Ross Beck-MacNeil
September 27th, 2017

Capstone Proposal

1 - Domain Background

My project draws upon the domain of document classification with an emphasis on the natural language processing component of document classification. Machine learning has been successfully used to classify documents by topic for several decades (Sebastiani 2002). However, machine learning techniques do not perform as well when performing sentiment analysis which requires the parsing of more complex language structures (Bo et al. 2002). This is where ideas from natural language processing must be applied.

My aim with this project is to make progress towards solving a complex natural language processing problem using machine learning. I think that it would very beneficial if computers could interpret and produce the same kind of natural language of which even young children are capable. This would allow machine learning to be applied to a wider variety of tasks than it is currently capable of solving. In particular, machine learning could be applied to many problems that are not well structured and for which there is not much training data.

2 - Problem Statement

I will attempt to solve the problem of determining whether a post on the reddit r\jokes subreddit has received more than 10 net upvotes, conditional on its title and submission text. The criteria of 10 net upvotes can be considered a proxy for the true metric of interest, the funniness of the post. However, humor is very subjective and hard to quantify and therefore the net upvotes from the r/jokes subreddit is being used as a proxy for the funniness of the post.

This is a problem that reoccurs every hour of every day, as users are constantly posting new submission to r/jokes. Not only is the r/jokes subreddit constantly producing a new stream of problems to be solved, there is ample historical data so that a model can be trained and learned.

This will be a supervised binary classification problem and as such many different measurable evaluation metrics could be applied. These will be further elaborated in *Section 6 - Evaluation Metrics*. Given the title and text of a post, the solution will be a machine learning model that receives as input the text and tile of the post and outputs either a class prediction or a class probability. The solution will be further described in *Section 4 - Solution Statement.*

3 - Datasets and Inputs

The primary dataset that is being used contains all posts from the Reddit subreddit r/jokes that were submitted between January 25, 2008 and September 10th, 2017. I personally obtained this dataset by using the reddit API to query and retrieve these submissions.

The input dataset is in comma separated values (csv) format and is entitled *AllSubsFrom_rJokes_14_09_2017.csv*. The 14_09_2017 indicates the date that the scraping finished: September 14th, 2017. The data set, before any cleaning or processing, contains 319,747 observations. Each observation corresponds to a different reddit post. There are a total of 9 variables in this data sets; they are, as described in Table 1.

Table 1

Variable	Description
id	Unique identifier assigned by Reddit
date	Post submission date, in Unix epoch time (seconds since January 1, 1970)
downs	Number of downvotes. Due to reddit protections, always equal to 0
score	Net upvotes (fuzzed)
text	Text of submission/post
title	Title of submission/post
ups	Number of upvoter. Due to reddit protections, this is equal to the score variable
upvote_ratio	Upvote ratio, ratio of upvotes to total votes
url	Link to original post.

Given the scope of the problem, it is very appropriate to use this data set, as it conforms exactly to the problem statement from section 2. Per the specifications of the problem statement, a new *funny* variable will be created such that for *ith* observation:

$$funny_i := \begin{cases} 1\ if\ score_i \geq 10 \\ 0\ if\ score_i \leq 1 \end{cases}$$

Jokes with 1 < score < 10 are considered neutral, neither funny nor unfunny, and will not be considered by the model and will dropped from the dataset. This approach is similar to the one employed by Bo et al (2002) where they focus on discriminating between positive and negative movie reviews and do not consider neutral ones. This *funny* variable will be the target variable that the solution will attempt to predict. At times it may be referred to as *y*.

After removing neutral posts, 12.5% of the observations will be randomly selected into a holdout test set. This will the set of observation upon which the final model will be evaluated. The models will be developed only against the remaining 87.5% of observations. This separation between the train and test sets is very important and will be discussed more in section 6. further in *Section 6 - Evaluation Metrics*.

In addition to the dataset containing the reddit posts, datasets consisting of pretrained GloVe word embeddings will used. These datasets are available from https://nlp.stanford.edu/projects/glove/ (Pennington et al. 2014). Each dataset contains one line per word, where each line starts with the word and is followed the elements of its vector representation. The word and its elements are separated by spaces. In order to test the effect of using vector embeddings of different dimensions, three of these pre-trained word embeddings files will be used. The data sets that will be used will correspond to vectors of 50, 100 or 200 dimensions. It is expected that the pre-trained embeddings will allow for the development of models that account for sematicsemantic similarities while avoiding overfitting to the relatively small training set.

