# BIBI System Description: Building with CNNs and Breaking with Deep Reinforcement Learning

**Yitong Li** and **Trevor Cohn** and **Timothy Baldwin**
School of Computing and Information Systems
The University of Melbourne, Australia
yitongl4@student.unimelb.edu.au, {tcohn,tbaldwin}@unimelb.edu.au

## Abstract

This paper describes our submission to the sentiment analysis sub-task of "Build It, Break It: The Language Edition (BIBI)", on both the builder and breaker sides. As a builder, we use convolutional neural nets, trained on both phrase and sentence data. As a breaker, we use Q-learning to learn minimal change pairs, and apply a token substitution method automatically. We analyse the results to gauge the robustness of NLP systems.

## 1 Introduction

Recently, deep learning models have made impressive gains over a range of NLP tasks (Bahdanau et al., 2015; Bitvai and Cohn, 2015). However, recent studies have exposed brittleness in the models, e.g. through adversarial examples (Szegedy et al., 2014; Goodfellow et al., 2015). In these papers, researchers construct cognitively implausible perturbations of raw image inputs to fool state-of-the-art deep learning models. These perturbations are cheap and easy to generate using a "fast-gradient" method, based on analysis of the derivative of the loss with respect to the input.

One issue with the generation of adversarial examples for NLP has been the fact that language data is discrete, and hence difficult to map the continuous outputs of "gradient" methods onto. Furthermore, the perturbations or mutations generated through adversarial methods may be nonsensical to humans. Given this background, the BIBI shared task was devised to study the reliability of NLP systems by generating adversarial test instances, and explicitly training systems to be robust against adversarial test instances. Specifically, the task is based on opposing sets of participants: *builders* aim to build systems robust to different inputs, and *breakers* try to construct instances which will cause the builders' systems to make incorrect predictions.

In this paper, we describe our builder and breaker submissions to the sentiment analysis subtask, which is a sentence-level binary classification task, to predict whether a given review sentence is positive or negative with respect to a given movie. The data set is derived from movie review data (Pang and Lee, 2005) and the Stanford Sentiment Treebank (Socher et al., 2013).

We participated both as a builder and breaker because we are interested in testing the robustness of state-of-the-art neural models, such as convolutional neural networks ("CNNs": Kim (2014)). Also, we were interested in the breaker task as an avenue for exploring how well we can automatically construct adversarial test instances. In the sentiment sub-task, the main job of breakers is to construct minimally-changed pairs that are able to fool the builders' sentiment analysers. For example, the following sentences can be considered to be a minimal pair, with positive (+1) sentiment:

(+1) I love this movie!
(+1) I'm mad for this movie!

## 2 Approach

Here, we describe the methods we used for both the builder and the breaker. Considering the expense of human judgements, especially for breakers, and the strong desire for the approaches to generalize, we decide to use automatic methods for both tasks.

### 2.1 Builder System Description

As a builder, we chose to use convolutional neural nets ("CNNs"), based on their strong performance over text classification tasks (Kim, 2014; Zhang et al., 2015). Specifically, we were interested in

testing the robustness of CNNs in NLP applications. We apply Kim (2014)'s model to this task, which is easy and fast to train. We train our models on both phrase-level labelled data (with neutral phrases removed), and sentence-level labelled data; we will refer to these as "phrased-based" and "sentence-based" CNNs, respectively. Below, we present a short outline of the CNN model.

#### 2.1.1 Convolutional Neural Network

The CNN model first operates by embedding each word using a look-up table which is stacked into the sentence matrix $\mathbf{E}_{\mathrm{S}}$. Then, a 1d convolutional layer is applied to $\mathbf{E}_{\mathrm{S}}$, which applies a series of filters over each window of $t$ words, with each filter employing a rectifier transform function. In practice, we use window widths of size $t \in \{3, 4, 5\}$, and 128 filters for each size. $\mathrm{MaxPooling}$ is applied to each of the three sizes separately, and the resulting vectors are concatenated to form a fixed-size representation of the given sentence or phrase. Finally, the representation vector is fed into a final $\mathrm{Softmax}$ layer to generate a probability distribution over classification labels.

The model is trained to minimize the loss — defined as the cross-entropy between the ground-truth and the prediction — using the Adam Optimizer (Kingma and Ba, 2015) with a learning rate of $10^{-4}$ and batch size of 64.

### 2.2 Breaker System Description

As our breaker, we borrow ideas from generating adversarial examples in computer vision (Szegedy et al., 2014).

