OPTION #1: Essay - Feature Engineering and Hyperparameter Tuning

Scott Miner

Colorado State University – Global Campus

OPTION #1: Essay - Feature Engineering and Hyperparameter Tuning

This paper addresses how feature engineering might enhance a hypothetical text classification model to identify abusive content online, also known as cyberbullying (CB) content or hate speech. The paper discusses features that might be extracted from a dataset to boost a classifier's predictions and the types of classifiers that one might use in such scenarios, including Support Vector Machines (SVMs), one of the most popular classifiers in hate speech detection (Schmidt & Wiegand, 2017). The paper also discusses the methods SVMs use for predicting, their hyperparameters, and the tuning techniques from which they might benefit.

Text Classification

Biniz *et al.* (2018) define text classification as separating texts into content-based classes, a process gaining popularity due to the proliferation of textual data online. Fenner (2019) writes that compared to data grouped in tables, text documents present some additional challenges to machine learning (ML) algorithms, including that they vary in length, are order-dependent, and are unaligned. Therefore, text classification problems involve two steps: (a) selecting specific features from all those available using feature engineering and (b) applying classification algorithms to those chosen features (Biniz *et al.*, 2018).

Feature Engineering

Schmidt and Wiegand (2017) explain *feature engineering* as a crucial step in the text classification pipeline since the features used by algorithms are one of their distinguishing factors. Biniz *et al.* (2018) describe feature engineering in text classification as comprising three main steps: (a) extracting all words from corpora, (b) selecting only significant words, and (c) attributing a weight to each word denoting its significance. Fenner (2019) decomposes feature engineering into several methodologies: (a) *feature extraction*, which converts low-level data like text data unsuitable for ML algorithms into higher-level tabular formats; (b) *feature*

selection, which removes unimportant or redundant features; (c) *scaling* and *normalization*, which adjusts the data's range and center to ease learning; and (d) *feature construction*, which creates new features from existing ones.

The Bag-of-Words (BOW) Approach

One common method to extract features from text is the bag-of-words (BOW) approach, which Scott and Matwin (1999) describe as dominating the text classification literature. Martins and Matsubara (2003) define the BOW approach as representing text documents and their terms in tabular forms, where each row represents a document and each column a word. The BOW approach uses Boolean indicators to indicate the presence or absence of simple (i.e., 1-gram) or composed (i.e., 2, 3, …, *n*-gram) words.

Character- and Word-Level Representations

Schmidt and Wiegand (2017) write that ML algorithms utilizing the simple surface-level features described in the BOW approach yield good classification performance on hate speech detection. Further, Mehdad and Tetreault (2016) found character-level *n*-grams more predictive than word-based features for hate speech detection since character-level attributes attenuate the spelling variations encountered by ML algorithms when dealing with user-generated text. Other surface features from which hate speech detection can benefit include capitalization, punctuation, URL mentions, and comment and token lengths (Schmidt & Wiegand, 2017).

Term Frequency-Inverse Document Frequency (TF-IDF)

Fenner (2019) expands upon the BOW approach described above, producing counts of all document words. Moreover, by normalizing these counts and producing a term frequency-inverse document frequency (TF-IDF) for each word, ML algorithms can inhibit longer documents from having undue influence on target variables and inhibit the contributions of

frequently occurring words because, as a word's frequency increases, its ability to distinguish between documents lessens.

Tokenization, Stopwords, and Stemming

Taking a step back, before ML algorithms can create BOW representations of documents, they must tokenize them. Scott and Matwin (1999) write that text classifiers often strip documents of their case information and punctuation during tokenization. Further, classifiers may remove *infrequent* words occurring below a certain threshold and *frequent* words, known as *stopwords*, like "the," "and," and "of." Finally, text classifiers commonly implement a feature engineering technique known as *stemming*, which aims to make features more statistically independent by mapping their morphological variants, such as in the cases of "learned" and "learning," to a common root form: "learn."

Feature Engineering for Detecting Hate Speech

Cyberbullying (CB), a conscious and persistent act of violence intended to harm individuals using online forms of communication repeatedly, is one of the most common online risks for adolescents (Cheng *et al.*, 2019; Talpur & O'Sullivan, 2020). Further, Talpur and O'Sullivan (2020) describe feature engineering as one of the most common approaches to improving CB detection classifiers' performance. Levy and Goldberg (2014) write that one of the most common features of hate speech detection classifiers is derived from neural network-inspired training methods. *Word embeddings* are dense vector word representations that aim to capture the semantic and syntactic relationships between words. Other useful features for improving hate speech detection algorithms include (a) *network-based features*, such as users' number of followers, (b) *activity features*, such as the times that users wrote messages, (c) *user features*, including users' age and gender, (d) *content-based features*, such as profane words and

part-of-speech (POS) tagging, and (e) *personality features*, like users' extraversion and neuroticism (Talpur & O'Sullivan, 2020).

Machine Learning Classifiers

Talpur and O'Sullivan (2020) describe ML approaches that detect hate speech as implementing either binary or multi-class classifiers and that choosing the best classifier is the most crucial phase of the text classification pipeline. Fortunately, Schmidt and Wiegand (2017) assert that, as the features that different text classifiers consume tend to vary between classifiers, the predominant category into which most hate speech classifiers fall is that of supervised learning. In problems of hate speech detection, the most common classifiers are Support Vector Machines (SVMs), though Recurrent Neural Network Language Models (RNNLMs) are becoming more prominent. Other learning methodologies include Naïve Bayes (NB), Decision Trees (DTs), Random Forests (RFs), and *k*-Nearest Neighbors (*k*-NNs). This paper describes SVMs since they are the most common classifier to predict abusive content online.

