Neural network approach to classifying alarming student responses to online assessment

Abstract

Automated scoring engines are increasingly being used to score the free-form text responses that students give to questions. Such engines are not designed to appropriately deal with responses that a human reader would find alarming such as those that indicate an intention to self-harm or harm others, responses that allude to drug abuse or sexual abuse or any response that would elicit concern for the student writing the response. Our neural network models have been designed to help identify these anomalous responses from a large collection of typical responses that students give. The responses identified by the neural network can be assessed for urgency, severity, and validity more quickly by a team of reviewers than otherwise possible. Given the anomalous nature of these types of responses, our goal is to maximize the chance of flagging these responses for review given the constraint that only a fixed percentage of responses can viably be assessed by a team of reviewers.

Introduction

Automated Essay Scoring (AES) and Automated Short Answer Scoring (ASAS) has become more prevalent among testing agencies BIBREF0 , BIBREF1 , BIBREF2 , BIBREF3 . These systems are often designed to address one task and one task alone; to determine whether a written piece of text addresses a question or not. These engines were originally based on either hand-crafted features or term frequency–inverse document frequency (TF-IDF) approaches BIBREF4 . More recently, these techniques have been superseded by the combination of word-embeddings and neural networks BIBREF5 , BIBREF6 , BIBREF7 . For semantically simple responses, the accuracy of these approaches can often be greater than accuracy of human raters, however, these systems are not trained to appropriately deal with the anomalous cases in which a student writes something that elicits concern for the writer or those

around them, which we simply call an `alert'. Typically essay scoring systems do not handle alerts, but rather, separate systems must be designed to process these types of responses before they are sent to the essay scoring system. Our goal is not to produce a classification, but rather to use the same methods developed in AES, ASAS and sentiment analysis BIBREF8 , BIBREF9 to identify some percentage of responses that fit patterns seen in known alerts and send them to be assessed by a team of reviewers.

Assessment organizations typically perform some sort of alert detection as part of doing business. In among hundreds of millions of long and short responses we find cases of alerts in which students have outlined cases of physical abuse, drug abuse, depression, anxiety, threats to others or plans to harm themselves BIBREF10 . Such cases are interesting from a linguistic, educational, statistical and psychological viewpoint BIBREF11 . While some of these responses require urgent attention, given the volume of responses many testing agencies deal with, it is not feasible to systematically review every single student response within a reasonable time-frame. The benefits of an automated system for alert detection is that we can prioritize a small percentage which can be reviewed quickly so that clients can receive alerts within some fixed time period, which is typically 24 hours. Given the prevalence of school shootings and similarly urgent situations, reducing the number of false positives can effectively speed up the review process and hence optimize our clients ability to intervene when necessary.

As a classification problem in data science, our problem has all the hallmarks of the most difficult problems in natural language processing (NLP) BIBREF12 ; alerts are anomalous in nature making training difficult, the data is messy in that it contains misspellings (both misused real words and incorrectly spelled words) BIBREF13 , students often use student specific language or multi-word colloquialisms BIBREF14 and the semantics of alerts can be quite complex and subtle, especially when the disturbing content is implicit rather than explicit. The responses themselves are drawn from a wide range of free-form text responses to questions and student comments from a semantically diverse range of topics, including many that are emotive in nature. For example, the semantic differences between an essay on

gun-control and a student talking about getting a gun can be very subtle. Sometimes our systems include essays on emotive topics because the difference in language between such essays and alerts can be very small. Students often use phrases like “kill me now" as hyperbole out of frustration rather than a genuine desire to end ones life, e.g., "this test is so boring, kill me now". To minimize false positives, the engine should attempt to evaluate context, not just operate on key words or phrases.

When it comes to neural network design, there are two dominant types of neural networks in NLP; convolutional neural networks (CNN) and recurrent neural networks (RNN) BIBREF15 . Since responses may be of an arbitrary length different recurrent neural networks are more appropriate tools for classifying alerts BIBREF16 . The most common types of cells used in the design of recurrent neural networks are Gated Recurrent Units (GRU)s BIBREF17 and Long-Short-Term-Memory (LSTM) units BIBREF18 . The latter were originally designed to overcome the vanishing gradient problem BIBREF19 . The GRU has some interesting properties which simplify the LSTM unit and the two types of units can give very similar results BIBREF20 . We also consider stacked versions, bidirectional variants BIBREF21 and the effect of an attention mechanism BIBREF22 . This study has been designed to guide the creation of our desired final production model, which may include higher stacking, dropouts (both regular and recurrent) and may be an ensemble of various networks tuned to different types of responses BIBREF23 . Similar comparisons of architectures have appeared in the literature BIBREF24 , BIBREF7 , however, we were not able to find similar comparisons for detecting anomalous events.

