \text{ on aggregated data } D\)
12. \(S_w, C_w \leftarrow \text{Train } S_w \text{ jointly with } C_2 \text{ on } X\)
13. return \(S_w, C_w\)

---

**Word Embedding (WE) selector:** To achieve overall speedup gains, we must have a selector that strikes a balance between informative word-selection and computational efficiency. A parsimonious word-selector could be efficient during the selection phase and improve the overall test-time. Based on this design principle, we propose a word-embedding-based selector (WE) that takes the word embedding (Pennington, Socher, and Manning 2014) as features and train a logistic regression classifier to decide if the corresponding word should be passed to the classifier. Intuitively, words with a similar meaning to important words are more likely to be beneficial to the classification task. Therefore, word embedding provides sufficient information to decide a word should be kept or not.

Formally, the WE selector outputs a binary vector \(z\), such that \(z_t\) associated with word \(x_t\) is

\[
P(z_t|x_t) \propto \text{sigmoid}(\theta^T \vec{x}_t) = \frac{1}{1 + \exp(-\theta^T \vec{x}_t)}
\]

where \(\vec{x}_t\) is the word embedding of the word \(x_t\), and \(\theta\) is the model parameters to be learned.

**Data Aggregation:**

Next, we discuss the data aggregation scheme for improving the training of the classifier.

As the selector impacts the distribution seen by the classifier, if the classifier is not robust enough to handle the variants of the input, it performs poorly in the test-time. Our preliminary experiments show that it is especially true when the classification model is complex (e.g., based on an LSTM model) and implicitly takes high-order combinations of words as features. We cannot use an off-the-shelf full-text classifier. Because at test-time the classifier sees selector outputs, which are sentences with missing words. Consequently, we must train an LSTM classifier to replicate
| Dataset       | #class | Classification Task | Vocabulary | Size (Train/Dev/Test)       | Avg. Len |
|--------------|--------|---------------------|------------|-----------------------------|---------|
| SST          | 2      | Sentiment Analysis  | 13,750     | 6,920 / 872 / 1,821         | 19      |
| IMDB         | 2      | Sentiment Analysis  | 61,046     | 21,143 / 3,857 / 25,000    | 240     |
| AGNews       | 4      | News Classification| 60,088     | 101,851 / 18,149 / 7,600   | 43      |
| Yelp         | 5      | Sentiment Analysis  | 1,001,485  | 600k / 50k / 50k            | 149     |
| Multi Aspect | 10     | Sentiment Analysis  | 147,761    | 51,675 / 1,000 / 1,000     | 144     |

Table 1: Statistics of the datasets we evaluate our framework on. Multi-Aspect refers to (Lei, Barzilay, and Jaakkola 2016), and SST refers to Stanford Sentiment Treebank.

Figure 2: The performance versus the fraction of words selected on Multi-Aspect and IMDB datasets. We present results using RCNN and LSTM classifier. Results demonstrate that with the data aggregation framework (SAG/WAG), a simple WE selector is competitive with a complex RCNN selector.

the scenario seen during run-time. On the other hand, if we train the classifier only on the selector outputs, it tends to overfit the training data and fails to generalize. This situation is particularly severe when the test-time budget is low, where the selector is too aggressive in removing words in text and there are no enough features to train a reasonably good model.

A similar issue has been observed and discussed in the reinforcement learning and imitation learning literature. When the search space is large, the model may end up in a different trajectory from the one it seen in the training phase, resulting in the learned model fails terribly. Inspired by the DAGGER algorithm (Ross, Gordon, and Bagnell 2010) proposed for solving this issue, we propose the following data aggregation approach for training a robust classifier. DAGGER is iterative; at each iteration it updates its policy by training a classifier. Experts then provide new data (visited states and actions) based on the updated policy. This data is aggregated with the existing data trajectory. In contrast, our blank-out datasets are found from the original train data and we aggregate these datasets only once. We do not use the trained classifier iteratively for data aggregation rather on the aggregated dataset we train the classifier.