Assuming the loss of the system $s$ is accessible and the true label $l$ of the sentence is known, then the given task can be seen as an optimization problem where we simultaneously minimize the loss between the perturbed sentence $h(x)$ and flipped label, and also the distance between them :

$$\min_{\theta_h} \mathcal{L}_s(h(x), \mathbf{1} - l) + \alpha \cdot \mathrm{distance}(h(x), x) \quad (1)$$

Here, system $s$ maps the input sentence $x$ into the label space, and $h$ is the perturbing function.

In text applications, this can be seen as an integer programming problem. Generally, integer optimization is NP-hard (Cunningham et al., 1996), although estimations can be found using heuristic methods, such as simulated annealing. However, considering the complexity of language, solving the given optimization function in only a discrete text space could lead to nonsensical outputs to a human, according to the results of our preliminary experiments [1]. Empirically, this can be attributed to the difficulty of defining an order over a natural language token set, as well as the non-convex nature of the semantic space in natural language generation.

Therefore, instead of optimizing Equation (1) directly, we split the problem into two subtasks: first, we use a reinforcement learning method to learn which tokens or phrases should be changed; and second, we apply a substitution method to those selected tokens, ensuring the quality of the new sentence.

#### 2.2.1 Reinforcement Learning Method

In order to learn the sentiment of a given text, most NLP systems use $n$-gram feature-based learning methods, including traditional bag-of-words methods (Pang and Lee, 2005) as well as deep learning models (Socher et al., 2013; Kim, 2014). Based on this observation, one intuitive method of fooling the system is to find the "important" tokens within a given sentence, and then modify these to trick a given system into making a wrong prediction.

In our method, we need a baseline system for our breaker method to attack. Here, we choose the sentence-based CNN model, as described above, as an imaginary enemy. For most black-box systems, it is impossible to access the internals of the model and parameters. Therefore, given an input instance, we only use the output of the system, such as the prediction and loss in our method.

To solve this discrete problem, we apply a Reinforcement Learning ("RL") method (Sutton and Barto, 1998), specifically Q-learning (Watkins and Dayan, 1992; Mnih et al., 2013), to model the probability of removing tokens or phrases from a given sentence. Given the token $x$ and an instance of context sentence $c$, the RL system learns a policy function $\pi(x|c) \rightarrow \{\mathrm{remove}, \mathrm{keep}\}$. We consider each instance as one game, consisting of several rounds. In the first round, $c$ is a randomly-selected sentence, $x$ is the first token in $c$, and $\pi$ is the decision process of removing $x$ or not. In each round, $\pi$ will be learned at the token level, and the resulting sentence will be taken as the new context. The game will be repeated iteratively un-

[1] We used simulated annealing method to solve the given constrain problem directly. The details of results are not presented in this paper.

| Builder system | $\mathcal{F}_1$ | % of broken examples | | | | | |
|---|---|---|---|---|---|---|---|
| | | Total | Average | Breaker 1 | Breaker 2 | Breaker 3 | Breaker 4 |
| Builder 1 | **0.528** | 25.43 | 26.76 | 35.35 | 22.62 | 39.79 | 9.28 |
| RNN (Socher et al., 2013) | 0.457 | 25.96 | 27.45 | 34.34 | 27.38 | 36.73 | 11.34 |
| DCNN (Kalchbrenner et al., 2014) | 0.483 | 25.09 | 25.95 | 34.84 | 21.42 | 36.73 | 10.82 |
| Bag-of-$n$-grams | 0.510 | 24.74 | 25.51 | 38.38 | 20.23 | 36.73 | 6.70 |
| Phrase-based CNN | 0.518 | **24.39** | **25.23** | 35.35 | 22.62 | 33.67 | 9.28 |
| Sentence-based CNN | 0.490 | 28.57 | 31.42 | 39.39 | 39.28 | 39.79 | 7.21 |

Table 1: The results of builder systems for BIBI blind test set based on average $\mathcal{F}_1$ (higher is better) and percentage of broken cases (lower is better). Details of each builder's system against the breaker's examples are also listed. The best results are indicated in **bold**.

til a max-round limitation is hit. When the game is terminated, the reward of the game will be the loss difference between the original sentence and the residual sentence, where the loss is calculated relative to the baseline system $s$. Additionally, we use the number of removed tokens as a penalty item in the final reward. The procedure will then randomly select a new instance and start a new game.

For training, we use the standard Deep Q-Learning algorithm, as described in Mnih et al. (2013). For hyper-parameters, max-round is set to 100 and $\gamma$ to 0.01. The feature extractor $\phi$ is a multi-layer perceptron over token embeddings, initialized by pre-trained word2vec vectors (Mikolov et al., 2013). The batch size is 128, the initial $\epsilon$ is set to 0.3, and the memory size is up to $10,000$. In order to change as few tokens as possible, we empirically set the distance penalty $\alpha$ to 2. The reward is calculated using pre-trained sentence-based CNN, as described above.