In section SECREF2 we outline the nature of the data we have collected, a precise definition of an alert and how we processed the data for the neural network. In section SECREF3 we outline the definition of the models we evaluate and how they are defined. In section SECREF4 we outline our methodology in determining which models perform best given representative sensitivities of the engine. We attempt to give an approximation of the importance of each feature of the final model.

Defining the Data

The American Institutes for Research tests up to 1.8 million students a day during peak testing periods. Over the 2016–2017 period AIR delivered 48 million online tests across America. Each test could involve a number of comments, notes and long answer free-form text responses that are considered to be a possible alerts as well as equations or other interactive items that are not considered to be possible alerts. In a single year we evaluate approximately 90 million free-form text responses which range anywhere from a single word or number to ten thousand word essays. These responses are recorded in html and embedded within an xml file along with additional information that allows our clients to identify which student wrote the response. The first step in processing such a response is to remove tags, html code and any non-text using regular expressions.

To account for spelling mistakes, rather than attempt to correct to a vocabulary of correctly spelled words, we constructed an embedding with a vocabulary that contains both correct and incorrectly spelled words. We do this by using standard algorithms BIBREF25 on a large corpus of student responses (approximately 160 million responses). The embedding we created reflects the imperfect manner in which students use words BIBREF26 . For example, while the words 'happems' and 'ocures' are both incorrectly spelled versions of 'happens' and 'occurs' respectively, our embedding exhibits a high cosine similarity between the word vectors of the correct and incorrect versions. The embedding we created was an embedding into 200 dimensional space with a vocabulary consisting of 1.12 million words. Using spelling dictionaries we approximate that the percentage of correctly spelled words in the vocabulary of this embedding is approximately 7%, or roughly 80,000 words, while the remaining 93% are either misspellings, made up words or words from other languages. Lastly, due to the prevalence of words that are concatenated (due to a missing space), we split up any word with a Levenstein distance that is greater than two from our vocabulary into smaller words that are in the vocabulary. This ensures that any sentence is tokenized into a list of elements, almost all of which have valid embeddings.

In our classification of alerts, with respect to how they are identified by the team of reviewers, we have two tiers of alerts, Tier A and Tier B. Tier A consists of true responses that are alarming and require urgent attention while Tier B consists of responses that are concerning in nature but require further review. For simplification, both types of responses are flagged as alerts are treated equivalently by the system. This means the classification we seek is binary. Table TABREF1 and Table TABREF2 outline certain subcategories of this classification in addition to some example responses.

The American Institutes for Research has a hand-scoring team specifically devoted to verifying whether a given response satisfies the requirements of being an alert. At the beginning of this program, we had very few examples of student responses that satisfied the above requirements, moreover, given the diverse nature of what constitutes an alert, the alerts we did have did not span all the types of responses we considered to be worthy of attention. As part of the initial data collection, we accumulated synthetic responses from the sites Reddit and Teen Line that were likely to be of interest. These were sent to the hand-scoring team and assessed as if they were student responses. The responses pulled consisted of posts from forums that we suspected of containing alerts as well as generic forums so that the engine produced did not simply classify forum posts from student responses. We observed that the manner in which the students engaged with the our essay platform in cases of alerts mimicked the way in which students used online forums in a sufficiently similar manner for the data to faithfully represent real alerts. This additional data also provided crucial examples of classes of alerts found too infrequently in student data for a valid classification. This initial data allowed us to build preliminary models and hence build better engines.

Since the programs inception, we have greatly expanded our collection of training data, which is summarized below in Table TABREF3 . While we have accumulated over 1.11 million essay responses, which include many types of essays over a range of essay topics, student age ranges, styles of writing as well as a multitude of types of alerts, we find that many of them are mapped to the same set of words

after applying our preprocessing steps. When we disregard duplicate responses after preprocessing, our training sample consists of only 866,137 unique responses.

Our training sample has vastly over-sampled alerts compared with a typical responses in order to make it easier to train an engine. This also means that a typical test train split would not necessarily be useful in determining the efficacy of our models. The metric we use to evaluate the efficacy of our model is an approximation of the probability that a held-out alert is flagged if a fixed percentage of a typical population were to be flagged as potential alerts.

This method also lends itself to a method of approximating the number of alerts in a typical population. we use any engine produced to score a set of responses, which we call the threshold data, which consisted of a representative sample of 200,014 responses. Using these scores and given a percentage of responses we wish to flag for review, we produce a threshold value in which scores above this threshold level are considered alerts and those below are normal responses. This threshold data was scored using our best engine and the 200 responses that looked most like alerts were sent to be evaluated by our hand-scorers and while only 14 were found to be true alerts. Using the effectiveness of the model used, this suggests between 15 and 17 alerts may be in the entire threshold data set. We aggregated the estimates at various levels of sensitivity in combination with the efficacy of our best model to estimate that the rate of alerts is approximately 77 to 90 alerts per million responses. Further study is required to approximate what percentage are Tier A and Tier B.

