Proactively Discouraging Cyberbullying

Natural Language Processing

Michael Kossyrev, ID:mkossyre

December 13, 2017

1 Motivation

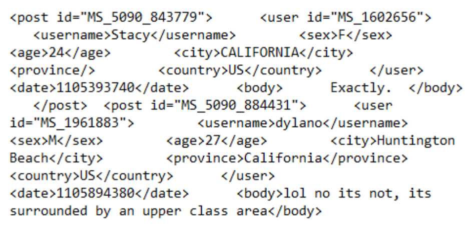
As of 2012, over 3 million kids per month are absent from school, and approximately 4,500 kids kill themselves each year at least in part because of cyberbullying [1]. Having a significant other who's a teacher, I hear first-hand about how seriously her kids are impacted by bullying, both in the real world and the virtual one. The hypothesis is that if we monitor conversations as they occur, we can issue reminders to students to reduce their harassment, and even take preventative action (such as locking them out of their account) if the harassment is persistent or severe enough.

I used Myspace conversations to train an SVM to predict whether there was bullying or not. I then implemented the SVM in a simulator as an on-the-fly monitor, as well as making it able to re-learn using new data.

2 Modeling

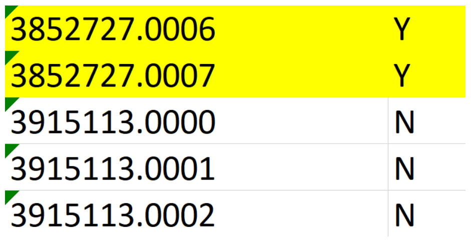
2.1 Data

The data used was a subset of the data gathered for the poster presentation by Bayzick [2]. This included 2,044 XML files, each one containing 10 lines of dialogue between users, as well as meta-data like age, sex, and username. The data also contained 11 XML files labeling the conversations as containing bullying or not. A sample of each type is shown below.



```
<post id="MS_5090_843779">      <user id="MS_1602656">
   <username>Stacy</username>        <sex>F</sex>
<age>24</age>        <city>CALIFORNIA</city>
<province/>        <country>US</country>      </user>
<date>1105393740</date>      <body>      Exactly.  </body>
   </post>  <post id="MS_5090_884431">      <user
id="MS_1961883">        <username>dylano</username>
<sex>M</sex>        <age>27</age>        <city>Huntington
Beach</city>        <province>California</province>
<country>US</country>      </user>
<date>1105894380</date>      <body>lol no its not, its
surrounded by an upper class area</body>
```

Figure 1: Sample input data.



3852727.0006	Y
3852727.0007	Y
3915113.0000	N
3915113.0001	N
3915113.0002	N

Figure 2: Sample input labels.

The data was pre-processed to remove the meta-data, leaving only the sentences to be used by the feature extractor. An additional pre-processing step was attempted to process the speech into a more coherent conversation using the python autocorrect[1] library, however, this processing took too much time, and so was removed. Two more data refining features were used. The first was implementing the standard string.printable filter from the string library on the input data. The second was refinement was removing all words longer than 15 characters. These tended to be either nonsense or urls. All of these features (including the failed autocorrect) were implemented with the hope of decreasing the feature space by decreasing uninformative variance in the data. Hate and "haet" mean the same thing.

[1]https://pypi.python.org/pypi/autocorrect/0.3.0

2.2 Feature Extractor

Several n-gram feature extractors were tested: 1 and 2-word word, as well as 2,3, and 4-character. The logic behind the first set of feature extractors is that bullying often manifests itself in short phrases like "you're stupid" or "go die". Bullies don't tend to go on eloquent diatribes against their victims. The character feature selectors were implemented to see if any significant information lies hidden in small character subsets. For example, maybe the letters "fu" could indicate a high likelihood of bullying.

First, a corpus of words was created by quickly parsing through all of the files. Then, all sentences were transformed into their vector representation by finding their words' locations in the corpus, and creating a sparse row vector from it. All of these were then stacked into one matrix. An example is shown in Figure 3.