4 - Solution Statement

The solution to this problem will be an algorithm that takes as input the submission title and text for a set of observations and outputs a class prediction for each observation. This class prediction could be expressed as the variable $\widehat{funny}$ where $\widehat{funny} \epsilon \{1,0\}$. In addition to predicting class for each observation, it preferred that the algorithm also produce a confidence score for each prediction. This confidence score could be expressed as $\hat{y} \epsilon [1,0]$ where a score of 1 indicates that the algorithm has perfect confidence that the observation deserves a label of $\widehat{funny} = 1$ and a score of 0 indicates that the algorithm is perfectly confident that the observations should be labeled as $\widehat{funny} = 0$. These confidence scores could help diagnose the algorithm and allow it to be applied in different settings. However they are note absolutely crucial, only preferred.

5 - Benchmark Model

The simplest benchmark result is the class that occurs most frequently. This would be “predicted” The simplest benchmark model would be one that always predicts the most commonly occurring class. Such a model would output the same result for every submission without consideringregardless of its text.actual context. This is a very simple benchmark, but it is imperative that the final model beat this benchmark.

Due to the simplicity of the above benchmark, it is important to have second, more advanced, benchmark. A slightly more complicated benchmark would be a logistic linear model that takes as input a bag of word structedstructured into a document term matrix. As stated in Joulin et al. (2016) linear classifiers that use bag of words features are a strong baseline. Due to the large number of input features in bag of word models, regularization is necessary to prevent overfitting. Therefore, a penalty will be applied to the l2 norm of the coefficients, resulting in what is called *ridge* logistic regression. Since the strength of the penalty is a hyperparameter that is not learned from the training data, cross-fold validation will be to used find appropriate lambda (regularization strength).

6 - Evaluation Metrics

There are many different metrics that cancould be used to evaluate the solution, given that this is binary classification problem. The most common metric is accuracy. If $funny_i$ is the true value of observation *i*, such that

$$funny_i \epsilon 0{,}1 \{1{,}0\}.$$

and $\widehat{funny}_i$is the predicted value of the *i*-th observation, then the accuracy of the predicted values can be calculated as:

$$accuracy\left(funny, \widehat{funny}\right) = \frac{1}{N}\sum_{i=1}^{N} 1\left(funny_i = \widehat{funny}_i\right)$$

where *N* is the number of samples and $1\left(funny_i = \widehat{funny}_i\right)$ is the indicator function such that:

$$1\left(funny_i = \widehat{funny}_i\right) := \begin{cases} 1\, if\ funny_i = \widehat{funny}_i \\ 0\, if\ funny\ _i \neq \widehat{funny}_i \end{cases}$$

However, the accuracy metric can be misleading in certain circumstance. For one, it can be hard to interpret accuracy when one class occurs much more frequently. This is known as an unbalanced classification problem.

One way to evaluate the results of a binary classifier even with unbalanced classes is to calculate the precision and recall performance of the classifier on each class.

Need new terminology.

Define number of true positives, *tp*, as number of number of observations that are funny and are correctly classified as funny, i.e:

$tp = \sum_{i=1}^{N} 1\left(funny_i = \widehat{funny}_i\, \& funny_i = 1\right)$

Define number of false positives, *fp*, as number of number of observations that are not funny yet are incorrectly classified as funny, i.e:

$tp = \sum_{i=1}^{N} 1\left(funny_i \neq \widehat{funny}_i\ \&\ funny_i = 0\right)$

Log loss is useful., especially as will be using linear logistic model as baseline, which produces this by default. However can be hard to inpretret,.

