# Byte-Pair and N-Gram Convolutional Methods of Analysing Automatically Disseminated Content on Social Platforms

Houjun Liu

*houliu@nuevaschool.org*

**The Nueva School**. 131 E. 28th Ave., San Mateo, CA 94025, United States

**Abstract**

In this experiment, an efficient and accurate network of detecting automatically disseminated (bot) content on social platforms is devised. Through the utilisation of parallel convolutional neural network (CNN) which processes variable n-grams of text 15, 20, and 25 tokens in length encoded by Byte Pair Encoding (BPE) (Gage, 1994), the complexities of linguistic content on social platforms are effectively captured and analysed. With validation on two sets of previously unexposed data, the model was able to achieve an accuracy of around 96.6% and 97.4% respectively — meeting or exceeding the performance of other comparable supervised ML solutions to this problem. Through testing, it is concluded that this method of text processing and analysis proves to be an effective way of classifying potentially artificially synthesized user data — aiding the security and integrity of social platforms.

**Keywords**: bot detection; machine learning; natural language processing; computational linguistics

## 1 Introduction

Because of the increasing demand for the acquisition of an audience in the promotion of myriad (and sometimes devious) causes, various social media platforms have seen an insurgence of automatically generated or disseminated content as a method for some to grow an "inorganic audience" to their content in hopes of the eventual acquisition of an "organic audience". (Ferrara, Varol, Davis, Menczer, & Flammini, 2014)

The inherent goal of these systems is to pass as a human (or at least a legitimate organization), and due to the sheer quantity of the generated content, it is often difficult and impractical to have human evaluators tag each of such accounts accurately and efficiently. Hence, such a task of classification must be handled with an artificial and automated approach.

A traditional way of approaching this problem, and indeed the method employed by many social platforms, is to use some implementation of a behavior analysis algorithm: that is, a model that focusses on analysis of the *activity* and not the *content* of the offending users; however, the spontaneity and diversity of the bot systems makes this method increasingly ineffective.

The solution, naturally, would be to create a model that is able to analyse features that exist within the content being disseminated itself and that is able to make classification decisions based on this analysis. Although there exist many implementations of such systems that approach this problem in more traditional formats such as e-mails or message boards, many suffer when trained against social media data because of the generative vocabulary base and complex (often rapidly changing) methods of expression.

In this experiment, a method is devised to address these difficulties on the social platform Twitter. By combining the advantages of text processing using convolutional neural networks, split n-gram processing, and character-level compression encoding (Byte-Pair Encoding) a reasonably efficient and accurate model is trained to classify automatically-generated and disseminated data.

## 2 Methodology

In this section, the tools used in the solution of this problem is illustrated, along with reasoning behind how these tools are advantageous in the solution of such problems.

### 2.1 Background

The implementation of this problem revolves around the use of convolutional neural networks and byte-pair encoded data to solve a linguistic analysis problem. This sub-section attempts to provide some background upon the methods aforementioned in a perspective that pertains to this problem.

**Byte-Pair Encoding** Because of the constantly evolving nature of social networks, it is difficult to capture the syntactic complexity of the raw text data using traditional text-embeddings such as Continuous Bag of Words (CBoW) or Skip-Gram; both the brevity and high rate of synthesis among the social data makes training such models difficult.

In this experiment, it is reasoned that the text could be represented most accurately and efficiently by a method that focuses on encoding features below the word level due to the neologistic nature of social networks. However, traditional character-embedding methods are not only inefficient, but often cannot extract useful features from linguistic data.

Hence, it is reasoned that the best method for representing such data within this experiment is to use an approach of subword data extraction — where character-based textual data are "summarized" with a certain deterministic algorithm to a certain fixed length — that provides a method of feature extraction that also handles unknown and synthesized vocabulary with reasonable representation, without relying on previously trained embeddings.

