CONTRASTING METHODS FOR CLASSIFYING MICROTEXT STATEMENTS CONTAINING MATHEMATICS

B. HASKINS and R.A. BOTHA
NELSON MANDELA METROPOLITAN UNIVERSITY

Abstract

Queries received by tutors on the Dr Math mathematics tutoring service are created in a domain-specific form of microtext. The aim of the service is to help South African school learners to master mathematical concepts, but not all of the queries received on the service contain content relevant to the tutoring process.

This paper contrasts various methods to classify learner queries automatically as relevant or not, in order to determine whether such a process could approximate human judgement. A back-propagation artificial neural network, a decision tree, a Bayesian filter, a k-means clustering algorithm and a rule-based filter are compared.

The results of the classification techniques are contrasted with the results of three human coders, using the metrics of precision, recall, F-measure and the Pearson correlation co-efficient. Both the rule-based filter and neural network deliver classification results which closely reflect the classifications made by the human coders.

Keywords: text processing, classification, microtext, mathematics, tutoring

1. INTRODUCTION

The widespread adoption of the cellular phone has made it possible to deliver services to people and places where they were impossible previously. This newfound connectivity has greatly increased the uptake of mobile phone applications, such as the chat application Mxit, among the South African youth.

Dr Math® is run as a service on Mxit and allows mathematics tutors to reach South African high school learners. The language used by the learners, when phrasing questions to the tutors, is problematic, as it contains a variety of short forms and abbreviations, commonly referred to as microtext. Microtext may be defined as the short snippets of text used in modern digital forms of communication (Hovy, Markman, Martell & Uthus, 2013). This form of text consists of various misspellings, informality, varied grammar and non-language forms, such as emoticons.

There is a great deficit with regard to the number of tutors compared to the number of school learners. This leads to situations where the available tutors may be inundated with learner queries during a given tutoring period. Unfortunately, the learners do not always realise the value of tutor time and try to start up conversations that are completely off topic. Therefore, this study is an investigation into various means which may be used to determine automatically whether individual queries are related to mathematics.

2. PROBLEM STATEMENT AND OBJECTIVES

Not all of the learner queries received on the Dr Math service are relevant to the tutoring process. Being able to sort the queries into those which contain mathematical equations (or indicators of equations) might save valuable time for the tutors. Although queries not categorised as containing a mathematical equation may not be completely ignored by the tutors, such a process may serve as an indicator as to which of the queries in a running feed may need deeper scrutiny by the tutor. In order for such a process to be of value, it would need to perform the classification task at a similar level to that of a human.

Therefore, the main objective of this study is to determine whether an automated process could classify microtext statements at a level which simulates human judgement. For the purposes of this study, a relevant query is referred to as a candidate. To support this main objective, two secondary objectives have been defined. The first of these is to identify and implement various techniques for classifying queries automatically on the Dr Math service. The second of the two secondary objectives is to compare and contrast the results of the automated classifiers with those of human participants.

3. BACKGROUND

In order to address a problem best, it is necessary first to investigate its related domain. Therefore, this section consists of a discussion of the Dr Math service, as well as the data sets on which all the tests in this study are conducted.

3.1. Dr Math

Mxit, the mobile chat application on which the learners access the Dr Math platform, may be installed on most makes and models of cellular phone. This, along with the low amount of bandwidth necessary for its functioning, has endeared it amongst the various South African economic and racial groups. It is because of this broad and diverse user base that Dr Math has reached such widespread adoption.

Dr Math was originally a system used to research whether learners would use their personal cellular phone to get assistance with mathematics homework (Butgereit & Botha, 2010b). The software receives an input queue from an external source (such as learner cellular phones) and is able to distribute the queue amongst various human recipients, who act as tutors to the learners sending in their requests.

Dr Math focuses specifically on providing mathematics tutoring to South African high school learners. This solution has been so successful that it has grown to accommodate the requests of tens of thousands of school learners (Butgereit & Botha, 2010a).

3.2. Data Set Preparation

In order to train, test and validate the classification process, it was necessary to source representative data. The Dr Math service stores all of the learner-tutor interactions as anonymous text files. For the purposes of this study, the 2009 to 2013 Dr Math query logs were retrieved with permission of the South African Council for Scientific and Industrial Research (CSIR). Although the entire data set is not allowed to be redistributed, a few example query statements are shown in Snippet 1.