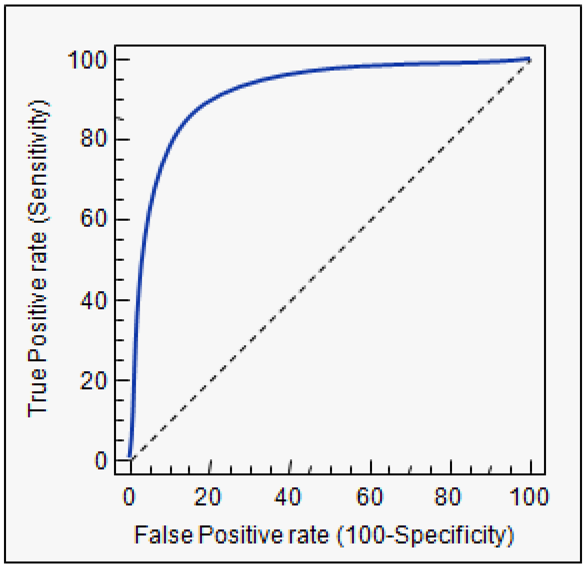
Log loss:

$$LogLoss(y, \hat{y}) = -\frac{1}{N}\sum_{i=1}^{N}[1(y_i = \hat{y}_i)]$$

In addition, it does not account for the confidence that the classifier has in its predictions. It seems reasonable to prefer a model that is more confident in its correct predictions and less confident in its incorrect predictions.

The Area Under the Receiver Operator Curve (AUROC) metric can overcome the shortcomings of the accuracy metric. The Receiver Operator Characteristic (ROC) is a graphical plot that illustrates the classification ability of a binary model as its discrimination threshold is varied. The The area under this curve (AUC of this curve) is the probability that a classifier will be more confident about randomly chosen positive example than a randomly chosen negative example. Figure 1 displays an example of an AUROC, where the blue line represents the ROC of the chosen classifier and the dashed line represents the results of a theoretical baseline classifier that predicts at random. The one drawback to the AUROC is that it can only be calculated for algorithms that output confidence scores. Algorithms such as support vector machines and k-nearest neighbors do not natively output such confidence scores. Therefore, both accuracy and the AUROC will be calculated when possible but for certain models it will only be possible to calculate accuracy.

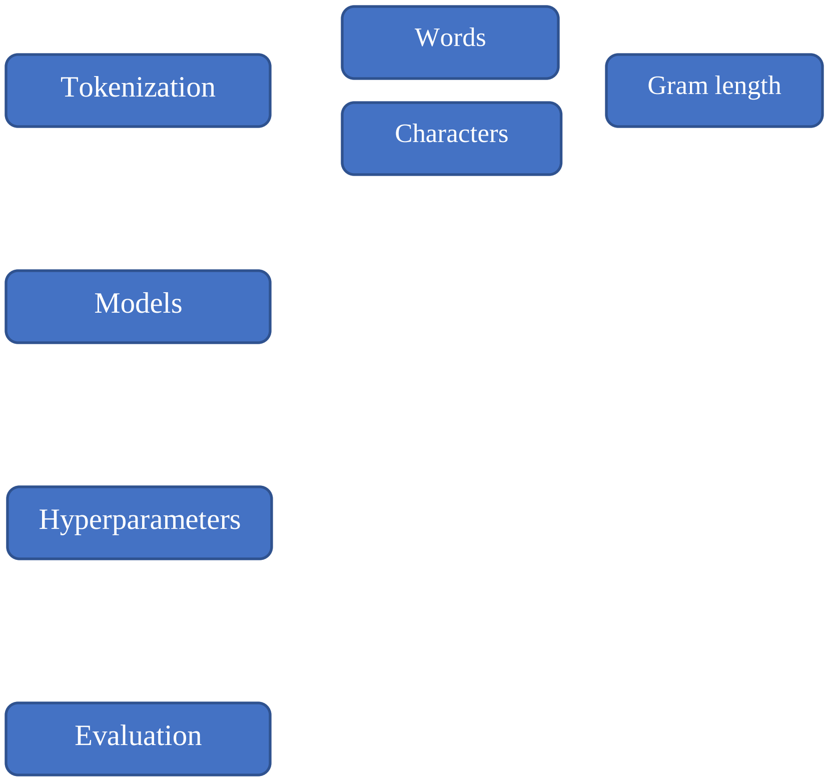
Figure 1 – Source: https://www.medcalc.org/manual/roc-curves.php



It is important to emphasis that proposed solution will be ultimately evaluated using its predictions on the hold-out test set. It is quite easy to develop a classifier that achieves perfect accuracy or AUROC on data that it has seen. It is quite another task to develop a classifier that does well on unseen data. However, care must be taken not to evaluate too many potential solutions against the hold-out set. Ideally model and hyperparametric selection will take place using cross-fold validation or a validation set that is different from the test set. Using the test set for these purposes could lead to overfitting in the same way as using to directly optimize the model parameters.