The text compression method called byte-pair encoding (Gage, 1994) is used in this experiment. This compression/encoding method has been shown to be effective in processing social media data (Bodapati, Gella, Bhattacharjee, & Al-Onaizan, 2019), as it revolves around the analysis of the frequencies of "byte-pair"s within an initial corpus, combining the most frequently occurring pairs into one token, and iterating over the compressed corpus until the desired size is reached. The algorithm is implemented as follows in this experiment:

**Algorithm 1:** Byte-Pair Encoding

```
input : An array of characters $C$ in a string
        Desired max length $M$
output: A compressed character string $O$
1  while $dim(C) \neq M$ do
2      $B \longleftarrow \emptyset$
3      for $C_i \in C$ do
4          $B \longleftarrow B + \{(C_i, C_{i+1})\}$
5      end
6      $P \longleftarrow mode(B)$
7      $O \longleftarrow \emptyset$
8      for $C_i \in C$ do
9          if $(C_i, C_{i+1}) \in B$ then
10             $O = O + \{(C_i) + (C_{i+1})\}$
11         else
12             $O = O + \{(C_i), (C_{i+1})\}$
13         end
14     end
15     $C \longleftarrow O$
16 end
17 return $O$
```

Used as a means of feature extraction for NLP applications — especially machine translation (Kunchukuttan & Bhattacharyya, 2016), (Sennrich, Haddow, & Birch, 2015) — the process of BPE allows for reasonably efficient data extraction and the ability to handle unknown vocabulary. As BPE preferentially combines the most common tokens, the compressed content will often feature tokens that include common word conjugations (e.g., $-ing$, $-ed$, etc.) and subwords (e.g., $un-$, $-ology$, etc.), creating an emulation of the common processes of language synthesis on social platforms and capturing potential misspellings of words. Furthermore, due to the fact that the "worst case" scenario for BPE is simply an uncompressed character embedding, completely unknown strings are also reasonably tokenized and captured.

**N-Gram CNN** After the selection of an appropriate algorithm of text preparation, it is also important to select an appropriate network topology for the classification model. Convolutional neural networks has shown state-of-the-art results (Zhang, Zhao, & LeCun, 2015), (Collobert et al., 2011) in linguistic processing because of their relatively shallow architecture (as compared to other linguistic processing methods such as RNNs), preventing dramatic overfitting or severe vanishing gradients.

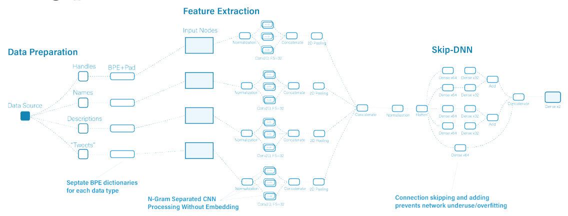
In earlier implementations, 1D convolutional filters are convolved against tokenized or embedded data, with many fixed-width one-dimensional filter sliding through the embedded text capturing a specific feature that is processed later in the network.

However, it is difficult to achieve accurate training results using this traditional method as the training data is sourced from the social network Twitter, and the syntactic integrity of each given Tweet is highly variable. Therefore, it is hypothesized that traditional 1D convolutions are not able to capture the syntactic complexities of even byte-pair encoded data: that it is difficult to predict the information contained in any given fixed interval, making it difficult to extract usable features.

As such, this experiment endeavours to create a network with a mix of parallel CNN layers, with each capturing a differently-length kernel of encoded tokens' information; the classification network then uses the concatenated information of each of such variable-length kernel to generate decisions. Through this method, it is hypothesized that the information extracted in any given convolutional layer would have an effect that compliments the others: that instead of information capture at one fixed length of tokens, one (or more) of the convolutional layers capturing at a fitting length for a given sequence would return usable features and activate the remaining network for prediction.

## 2.2 Network Topology

The design of the network assumes that the input sequences have been appropriately encoded and padded, and begins by four input tensors processing each type of signal (usernames, handles, user descriptions, and their "status line" respectively, which is the content being evaluated.) Each input then is normalized to improve training performance.