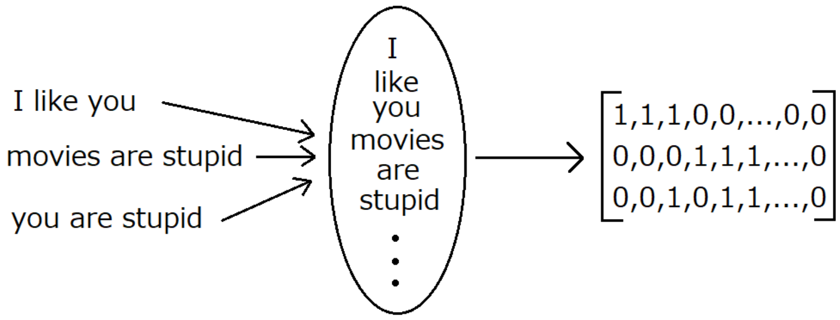


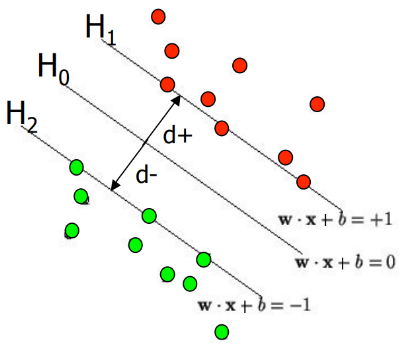
Figure 3: Sample of feature extraction.

2.3 Models

To train the models, the data was split 64-16-20 as training-dev-test [2], and shuffled. Two different scikit-learn SVM models were trained and dev-tested: a linear SVC (Support Vector Classifier) as a baseline and a NuSVC.

2.4 SVMs - Brief Overview

An SVM classifies data by constructing a hyperplane with the largest margin, where margin is the closest distance between plane and nearest points to it on either side (known as support vectors). At optimum, it can be shown [3] that margin $\propto ||w||^{-1}$, which leads to quadratic optimization:



$$\begin{aligned} &\text{minimize} && \frac{1}{2}||w||^2 \\ &\text{s.t.} && y_i(w \cdot x_i + b) = 1 \end{aligned}$$

Figure 4: SVM hyperplane in 2D [3].

[2]64% came by accident; the data was first divided 80-20 train-test, then the train was divided 80-20 train-dev. This was deemed an acceptable ratio

[3]http://web.mit.edu/6.034/wwwbob/svm-notes-long-08.pdf

This can be solved using Lagrange multipliers, then solving the dual problem of the one that appears. In the end, the following equations come out:

$$\max_{\alpha} W(\alpha) = \sum_{i}^{m} \alpha_i - \frac{1}{2}\sum_{i,j}^{m} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle$$
$$\text{s.t.} \quad 0 \le \alpha_i \le C$$
$$\sum_{i}^{m} \alpha_i y_i = 0$$
$$w = \sum_{i}^{m} \alpha_i y_i x_i$$

Note that most of the α_i's will actually end up being 0, with the ones that aren't corresponding to the support vectors. Ultimately, w is what we care about, as that is what we will use to test new data, and that gets reconstructed from the scaled support vectors.

2.5 SVM Models

The linear SVC had the advantage of being very quick to implement, however, as shown in the results, the accuracy was not acceptable. Because of this, a NuSVC model was adopted. This model is identical to the SVC, except it uses a Nu term to force a certain percentage of the data to be support vectors. Naturally, a larger Nu yields more vectors, thereby better results on the training data. However, this will also lead to over-fitting and high variance when comparing against unknown data. The SVM was manually tuned to have a Nu factor of 5%. This gave good results on the testing and dev data, while still being deemed "low enough" to generalize well.

3 Results

The main metrics of success were prediction accuracy ($\frac{correct}{total}$), and bullying accuracy ($1 - \frac{guess=0 \wedge truth=1}{total}$). Bullying accuracy refers to the number of times the SVM missed a time when bullying occurred. The thinking behind this is that it's better to be aggressive about identifying bullying, and be wrong some of the time, than it is to miss a potential bullying threat.

Gram type	Feature Extraction Time (s)	SVC Dev Accuracy	5% NuSVC Dev Accuracy	5% NuSVC Bullying Dev Accuracy
2-char	92.1	46%	95%	97%
3-char	695.5	51%	95%	96%
4-char	2856	46%	96%	97%
1-word	62.8	70%	96%	99%
2-word	291.3	43%	95%	97%

Table 1: 5% NuSVC gave 95% accuracy and 98% bullying accuracy on the test set.

Of note is that the feature extraction took seemingly polynomial longer time going from 2-char to 3-char and 4-char (90 seconds to 12 minutes to 50 minutes). This was surprising, given that in theory the code would scan less 4-char combinations per example, but presumably, the slowdown is due to the fact that overall there are only $\sim 26^2$ 2-char combinations, whereas there are $\sim 26^4$ 4-char combinations. While a set was used in creating the feature list, creating data matrix required finding the index of the gram in a feature list, which is slow.

While I initially thought the character extractors would do better, this turned out to not be the case. One possible explanation is that people use a lot of slang, misspelling, and l33tsp3ak on the internet, which confused the extractor more than it helped. Given that all of the extractors yielded roughly the same results, and that the 1-word extractor was fast and gave extremely good results, that one was set as the dedicated extractor for the SVM.