In our approach, we aggregate the training data from different cost regions and different class of selector and train a single classifier on the aggregation. By aggregating the data form different settings, the classifier learns to be robust to against missing words and the discrepancy between training and test sets is reduced.

We consider the following three schemes to select segments of words in different granularity.

Sentence Aggregation (SAG): For document classification tasks, where each instance involves multiple sentences, we augment the data set by blanking out some sentences in each of the original training instances and keeping the rest of the sentences without modification. This blank out mimics the output from a selector which selects rationales consecutively as a sentence.

Phrase Aggregation (PAG): In this scheme, for each sentence, we augment it by a data set by blanking out some phrases in the original training sentence and keeping the rest unchanged. This blank out mimics the output from a selector which selects rationales at phrase granularity.

Word Aggregation (WAG): In this scheme, we augment a data set by blanking out words. This blank out mimics the output from a selector which selects sparse word-level rationales. We can utilize a random sampling scheme, whereby, we blank out words/sentence randomly (random blank out schema) or use pre-trained selector’s output (selector blank out schema).

The data aggregation training scheme is summarized in Algorithm 1. In step 1, we first train a classifier on the original training set. Then, in step 2, we train several versions of selectors with different classes of architecture and budget levels jointly with the classifier initializing with the pre-trained classifier from step 1 which gives the model a warm-start. As mentioned in Section , the percentage of the text selected by the selector can be controlled, the selectors produce the selected (e.g., reduced) training data from different
Figure 3: The performance versus test-time on Multi-Aspect and IMDB datasets. We present results using RCNN and LSTM classifier. Results demonstrate that RCNN selector is way slower than WE and our data aggregation framework (SAG/WAG) is even faster than WE. With them the proposed simple WE selector achieves better performance given the same test-time budget.

cost regions (i.e, percentage of text) from only a small fraction of text to almost no reduction. Any type of selectors (e.g., RCNN, LSTM, WE, BCN) and any blanking out (e.g., SAG, PAG, WAG) style can work.

In step 3, we aggregate the training data using the pre-trained selectors from step 2 and train the classifier. Our aggregated data has different copies of the same instance with different amount (i.e, percentage) of selected text. Therefore, the classifier trained on this aggregated datasets, sees all the different budgeted text distribution in training time. As a result, it becomes robust and the full text performance (i.e, classifier without any selector) improves. In step 4, we train our WE selector jointly with this improved classifier. As the classifier has already achieved some performance improvement, the simple WE selector performs sufficiently well. As now, with this WE selector, we can identify the rationales with competitive precision in no time, the overall text classification speedup well enough in compare to the existing baselines.

Experiments
In this section, we evaluate the proposed approaches on real-world text classification datasets. Results from quantitative and qualitative analyses demonstrate that the proposed framework enables us to learn a robust text classifier under test budget constraints that is given a time limit (e.g., 10 sec), we achieve the best accuracy. To validate our claim, we first show that our framework makes any state-of-art classifier robust and hence despite the complex RCNN selector performs well in identifying important words (Lei, Barzilay, and Jaakkola 2016), working with simple WE selector can achieve better accuracy. We also show that any state-of-art classifier trained by our framework can even outperform its standard performance. We then show that in addition to better accuracy our framework (WE selector) is way faster than RCNN selector. Next, we compare our framework with existing budget-learning (i.e., speedup) frameworks. Finally, we analyze our framework qualitatively and also in terms of the network latency/memory.

Experimental Setup
We consider the following five datasets in the experiments. The statistics of the datasets are summarized in Table 1. By default, we use the WE selector with budget set $B = \{0.5, 0.6, \ldots, 1.0\}$ (See Algorithm 1). WAG selection scheme, Glove (Pennington, Socher, and Manning 2014) word embeddings for our framework and evaluate in terms of accuracy/error (error = 1 - accuracy) metric unless stated otherwise.