**Feature Extraction** The normalized input sources are fed each in a separate information extraction network following the 1D-ngram design aforementioned. The expanded features are processed simultaneously by three convolutional layers, each having a different kernel size with the same number of filters, aiming to extract features from information of different lengths.

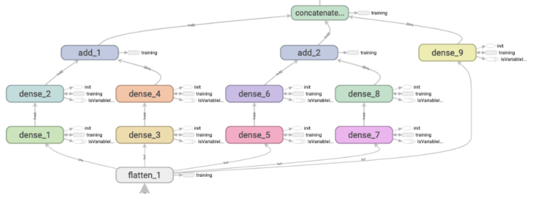
The parallel convolutional layers used in the data extraction network has all 36 filters, each having kernel sizes of 10, 15, and 20 respectively: aiming to extract information at varying syntactical lengths. The output from each of such convolutional layers are then concatenated and pooled — becoming the feature map of each input.

Two of the four features' final weights representing user names and handles are at this point L2 regularized, a measure taken to prevent the model's sheer memorisation (then eventual overfitting) of known bot user handles.

Non-linearities in this portion of the network is introduced using the ReLU activation function.

**Classification** The classification network is an adaptation of a skipped-connection dense neural network. The method of connection skipping is both a method of forcing the network to generalise, but also force the network to capture information from minimal processing. Various dense layers are connected in groups of two, then the groups are added and concatenated. This method allows a network with a large amount of parameters that is not extremely prone to overfitting.

The procedure of the connection skipping is as follows:



All hidden layers within this network is activated using the ReLU function, whereas the last layer is activated using the Softmax function for the task of classification.

## 3 Implementation

### 3.1 Data Selection and Preparation

There are two sets of data collected to be used in this experiment for training. Both of the datasets used are sourced from the curation of the Indiana State University bot detection dataset repository (of Indiana University, 2015); specifically, the manually-annotated human/bot account dataset created by Michele Mazza, et al., (Mazza, Cresci, Avvenuti, Quattrociocchi, & Tesconi, 2019) and a combination of known verified Twitter accounts & registered bots in "botwiki" curated by Yang, Kai-Cheng Yang, et al. (Yang, Varol, Hui, & Menczer, 2019)

The user objects sourced then are scraped for their historical Tweets, each containing the content (stylised "status") of the Tweet, the timestamps at which the action occurred, and Twitter's hypothesized language. Each user's name, handle, and description are also downloaded for use by the model. A maximum of 1,000 items is downloaded for each user object.

As the design of this study is limited to Roman-derived languages, the downloaded data are filtered to remove tweets containing pictorial language scripts (i.e. Chinese characters, Hangul, Hiragana and Katakana, Arabic abjad, etc.) as it is more difficult to capture its syntax using the encoding methods devised. Unicode Emojis within Roman-conforming content are, however, allowed, and are treated as its own byte during Byte-Pair encoding (BPE).

All data is then compressed to a factor of 0.2 of original size using BPE, with each type of content encoded separately.

### 3.2 Model Performance

During this experiment, the model is initialized using Xavier initialization on all weights, and is trained using the Adam optimizer optimizing for Binary Cross-Entropy Loss.

The performance of the proposed technique below are tested using 2,000 samples of previously unexposed Tweet data curated by Cresci, et al. (Cresci, Di Pietro, Petrocchi, Spognardi, & Tesconi, 2017) and training data withheld from the model curated by Yang, et al. (Yang et al., 2019), with more than 50% of each group of testing data being entirely new user accounts never exposed to the model.

Table 1: Supervised Model Performance During Testing

| Model | Data | Acc. | Prec. | Rec. |
|---|---|---|---|---|
| **Proposed** | Cresci, et al. | 96.1% | 89.76% | 99.6% |
| Cresci, et al. | Cresci, et al. | 99.7% | 97.7% | 98.2% |
| Yang, et al. | Cresci, et al. | 50.6% | 56.3% | 17.0% |
| Wei, et al. | Cresci, et al. | 96.1% | 94.0% | 97.6% |
| **Proposed** | Yang, et al. | 97.4% | 93.22% | 99.9% |

Acc. = Prediction Accuracy, Prec. = Precision, Rec. = Recall

## 4 Conclusion/Discussion

In this experiment, a novel method of automatic detection of inorganic data on social platforms is proposed. The utilisation of Byte-Pair Encoding as an encoding method offers an efficient and effective way of processing raw linguistic data; the direct processing of n-grams of encoded data using a convolutional neural network serves as a deeper and trainable feature extraction method that enhances training.