Recurrent Structures Considered

Since natural languages contain so many rules, it is inconceivable that we could simply list all possible combinations of words that would constitute an alert. This means that the only feasible models we create are statistical in nature. Just as mathematicians use elementary functions like polynomials or periodic

functions to approximate smooth functions, recurrent neural networks are used to fit classes of sequences. Character-level language models are typically useful in predicting text BIBREF27 , speech recognition BIBREF28 and correcting spelling, in contrast it is generally accepted that semantic details are encoded by word-embedding based language models BIBREF29 .

Recurrent neural networks are behind many of the most recent advances in NLP. We have depicted the general structure of an unfolded recurrent unit in figure FIGREF4 . A single unit takes a sequence of inputs, denoted INLINEFORM0 below, which affects a set of internal states of the node, denoted INLINEFORM1 , to produce an output, INLINEFORM2 . A single unit either outputs a single variable, which is the output of the last node, or a sequence of the same length of the input sequence, INLINEFORM3 , which may be used as the input into another recurrent unit.

A layer of these recurrent units is a collection of independent units, each of which may pick up a different aspect of the series. A recurrent layer, consisting of INLINEFORM0 independent recurrent units, has the ability to take the most important/prevalent features and summarize those features in a vector of length INLINEFORM1 . When we feed the sequence of outputs of one recurrent layer into another recurrent layer, we call this a stacked recurrent layer. Analogous to the types of features observed in stacking convolutional and dense layers in convolutional neural networks BIBREF30 , it is suspected that stacking recurrent layers allows a neural network to model more semantically complex features of a text BIBREF31 , BIBREF32 .

The collections of variables associated with the state of the recurrent units, which are denoted INLINEFORM0 in figure FIGREF4 , and their relations between the inputs, INLINEFORM1 , and the outputs are what distinguishes simple recurrent units, GRUs and LSTM units. In our case, INLINEFORM2 is a sequence of word-vectors. The underlying formulas for gated recurrent units are specified by the initial condition INLINEFORM3 and $z_t = g\left(W_z x_t + U_z h_{t-1} + b_z\right)$,

rt = g (Wr xt + Ur ht-1 + br),

ht = zt ht-1 + zt yt,

yt = h (Wh xt + Uh(rt ht-1) + bh), where INLINEFORM0 denotes the element-wise product (also known as the Hadamard product), INLINEFORM1 is an input vector INLINEFORM2 is an output vector, INLINEFORM3 is and update gate, INLINEFORM4 , INLINEFORM5 is a reset gate, subscripted variables INLINEFORM6 , INLINEFORM7 and INLINEFORM8 are parameter matrices and a vector and INLINEFORM9 and INLINEFORM10 are the original sigmoid function and hyperbolic tangent functions respectively BIBREF17 .

The second type of recurrent unit we consider is the LSTM, which appeared in the literature before the GRU and contains more parameters BIBREF18 . It was created to address the vanishing gradient problem and differs from the gated recurrent unit in that it has more parameters, hence, may be regarded as more powerful. ft = g (Wf xt + Uf ht-1 + bf),

it = g (Wi xt + Ui ht-1 + bi),

ot = g (Wo xt + Uo ht-1 + bo),

ct = ft ct-1 + it yt,

ht = ot h(ct),

yt = h (Wz xt + Uz ht-1 + bz), where INLINEFORM0 is the input, INLINEFORM1 is the cell state vector, INLINEFORM2 is the forget gate, INLINEFORM3 is the input gate, INLINEFORM4 is the output gate and

INLINEFORM5 is the output, INLINEFORM6 is a function of the input and previous output while subscripted variables INLINEFORM7 , INLINEFORM8 and INLINEFORM9 are parameter matrices and a vector. Due to their power, LSTM layers are ubiquitous when dealing with NLP tasks and are being used in many more contexts than layers of GRUs BIBREF33 .

Given a recurrent unit, the sequence INLINEFORM0 is fed into the recurrent unit cell by cell in the order it appears, however, it was found that some recurrent networks applied to translation benefited from reversing the ordering of the sequence, so that the recurrent units are fed the vectors from last to first as opposed to first to last. Indeed, it is possible to state the most important information at the beginning of a text or at the end. The idea behind bidirectional recurrent units is that we double the number of set units and have half the units fed the sequence in the right order, while the other half of the units are fed the sequence in reverse. Due to the lack of symmetry in the relations between states, we are potentially able to model new types of sequences in this way.