SST-2: SST-2 refers to binary classification problem of Stanford Sentiment Treebank (Socher et al. 2013). For each
Model | SST-2 | IMDB | AGNews | Yelp
--- | --- | --- | --- | ---
| Acc(%) | speedup | Acc (%) | speedup | Acc (%) | speedup | Acc (%) | speedup
Logistic regression (L1) | 82.4 | 3.3x | 86.6 | 2.5x | - | - | - | -
LSTM-jump | - | - | 89.4 | 1.6x | 89.3 | 1.1x | - | -
skim-RNN | 86.4 | 1.7x | 91.2 | 2.3x | 93.6 | 1.0x | - | -
classifier | LSTM | 86.4 | 1.3x | 88.1 | 1.2x | 90.0 | 1.3x | 62.1 | 1.2x
| Standard | 85.2 | 1x | 87.1 | 1x | 92.8 | 1x | 66.7 | 1x
| Our framework | 86.4 | 1.3x | 88.1 | 1.2x | 90.0 | 1.3x | 62.1 | 1.2x
classifier | BCN | 88.3 | 1x | 92.1 | 1x | 93.2 | 1x | 66.3 | 1x
| Standard | 85.7 | 1x | 91.0 | 1x | 92.3 | 1x | 65.8 | 1x
| Our framework | 88.3 | 1x | 92.1 | 1x | 93.2 | 1x | 66.3 | 1x

Table 2: The test performance and the overall speed-up. Without specifying, our framework uses the WAG strategy. Results with *, and $ are using PAG, and SAG schemes, respectively. All results are the average of 3 runs. For each classifier, we present two rows of results. First one (top row) denotes the best speedup performance and the second one (bottom row) denotes the best accuracy achieved by our framework without any speedup.

| Model | Multi-Aspect | IMDB |
| --- | --- | --- |
| classifier | MSE speedup | Acc (%) speedup |
| LSTM Baseline | 0.01250 | 1x | 87.1 | 1x |
| Our framework | 0.01195 | 2.5x | 88.1 | 1.2x |
| classifier | BCN |
| RCNN Baseline | 0.01083 | 1x | 87.1 | 1x |
| Our framework | 0.01188 | 2.5x | 88.1 | 1.2x |

Table 3: Experimental results of our framework on Multi-Aspect and IMDB datasets with LSTM and RCNN classifier. The best result in each section is boldfaced. The speedup considers full pipelined test-time (selector + classifier). We compare our framework with the LSTM and RCNN baseline of (Lei, Barzilay, and Jaakkola 2016).

sentence, it annotates sentiment labels for the entire sentence and for the phrases in the parse trees. As each instance consists of only one sentence, we perform experiments using WAG and PAG schemes.

**IMDB:** IMDB refers to (Maas et al. 2011). Each IMDB instance is a paragraph which consists of a number of sentences. Therefore, we perform experiments with both of SAG, and WAG scheme.

**Multi-Aspect:** Multi-Aspect refers to (Lei, Barzilay, and Jaakkola 2016). For this dataset, we use word embeddings provided with the dataset and while aggregating the data we use both RCNN and WE selector using both SAG, and WAG with $B = \{0.1, 0.2, \ldots, 1.0\}$. To compare our model we follow (Lei, Barzilay, and Jaakkola 2016) and use evaluation metric mean square error (MSE) for this dataset.

**AGNews:** We collect the dataset from the publicly available repository of (Zhang, Zhao, and LeCun 2015). Each instance has a title and a small paragraph.

**Yelp:** Yelp dataset refers to (Conneau et al. 2016). Each instance is a short paragraph. We perform experiments with LSTM, BCN classifier and both of SAG, and WAG aggregation scheme.

We consider the following neural network architectures for the selector and the classifier in our framework:

**Recurrent Convolution Neural Network (RCNN):** RCNN (Lei, Barzilay, and Jaakkola 2016) is a refined local n-gram convolutional neural network model. The recurrent part learns the average features in a dynamic fashion and the convolution part learns the n-gram features that are not necessarily contiguous. We explore RCNN model both as the selector and the classifier.