Support Vector Machines (SVMs)

Fenner (2019) describes SVMs as the nonlinear extensions of Support Vector Classifiers (SVCs), which use *kernels* to construct features automatically. The heart of both SVCs and SVMs involves (a) finding the support vectors that maximize the margin separation between examples and (b) minimizing the training errors. Kernels redescribe the data in terms of its relationship between examples rather than its relationship between features, providing a versatile way to automate feature construction. Learners like SVMs that implement kernels do not require data in tabular formats and process strings, trees, and sets.

Kernels

Talpur and O'Sullivan (2020) used SVMs with *Radial Basis Function* (RBF) kernels, also called *Gaussian* kernels, to detect the severity of CB in tweets. Fenner (2019) writes that

using an RBF kernel is equivalent to projecting the examples of a dataset into an infinite-dimensional space, meaning that for as many new columns as the implementation can add, there are always more to be added. However, as observations move further away from each other, their similarities quickly decrease. The *gamma* parameter (γ) controls how far the influence of an example travels, and it is related to the inverse of the standard deviation (σ) and variance (σ^2): $\gamma = \frac{1}{2\sigma^2}$. A small value for *gamma* means a big variance in the data, which, in turn, means that things far apart from each other can still be strongly related.

Hyperparameters

Claesen and De Moor (2015) describe hyperparameters as a group of algorithm characteristics that must be set appropriately to maximize the effectiveness of any ML approach. Hyperparameters control an algorithm's *bias-variance trade-off*, a key balancing act in ML that describes choosing an appropriate level of model complexity. Further, Anguita *et al.* (2003) write that the design of SVMs requires tuning hyperparameters, which affect the classifier's ability to generalize on new, unseen data. Ways to find the best combination of hyperparameter settings are known as *searches*, which are often performed either manually or by testing hyperparameter sets on pre-defined grids.

The Grid and Random Searches

Fenner (2019) describes two types of searches to tune hyperparameters: (a) the grid search and (b) the random search. The grid search performs an exhaustive search of all hyperparameter combinations, whereas the random search samples a given number of candidates from a hyperparameter space using specified distributions. If many hyperparameters exist, or the range of values for a hyperparameter is large, random searches provide a more practical approach than grid searches. Typical hyperparameters to set in SVMs include *C*, *kernel*, and *gamma*. Fenner advises using RBF kernels when a dataset's observations exceed its features, as

is often the case with text classification problems. Users need to select values for *gamma* and *C* when using RBF kernels. *C* represents the model's *regularization*, the total weight of its complexity penalty. The regularization penalty for SVMs in scikit-learn is the squared L_2 penalty, the strength of which decreases as *C* increases (*3.2. Tuning the Hyper-Parameters of an Estimator*, n.d.). Therefore, SVMs intending to identify abusive content from online texts might benefit from implementing the grid or random searches described above, depending on the user-defined hyperparameter search spaces.

Conclusion

This paper described how feature engineering might be used to enhance the performance of a hypothetical text classifier to identify abusive content online. Methods discussed included feature extraction, selection, and construction, including the BOW approach, the removal of stopwords and stemming, character- and word-level representations, TF-IDF representations, and kernel methods to generate features automatically. The paper also discussed features that will boost the performance of hate speech detection algorithms, including network-based features, like users' number of followers, and content-based features, like profane words and POS tagging. Finally, the paper described SVMs, the most popular classifiers for detecting abusive text content, their methods for making predictions, their hyperparameters, and the techniques for tuning them.

References

3.2. Tuning the hyper-parameters of an estimator. (n.d.). Scikit-Learn. Retrieved September 25, 2021, from https://scikit-learn/stable/modules/grid_search.html

Anguita, D., Ridella, S., Rivieccio, F., & Zunino, R. (2003). Hyperparameter design criteria for Support Vector Machines. *Neurocomputing*. https://doi.org/10.1016/S0925-2312(03)00430-2

Biniz, M., Boukil, S., Adnani, F., Cherrat, L., & Moutaouakkil, A. (2018). Arabic Text Classification Using Deep Learning Technics. *International Journal of Grid and Distributed Computing*, *11*, 103–114. https://doi.org/10.14257/ijgdc.2018.11.9.09

Claesen, M., & De Moor, B. (2015). Hyperparameter Search in Machine Learning. *ArXiv:1502.02127 [Cs, Stat]*. http://arxiv.org/abs/1502.02127

Fenner, M. (2019). *Machine Learning in Python for Everyone*. Addison-Wesley.

Levy, O., & Goldberg, Y. (2014). Dependency-Based Word Embeddings. *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 302–308. https://doi.org/10.3115/v1/P14-2050

Martins, C., & Matsubara, E. (2003). *Reducing the dimensionality of bag-of-words text representation used by learning algorithms*.

Mehdad, Y., & Tetreault, J. (2016). Do Characters Abuse More Than Words? *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, 299–303. https://doi.org/10.18653/v1/W16-3638

Schmidt, A., & Wiegand, M. (2017). A Survey on Hate Speech Detection using Natural Language Processing. *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media*, 1–10. https://doi.org/10.18653/v1/W17-1101

Scott, S., & Matwin, S. (1999). Feature engineering for text classification. *ICML*, *99*, 379–388.

Talpur, B. A., & O'Sullivan, D. (2020). Multi-Class Imbalance in Text Classification: A Feature Engineering Approach to Detect Cyberbullying in Twitter. *Informatics*, *7*(4), 52.