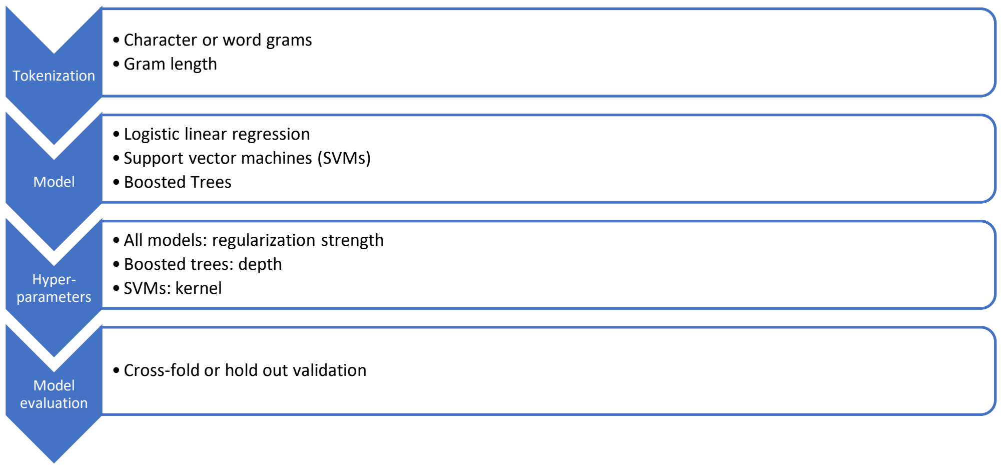
7 - Project Design

I plan to build my models in python using two different machine learning libraries. The first library, scikit-learn (sklearn), is a comprehensive machine learning library that includes many models and tools are useful in building a machine learning pipeline. I will use sklearn to develop and test several different models that use bag of words features, including the baseline logistic linear classifier. Please refer to Figure 2 for an outline of the workflow in sklearn.



Once I am satisfied that I have found the most performant bag of words classifier, I will focus on neural networks and deep learning techniques. For these approaches, I will the Keras interface to tensorflow. Keras and tensorflow are two separate libraries: tensorflow is the backend that performs the actual calculations while is Keras is a convenient frontend to tensorflow. I plan to investigate several neural network structures. The first approach that I will try will be the simple embedding model + averaging model proposed by Joulin et al. (2016). Afterwards, I will experiment with using the pre-trained Glove embeddings that were described in *Section 3 - Datasets and Inputs.* It is hoped that the use of these pre-trained word embeddings will allow for models that are more focused on learning about humor in general rather than learning the meaning of specific words. I will be employing these pretrained embeddings within recurrent neural networks (RNNs) that can learn long term dependencies. I also hope to be able to implement the hierarchal attention model of Yang, et al (2016).

Figure 2


Tokenization
• Character or word grams
• Gram length
Model
• Logistic linear regression
• Support vector machines (SVMs)
• Boosted Trees
Hyper-parameters
• All models: regularization strength
• Boosted trees: depth
• SVMs: kernel
Model evaluation
• Cross-fold or hold out validation

Citations

Joulin, Armand, et al. "Bag of tricks for efficient text classification." *arXiv preprint arXiv:1607.01759* (2016).

Pang, Bo, Lillian Lee, and Shivakumar Vaithyanathan. "Thumbs up?: sentiment classification using machine learning techniques." *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*. Association for Computational Linguistics, 2002.

Pennington, Jeffrey, Richard Socher, and Christopher Manning. "Glove: Global vectors for word representation." *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 2014.

Sebastiani, Fabrizio. "Machine learning in automated text categorization." *ACM computing surveys (CSUR)* 34.1 (2002): 1-47

Yang, Zichao, et al. "Hierarchical Attention Networks for Document Classification." *HLT-NAACL*. 2016.