The last mechanism we wish to test is an attention mechanism BIBREF22 . The key to attention mechanisms is that we apply weights to the sequences, INLINEFORM0 , outputted by the recurrent layer, not just the final output. This means that the attention is a function of the intermediate states of the recurrent layer as well as the final output. This may be useful when identifying when key phrases are mentioned for example. This weighted sequence is sent to a soft-max layer to create a context vector. The attention vector is then multiplied by INLINEFORM1 to produce resulting attention vector, INLINEFORM2 . We have implemented the following attention mechanism ht = ct ht,

ct = t,j hj,

i,j = (eij)(eik), where INLINEFORM0 was the output from the LSTM layer, the INLINEFORM1 are linear transformations of the INLINEFORM2 and INLINEFORM3 is the attended output, i.e., the output of the

attention layer . This mechanism has been wildly successful in machine translation BIBREF34 , BIBREF35 and other tasks BIBREF36 .

Methodology and Results

Unlike many tasks in NLP, our goal is not to explicitly maximize accuracy. The framework is that we may only review a certain percentage of documents, given this, we want to maximize the probability than an alert will be caught. I.e., the cost of a false-positive is negligible, while we consider false negatives to be more serious. Conversely, this same information could be used to set a percentage of documents required to be read in order to have have some degree of certainty that an alert is flagged. If we encode all alerts with the value 1 and all normal documents with a value of 0, any neural network model will serve as a statistical mechanism in which an alert that was not used in training will, a priori, be given a score by the engine from a distribution of numbers between 0 and 1 which is skewed towards 1 while normal documents will also have scores from another distribution skewed towards 0. The thresholds values where we set are values in which all scores given by the engine above the cut-off are considered possible alerts while all below are considered normal. We can adjust the number of documents read, or the percentage of alerts caught by increasing or decreasing this cut-off value.

To examine the efficacy of each model, our methodology consisted of constructing three sets of data:

The idea is that we use the generic test responses to determine how each model would score the types of responses the engine would typically see. While the number of alerts in any set can vary wildly, it is assumed that the set includes both normal and alert responses in the proportions we expect in production. Our baseline model is logistic regression applied to a TF-IDF model with latent semantic analysis used to reduce the representations of words to three hundred dimensions. This baseline model performs poorly at lower thresholds and fairly well at higher thresholds.

To evaluate our models, we did a 5-fold validation on a withheld set of 1000 alerts. That is to say we split our set into 5 partitions of 200 alerts, each of which was used as a validation sample for a neural network trained on all remaining data. This produced five very similar models whose performance is given by the percentage of 1000 alerts that were flagged. The percentage of 1000 alerts flagged was computed for each level of sensitivity considered, as measured by the percentage of the total population flagged for potentially being an alert.

Each of the models had 512 recurrent units (the attention mechanisms were not recurrent), hence, in stacking and using bidirectional variants, the number of units were halved. We predominantly trained on using Keras with Tensorflow serving the back-end. The machines we used had NVIDIA Tesla K80s. Each epoch took approximately two to three hours, however, the rate of convergence was such that we could restrict our attention to the models formed in the first 20 epochs as it was clear that the metrics we assessed had converged fairly quickly given the volume of data we had. The total amount of GPU time spent on developing these models was in excess of 4000 hours.

To give an approximation of the effect of each of the attributes we endowed our models with, we can average over the effectiveness of each model with and without each attribute in question. It is clear that that stacking two layers of recurrent units, each with half as many cells, offers the greatest boost in effectiveness, followed by the difference in recurrent structures followed by the use of attention. Using bidirectional units seems to give the smallest increase, but given the circumstances, any positive increase could potentially save lives.

Conclusions

The problem of depression and violence in our schools is one that has recently garnered high levels of media attention. This type of problem is not confined to the scope of educational research, but this type of

anomaly detection is also applicable to social media platforms where there are posts that indicate potential cases of users alluding to suicide, depression, using hate-speech and engaging in cyberbullying. The program on which this study concerns is in place and has contributed to the detection an intervention of cases of depression and violence across America. This study itself has led to a dramatic increase in our ability to detect such cases.

We should also mention that the above results do not represent the state-of-the-art, since we were able to take simple aggregated results from the models to produce better statistics at each threshold level than our best model. This can be done in a similar manner to the work of BIBREF23 , however, this is a topic we leave for a future paper. It is also unclear as to whether traditional sentiment analysis provides additional information from which better estimates may be possible.

Acknowledgements

I would like to thank Jon Cohen, Amy Burkhardt, Balaji Kodeswaran, Sue Lottridge and Paul van Wamelen for their support and discussions.