**LSTM:** LSTM is widely used for text processing (Zhang, Zhao, and LeCun 2015; Seo et al. 2016; 2017; Yu, Lee, and Le 2017). LSTM sequentially reads words in a passage and updates its hidden state to captures features from the text.

**Biattentive Classification Network (BCN):** BCN is a generic text classification model (McCann et al. 2017a). It comprises of biLSTM, Biattention, and Maxout networks. BCN provides a strong baseline on many datasets, such as SQuAD, Stanford Sentiment tree (SST), TREC, IMDB, and SNLI datasets (McCann et al. 2017b).

---

We leverage the sentiment labels for phrases but do not use the parsing tree information.

3https://github.com/mhjabreel/CharCNN/tree/master/data/ag_news_csv
We use the Allennlp implementation\(^4\) of BCN in our experiments.

**Word Embedding (WE) selector:** See Section “A Data Aggregation Framework”.

### Robustness of Our Proposed Framework

In this section, we show that our framework makes any state-of-art classifier robust for which, with a WE selector it can achieve better accuracy than with a complex RCNN selector. We also show that the state-of-art classifier trained by our framework can even achieve better accuracy than its standard full text (i.e., non-important words are also passed) performance. Given an RCNN or an LSTM classifier, we consider the following four settings:

1. The classifier is trained along with the proposed WE selector.
2. The classifier is trained along with a complex RCNN selector. When the classifier is the RCNN model, this setting is the same as (Lei, Barzilay, and Jaakkola 2016).
3. Similar to the setting 1, but the classifier is trained using the sentence-level aggregation (SAG) strategy.
4. Similar to the setting 1, but the classifier is trained using the word-level aggregation (WAG) strategy.

Figure 2 demonstrates the trade-off between the performance and the fraction of words selected by each setting. Overall, the error increases when the fraction of the text selected is lower. On the Multi-Aspect dataset (see Figure 2(a)), the performance of the proposed WE selector is competitive with the complex RCNN selector. When training the classifier with word-level data aggregation strategy, the model further improves and requires only 12% of selected text to achieve error rate different from predicting on full text by merely 0.1%. Similarly, the WE selector and its variant perform well when the classifier is an LSTM model (see Figure 2(b)). On the IMDB data (see Figure 2(c)), WE selector has similar performance trade-off as the RCNN selector and further confirms that a simple selector is sufficient for identifying rationales. With sentence-level data aggregation, the model performs the best and achieves lower error rate than the baseline RCNN model.

From Table 2, on SST-2, IMDB, AGNews dataset, the standard LSTM, BCN classifier (without any selector) has the accuracy of 85.2, 87.1, 92.8, and 85.7, 91.0, 92.3 respectively. Corresponding LSTM and BCN trained with our framework can achieve the accuracy of 87.4, 88.1, 92.9 and 88.3, 92.1, 93.2 respectively. On Yelp dataset, our framework improves the performance of BCN classifier from 65.8 to 66.3 which is even better than the state-of-art performance of 64.72 (Conneau et al. 2016).

#### Performance vs. Test Time

We further report the performance versus test running time in Figure 3. Despite the RCNN selector performs well in identifying important words, its complexity is too high. Therefore, the overall test-time is almost double in all the cases (see Figure 3(a), 3(d)). In Figure 3(b), 3(c), and Figure 3(e), 3(f), we show that our framework (SAG/WAG) not only significantly outperforms but also is even more faster than WE selector. Table 3 summarizes the performance and test speed compared with the baseline models. Our approach accelerates both the LSTM and the RCNN models by 2.5 and 1.2 times with competitive or even better performance on Multi-Aspect and IMDB, respectively.