#### 2.2.2 Token Substitution Method

Once the algorithm has decided which tokens should be changed, the next move is to find appropriate substitutions. As described above, most systems are based on $n$-grams, making them very sensitive to unknown tokens. Therefore, we came up with some heuristics.

The first approach draws on our earlier work on learning robust text representations (Li et al., 2017), and is based on synonyms of the given token, based on Princeton WordNet (Miller et al., 1990) using the NLTK API (Bird, 2006). Here, we test possible synonyms, considering their part-of-speech tag, asking the system $s$ whether the loss is reduced after substitution. We also tried to find antonyms that cannot be recognized by the system, causing the predicted sentiment label to not flip. Finally, we add a small amount of human supervision to ensure the fluency of the output sentences, including removal of garbled examples and minor grammar corrections, and to ensure they have the correct sentiment label. To be specific, we discard the "bad-attacked" pairs with loss difference less than 1 empirically. These "bad-attacked" pairs might be able to fool the sentence-based CNN but with low confidence, such that we did not expect them to be good enough to fool other builders' systems. We also filter out sentences with wrong or ambiguous sentiment labels manually. For example, the system sometimes generates expressions with correct grammar but strange sentiment — e.g., *I don't like this lovely movie* which is contradictory and possibly interpretable as ironic — which remains a challenging problem for us to totally eliminate during generation. Last, we slightly modify the outputs to fix minor grammar errors, such as adding or removing the determiner *a* or *the*.

## 3 Results and Analysis

In this section, we detail the results of our methods, and perform error analysis.

### 3.1 Builder

The results for the builder systems over the test set are shown in Table 1. To evaluate the robustness of the builder systems, there are two evaluation criteria: average F-score ("$\mathcal{F}_1$") across all breaker test cases (higher is better), and the percentage of breaker test cases that break the system (lower is better). Having a builder fail over only one example in a given minimal pair is considered to have broken that system.

We observed that all the systems are very close over these two criteria. We also see that the phrase-based CNN achieved competitive performance, while the sentence-based CNN is not as

robust. This aligns with our intuition, as feeding phrase-labeled data is more precise for model training, and it is much easier for the sentence-based model to overfit the data, according to our analysis. For example, it might consider *the performance is* as a strong positive trigram feature instead of a neutral one, because the expression has higher frequency in the positive training set than that in the negative set. This also occurs for certain entity tokens, such as people's names and places.

To better understand the advantages and disadvantages of CNNs, we perform some error analysis.

One major class of breaker attack is modifying the polarity of a sentence, either syntactically (e.g. by adding/removing *not*) or morphologically (e.g. by adding the prefix *un-*), but actually the CNN is relatively robust to this. We believe the reason is that the $n$-gram features our CNN learns are more robust representations of words and short phrases. This also explains the performance of the bag-of-$n$-gram (BoN) system. However, CNN is still slightly better than BoN because CNN only learns the most important features through the $\mathrm{MaxPooling}$ operation, and using word embeddings appears to help the model deal with synonyms and antonyms at the word level.

It is almost the same situation when the systems encounter out-of-vocabulary words (OOV). Although OOVs are a significant challenge, we believe they can be overcome by training better sentiment-sensitized word embeddings (Mrkšić et al., 2016), or combining the system with character-level normalization methods (Han and Baldwin, 2011).

However, CNNs are not good at dealing with complex grammatical structures or long-distance dependencies. For instance, changing a comparative from *more than* to *less than* flips the sentiment and is something that humans are sensitized to, but CNNs tend not to capture this difference. Also, CNNs are not sensitive to tense, such as changing the present tense *is* to the past tense *was* to capture pragmatic/connotative effects. For these kinds of examples, we expected to see higher performance among models which better capture syntactic structure, such as recursive neural nets ("RNNs": Socher et al. (2013)) and dynamic convolutional neural networks ("DCNNs": Kalchbrenner et al. (2014)). In practice, however, this was not the case. We cannot conclude the exact reasons without further analysis of these models, however this might indicate that these perceptron-based deep models struggle to capture the logic in human langauge.

To conclude, among traditional models and state-of-the-art deep learning models for sentiment analysis, CNNs are relatively robust.

| Test set | Average $\mathcal{F}_1$ | Score |
|---|---|---|
| Breaker 1 | 0.79 | 28.64 |
| Breaker 3 | 0.84 | **31.17** |
| Breaker 4 | 0.83 | 7.48 |
| Breaker 2 (our method) | 0.75 | 19.28 |

Table 2: The final score of the breakers. The average $\mathcal{F}_1$ over the original sentences of all builders' systems is also listed for each break test set.