When tested on two different sets of previously unexposed data, the model's performance both in terms of accuracy and confusion matrix values meets, and sometimes exceeds, other comparable solutions of the same classification problem. The purely linguistic approach of this solution offers a high level of versatility to this method as it requires only the content being disseminated to make classification decisions, and not any previous knowledge of the behavior of the given account.

In the future, this experiment could be adapted to analyse text not simply in the Roman-derived alphabets (as is necessitated here by Byte-Pair encoding), as a large amount of content on the social platforms is of non-Roman origin. Furthermore, as the proposed network is a fairly large model, future work could be done to increase the performance of the network. Nevertheless, this experiment offers a promising step to solving the influx of unwanted inorganic content on social platforms, creating higher standards of safety and integrity for such platforms.

## 5 Acknowledgements

This experiment would not have been possible without the generous support of Mr. Klint Kanopka, Mr. Wes Chao, and Mrs. Jennifer Selby for lending their patience and expertise in mentorship. Gratitude is also due to Dr. John Feland, Mrs. Monika Almudafar-Depeyrot, and Ms. Susan Cole for their help with this project.

## References

Axelbrooke, S., & Gosukonda, S. (2017). Bpe. Retrieved February 19, 2020, from https://github.com/soaxelbrooke/python-bpe/

Bodapati, S. B., Gella, S., Bhattacharjee, K., & Al-Onaizan, Y. (2019). Neural word decomposition models for abusive language detection. eprint: arXiv:1910.01043

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., & Kuksa, P. (2011). Natural language processing (almost) from scratch. *Journal of machine learning research*, *12*(Aug), 2493–2537.

Cresci, S., Di Pietro, R., Petrocchi, M., Spognardi, A., & Tesconi, M. (2017). Social fingerprinting: Detection of spambot groups through dna-inspired behavioral modeling. *IEEE Transactions on Dependable and Secure Computing*, *15*(4), 561–576.

Ferrara, E., Varol, O., Davis, C., Menczer, F., & Flammini, A. (2014). The rise of social bots. doi:10.1145/2818717. eprint: arXiv:1407.5225

Gage, P. (1994). A new algorithm for data compression. *C Users J. 12*(2), 23–38.

Jacovi, A., Shalom, O. S., & Goldberg, Y. (2018). Understanding convolutional neural networks for text classification. arXiv: 1809.08037

Kunchukuttan, A., & Bhattacharyya, P. (2016). Learning variable length units for smt between related languages via byte pair encoding. eprint: arXiv:1610.06510

Mazza, M., Cresci, S., Avvenuti, M., Quattrociocchi, W., & Tesconi, M. (2019). Rtbust: Exploiting temporal patterns for botnet detection on twitter. eprint: arXiv:1902.04506

of Indiana University, T. T. (2015). Bot repository. Retrieved February 19, 2020, from https://botometer.iuni.iu.edu/bot-repository/datasets.html

Sennrich, R., Haddow, B., & Birch, A. (2015). Neural machine translation of rare words with subword units. arXiv: 1508.07909

Vo, D. T., & Khoury, R. (2019). Language identification on massive datasets of short message using an attention mechanism cnn. arXiv: 1910.06748

Yang, K.-C., Varol, O., Hui, P.-M., & Menczer, F. (2019). Scalable and generalizable social bot detection through data selection. eprint: arXiv:1911.09179

Zhang, X., Zhao, J., & LeCun, Y. (2015). Character-level convolutional networks for text classification. arXiv: 1509.01626