Finally, we also compare our approach to the baseline that accelerates a model by filtering out stop-words.\(^5\) This approach can speed up the RCNN model by 2 times, but the performance drops from 0.11 to 0.17 in mean square error (MSE). The performance drop is due to some stop-words carry important information for the targeted classification task. For example, the stop-words “but”, and “not” play a very significant role in determining the polarity of the full sentence. On the other hand, some words such as “ordered”, “beef”, “kabob” are not important for sentiment analysis task even if these words are not stop-words. Therefore, to achieve overall speedup gains, we must have a selector that strikes a balance between informative word-selection and computational efficiency.

### Comparisons with other Budget Learning Frameworks

Our framework is generic and can be incorporated with any text classification model. We demonstrate the performance of proposed framework with two widely used text classification models LSTM, and BCN in Table 2. We also compare our model with LSTM-jump (Yü, Lee, and Le 2017) and Skip-RNN (Seo et al. 2017) approaches that are designed for accelerating the LSTM model.\(^6\)

Results show that the data aggregation approach improves the baseline LSTM and BCN models when the selector picks all the words. For example, the accuracy of the BCN classifier improves 2.6%, 1.1%, 0.9%, 0.5% on SST-2, IMDB, AGNews, Yelp datasets, respectively.

Skip-RNN performs better than LSTM-jump on IMDB and AGNews datasets. Despite skim-RNN performs well in some cases, it is unstable and is hard to control the trade-off between performance and the test-time budget. For example, the baseline LSTM achieves 93.5% on AGNews dataset. Skip-RNN is slower than the baseline method with significantly accuracy drops. Also, on the IMDB dataset, skim-RNN achieves an accuracy of 91.2 with 2.3x speedup but for a lower accuracy of 88.7, the speedup is worse (1.5x).

In contrast to Skim-RNN and LSTM-jump, our approach is generic and is not designed for LSTM. However, our model achieves competitive performance to these two approaches when using the LSTM model as the classifier. By leveraging the BCN model, our model achieves better test performance on SST and IMDB and competitive performance on AGNews.

---

\(^4\)https://github.com/allenai/allennlp/blob/master/allennlp/models/biattentive_classification_network.py

\(^5\)The word list is from the NLTK library.

\(^6\)Our results are representative. We attempted to replicate the exact experimental setup provided in (Yu, Lee, and Le 2017; Seo et al. 2017) except could not get the exact randomization split.
Japanese nuclear plant searched. Kansai Electric Power #39;s nuclear power plant in Fukui, Japan, was searched by police Saturday during an investigation into an Aug. 9 mishap.

Sports

Shane Warne takes six but India establish handy lead (Reuters). Reuters- World test wicket record holder Shane Warne grabbed six wickets as India established a handy 141-run first innings lead in the second test on Saturday.

Sci/Tech

Handset Makers Raising Virus Defenses (Reuters). Reuters - Software security companies and handset makers, including Finland’s Nokia (NOK1V.HE), are gearing up to launch products intended to secure cell phones from variants of the Internet viruses that have become a scourge for personal computer users.

Table 4: Examples of the selector output on AGNews. Bold words are selected by the selector, while the remainders are filtered out. Although words like “during an” seem non-important in these examples, appearing in phrases like “bomb exploded during an Independence Day parade” (World-News) and “undefeated during an entire season” (Sports-News), provide a hint for the model to understand the sentences.

Latency Analysis

In contrast to skim-RNN and LSTM-Jump that sequentially visit the words in passage. Our model design allows the WE selector to process words in passage in parallel. In practice, as the computation involves in the WE selector is simple, the running time of the selector can be negligible. For example, the WE selector takes overall only 14s seconds to identify important words on the Yelp dataset, and the LSTM models take up to 316.5 seconds to process the selected words.