### 3.2 Breaker

Table 2 gives the final scores of the breaker teams. The final score is calculated by averaging the $\mathcal{F}_1$ of each builder's system on the original sentences, multiplied by the percentage of examples that break that system (shown in Table 1).

Overall, about one third of the breakers' examples were able to fool the builders' systems, which is not surprising. On the one hand this is encouraging, in that, without taking the untapped test cases into consideration, each builder can handle more than half of the break examples. On the other hand, still nearly one third of the break sentences cannot be handled by state-of-the-art statistical models.

For our breaking approach, we observe that all the builders' systems have lower $\mathcal{F}_1$ on our test set, indicating that our method tends to generate difficult sentences, where systems might have lower confidence. Actually, in our final submission, we only chose $42$ pairs as the final break data from among the $521$ test data instances provided by the organisers, as the rest of the generated pairs were removed due to low confidence or bad quality sentences. This unfortunately indicates that our automatic method cannot be applied to all examples, making our approach limited in application. Therefore, we can't really conclude that our automatic approach is a success, and we should explore more flexible approaches in the future. However, the approach itself still achieves a break rate higher than the error rate on the origi-

nal test set. Additionally, our method breaks the sentence-based CNN — which our RL model is built on — with $39.28\%$ break rate.

Based on error analysis over the broken examples, we found that using tokens with opposite sentiment in an example worked across builder systems in most cases. For instance, *if you love to waste your time* could confuse most systems because of the contrast between *love* and *waste time*, because they indicate opposing sentiment in isolation, while the phrase itself is focused on *waste time*, which is very difficult for most NLP systems to understand. This indicates that representations of natural language may be doing more than simply adding or transforming the word embeddings, and instead non-compositionally transforming the logic structure of the sentence.

Also, attacking words or phrases which are ambiguous between positive and negative sentiment is also a potentially effective approach. For example, *rock* is used predominantly in positive-sentiment contexts, in reference to jewels or strong/reliable people, meaning that systems are likely to learn that it has exclusively positive sentiment due to bias in the training set. However, when the negative substitution phrase *on the rocks* (meaning "in trouble") is used, the builders' systems might still predict the idiom as having positive sentiment.

Based on these observations, we can conclude that state-of-the-art statistical models have only minimal "understanding" of natural language. The examples we showed above are relatively simple, but in real cases, they can be more complex. And we are not even considering the ambiguity of language or tone of the language in different contexts. To summarize, our approach provides a method to study the robustness of modern NLP systems over a sentiment analysis task. Our results demonstrate that NLP systems are still far from turning the corner to real language understanding.

## 4 Conclusions

In this paper, we have described our builder and breaker systems, in the context of the BIBI the Language Edition shared task. We built sentiment analysis systems using text-based convolutional neural networks, trained on either phrase- and sentence-level data. Also, we used reinforcement learning and substitution methods to generate adversarial test examples automatically. We performed error analysis to better understand the robustness of statistical NLP models.

## Acknowledgements

We would like to thank the three anonymous reviewers for their helpful feedback and suggestions, and to Meng Fang for assisting with the implementation of the RL system.

## References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations*.

Steven Bird. 2006. NLTK: The natural language toolkit. In *Proceedings of the COLING/ACL 2006 Interactive Presentation Sessions*. pages 69–72.

Zsolt Bitvai and Trevor Cohn. 2015. Non-linear text regression with a deep convolutional neural network. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. pages 180–185.

William H Cunningham, S Thomas McCormick, and Maurice Queyranne. 1996. *Integer Programming and Combinatorial Optimization*. Springer.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. In *Proceedings of the International Conference on Learning Representations*.

Bo Han and Timothy Baldwin. 2011. Lexical normalisation of short text messages: Makn sens a #twitter. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. pages 368–378.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. pages 655–665.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. pages 1746–1751.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations*.

Yitong Li, Trevor Cohn, and Timothy Baldwin. 2017. Robust training under linguistic adversity. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. pages 21–27.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26*. pages 3111–3119.

George A. Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine J. Miller. 1990. Introduction to WordNet: an on-line lexical database. *International Journal of Lexicography* 3(4):235–244.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. 2013. Playing atari with deep reinforcement learning. *CoRR* abs/1312.5602.

Nikola Mrkšić, Diarmuid Ó Séaghdha, Blaise Thomson, Milica Gašić, Lina M. Rojas-Barahona, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. Counter-fitting word vectors to linguistic constraints. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. pages 142–148.

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

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. pages 1631–1642.

Richard S Sutton and Andrew G Barto. 1998. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, USA.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2014. Intriguing properties of neural networks. In *Proceedings of the International Conference on Learning Representations*.

Christopher JCH Watkins and Peter Dayan. 1992. Q-learning. *Machine learning* 8(3-4):279–292.

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*. pages 649–657.