However, by far the biggest difference came from changing the model to the NuSVC. This increased accuracy by at least 20%, even over the best SVC model. The logic behind this is that a lack of data led to underfitting when just using the SVC, so forcing more of the data to be supports bolstered the results greatly. Yet, since this model may be implemented in the field without a lot of data, the ability to re-learn would be very useful.

4 Adaptive Re-Learning

A simulator was created in which a user could talk to, and bully, themselves. After every sentence, the words would be transformed into an SVM-predictable vector (see: Feature Extractor), and the SVM would either output "bully" or nothing at all. This was a failure, as the machine predicted a simple "hello" as bullying behavior.

The model was adjusted to include a cross-validation probability output [4]. As expected, the probability of "hello" being bullying was just 51%. Given that the model was trained on 10 lines of text per data point, the model would naturally have a hard time predicting a super-sparse data point of just 1 dimension. The simulator was adjusted to 60% confidence or higher, which gave much more reasonable outputs of when bullying occurred.

Still, this was not enough. While users were no longer erroneously classified as bullies, the machine regularly missed instances like "you are stupid". To correct this, the model and simulator were adjusted one final time: After each time a message is sent, if a user believes the machine misclassified, they can input a special command ("s bully") to tell it that the previous line was a bullying instance. Correcting false positives was also implemented similarly. Then, after the conversation was finished, the model would first build up its corpus to include any new words it saw. Then, it would add the new data to its training data, and re-train. A sample conversation is shown below.

```
Have a conversation with yourself!
you are dumb
s bully
you suck
s bully
Done
Number of features added to the list: 1
Fitting new model...
Done
>>>
Have a conversation with yourself!
you are dumb
Serious bullying detected!
you
Some bullying detected. Please moderate your language.
```

As a note, user-classification is clearly prone to abuse. This simulator was built as a demonstration, and in real life, the classification would be done by forum administrators or the like.

The results were encouraging, though far from ideal. After user-classifying "you are dumb", for example, the machine was able to accurately predict that that was bullying, however it now also predicted "you" as an offense. I believe this is because of a data mismatch between the training data and simulator data. The machine was trained on conversations, whereas the application was on a sentence-by-sentence basis. This resulted in inaccurate predictions on what is essentially a different data type. However, the results point to the possibility that this system could eventually learn bullying in short sentences, despite being trained on conversations.

Also, severity was not at all accurate. Hate-comments like "I'm going to find you and kill you" were marked as mild bullying, while "I don't like you today" was marked as severe. This is probably because SVM prediction strength ultimately does not correlate to bullying severity.

5 Conclusion/Future Work

The project achieved its stated goal of classifying bullying in conversations. Compensating for a lack of data by adding a Nu term to the SVC pushed accuracy to above 90%, and the made the machine worth using over human readers.

As the most novel part of the project, I would like to more thoroughly explore the on-the-fly re-learning in the future. Having a model adapt to a new data mean (from paragraphs to sentences, as an example) could be a powerful paradigm with many other applications. In particular, training with "close enough" data that is easy to acquire, and reinforcing with real world data is often-times much easier and cheaper to do than getting many real-world samples. An example that comes to mind is Lunar soil experiments, which have many synthetics, but not a lot of original soil to work with. I would also like to test my hypothesis that the Nu term compensated for a lack of data by decreasing it as more data was acquired over time.

Finally, it would be quite interesting to run NLP sentiment analysis on the data. If we could use NLP to show that bullying was correlated to strong emotions, like anger and fear, a lot of innocent conversations could be correctly classified just by noticing that the emotions are not present. We could potentially reduce the feature space of our data to just emotions and a few identifier words, which could lead to better scalability (as SVMs tend to be slower to train) and hopefully better predictions based on deeper insights in the data.

References

[1] *Cyberbullying: How Bullies Have Moved From the Playground to the Web.* OnlineCollege.org. 2012

[2] J. Bayzick, A. Kontostathis, and L. Edwards, "Detecting the Presence of Cyberbullying Using Computer Software." Poster presentation at WebSci11, June 14-17, 2011, Koblenz Germany.

[3] A. Ng, "Support Vector Machines". http://cs229.stanford.edu/notes/cs229-notes3.pdf

[4] "sklearn.svm.NuSVC: predict_proba".
http://scikit-learn.org/stable/modules/generated/sklearn.svm.NuSVC.html

[5] F. Pedregosa *et al.* “Scikit-learn: Machine Learning in Python,” *Journal of Machine Learning Research* 12, pp. 2825-2830, 2011.