The benefit is more obvious when the text classification model is employed in cloud computing setting. The local devices (e.g., smart watches or mobile phones) do not have enough memory and computational power to execute a complex classifier. Therefore, the test instance has to be sent to a cloud server and classified by the model on the cloud. In this setting, our approach can employ the selector in the local device, and send only important words to the cloud server. In contrast, skim-RNN and LSTM-jump, which process the text in a sequential nature have either to send the entire text to the server or require multiple rounds of communication between server and local devices. In either case, the network latency and bandwidth may restrict the speed of the classification framework. For WE selector, selection depends only on the embedding. Therefore, instead, we can cache the predictions and store only a list of important words.

Qualitative Analysis

One advantage of the proposed framework is that the output of the selector is interpretable. In Table 4, we present three examples from the AGNews dataset. Results demonstrate that our framework correctly identifies words such as “Nokia”, “nuclear”, “plant”, “Shane Warne”, “software” and phrases such as “searched by police”, “takes six but India establish handy lead” are important to the document classification task. It also learns to filter out words (e.g., “Aug.”, “products”, “users”) that are less predictive to the classification labels.

Conclusion

In this paper, we proposed a budgeted learning framework for learning a robust classifier under test-time budget constraints. We demonstrate that the proposed WE selector effectively selects important words for classifier to process and the data aggregation strategy further improve the model performance. The future work includes applying the proposed framework for other text reading tasks and improving the data aggregation strategy by applying learning to search approaches (Chang et al. 2015).

References

Bengio, E.; Bacon, P.-L.; Pineau, J.; and Precup, D. 2015. Conditional computation in neural networks for faster models. arXiv preprint arXiv:1511.06297.

Bolukbasi, T.; Chang, K.-W.; Wang, J.; and Saligrama, V. 2017a. Resource constrained structured prediction. In AAAI, 1756–1762.

Bolukbasi, T.; Wang, J.; Dekel, O.; and Saligrama, V. 2017b. Adaptive neural networks for efficient inference. In Precup, D., and Teh, Y. W., eds., Proceedings of the 34th International Conference on Machine Learning, volume 70 of Proceedings of Machine Learning Research, 527–536. International Convention Centre, Sydney, Australia: PMLR.

Chandrashekar, G., and Sahin, F. 2014. A survey on feature selection methods. Computers & Electrical Engineering 40(1):16–28.

Chang, K.-W.; Krishnamurthy, A.; Agarwal, A.; III, H. D.; and Langford, J. 2015. Learning to search better than your teacher. In ICML.

Choi, E.; Hewlett, D.; Uszkoreit, J.; Polosukhin, I.; Lacoste, A.; and Berant, J. 2017. Coarse-to-fine question answering for long documents. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers, 209–220.

Conneau, A.; Schwenk, H.; Barrault, L.; and Lecun, Y. 2016. Very deep convolutional networks for natural language processing. arXiv preprint arXiv:1606.01781.

Faruqui, M.; Tsvetkov, Y.; Yogatama, D.; Dyer, C.; and Smith, N. 2015. Sparse overcomplete word vector representations. arXiv preprint arXiv:1506.02004.

He, H.; Daumé III, H.; and Eisner, J. 2013. Dynamic Feature Selection for Dependency Parsing. In EMNLP, 1455–1464.

Karayev, S.; Fritz, M.; and Darrell, T. 2013. Dynamic feature selection for classification on a budget. In International Conference on Machine Learning (ICML): Workshop on Prediction with Sequential Models.

Kusner, M. J.; Chen, W.; Zhou, Q.; Xu, Z. E.; Weinberger, K. Q.; and Chen, Y. 2014. Feature-Cost Sensitive Learning with Submodular Trees of Classifiers. In AAAI, 1939–1945.

Lei, T.; Barzilay, R.; and Jaakkola, T. S. 2016. Rationalizing neural predictions. CoRR abs/1606.04155.

Leroux, S.; Bohez, S.; De Coninck, E.; Verbelen, T.; Vankeirsbilck, B.; Simoens, P.; and Dhoedt, B. 2017. The cascading neural network: building the internet of smart things. Knowledge and Information Systems 1–24.