Since each automated process has to emulate the judgement of a human tutor in order to select suitable candidates, it was necessary to create tagged versions of the logs. For this purpose random extracts of the 2010 and 2011 logs were split into two groupings of 2500 entries (queries) each. One group was selected to act as a training data set and another as a test set. For both these data sets, each entry was manually scrutinised and tagged as being a suitable candidate or not. All the candidate identification techniques employed in this study make use of the training and test set for their initial training and subsequent testing.

ok wt dnt u knw in trig
at wht tym wil u be onlyn again cuz nw im buzy eatng?
ill be on till five
ok mr i'l get bck 2 u b4 fiv
ok cool
gud dy mr
...
hello - how may i help you?
4 wat
help with math homework
wat is the answer 4 15-12
use our calcluator type in .c 15 - 12
i want u 2 answer
sorry you have to do the dirty work ;-)
hv u make sex before like me;)

bye bye
no i want 2 hv sex with u
...
next pr0blem
(4p) x (-4p)
wait ign0re that
and x is multiply?
the sum is 40ab^2 divided by -5ab
are we skipping 4p x -4p?
yup
ok so 40ab^2 / -5ab what is common on top and bottom?
5ab
yes that is common so what is left on top and bottm?
-8b over 1?

Snippet 1: A few example queries

In order to contrast the results of the various techniques with independent human judgement, a validation data set was created. To create this data set, a further 1000 entries were selected from the 2013 Dr Math system logs. The validation data set was given to three human coders, in order to determine which entries they would select as containing content related to mathematics. The level of agreement between the three human coders was calculated at 0.75, using Krippendorff's α. Krippendorff suggests that values between 0.667 and 0.8 may be used for drawing tentative conclusions (Krippendorff, 2004: 241). As this relatively low level of agreement does not inspire a high level of confidence, each of the 1000 entries in the dataset was labelled as containing mathematical content only if all three coders were in agreement that it did. Using this method revealed that only 218 of the 1000 entries contained mathematical content.

The relatively low level of agreement between the human participants highlights the need for an automated classification approach. Such an approach would not be bound by differing personal opinions as would the response of a human, but would rather be guided by set rules and statistical measures.

4. DISCUSSION AND IMPLEMENTATION

Various techniques are applicable to the process of identifying statements automatically containing mathematical content (candidates). For this study, we have selected five techniques to apply to the problem domain. These techniques are a rule-based filter, an artificial neural network, a decision tree, a Bayesian filter and a k-means clustering algorithm. The following sub-sections briefly discuss the theory related to each technique before describing how the technique was implemented in this study.

4.1. Rule-based Filters

Haskins & Botha (2012) created rule-based filters for the purpose of categorising Dr Math microtext queries as candidates for translation into mathematical statements. The first of the three filter-types, devised for the aforementioned study, worked on the premise that an entry may be a valid translation candidate if it contains numbers. This assumption did not hold true, as some of the entries contained numbers related to personal information, such as birthdays or placement in contests. The rules were adjusted to compensate for this and to lower the number of false positives flagged by the filter.

The mathematical operators, which acted as flags for the second filter, were sourced from South African high school mathematics textbooks. These symbols were constrained to those which may be constructed using a basic (non-touch screen) cellular phone keypad. The log entries contain various built-in auto response commands and emoticons, which were removed by a pre-processing step, in order to improve the accuracy of the process. Rules were also added to the filter to ensure that queries containing the basic mathematics operators, such as plus, minus, multiply, divide and equal were given extra weight.

For the final filter, it was hypothesised that a mathematical query may also be put into words or its description may be supported by certain key words. Using a variety of techniques, such as stemming, adjacency, n-grams, Levenshtein distance and word decay, 233 tag words were identified (Haskins & Botha, 2013). In order to be flagged by the filter, every processed log entry had to contain at least two of these tag words. Extra weighting was given to certain combinations of words, such as divide and by.

For the purposes of the current study, the rule-based filters are used not to select candidates for translation, but to identify candidates which may contain content related to mathematics.

4.2. Artificial Neural network (ANN)

The human brain is very adept at identifying patterns in seemingly random data. ANNs are computational constructs which attempt to mimic the functioning of neurons in a human brain, in order to transform a set of inputs into (a) meaningful output(s). In these networks, neurons are represented as nodes, operating as nonlinear summing devices which co-operate to perform a desired function (Dayhoff & DeLeo, 2001). ANNs have been used in the past for text classification tasks such as email spam filtering (Clark, Koprinska & Poon, 2003), text categorization (Lam & Lee, 1999) and topic spotting (Lam & Lee, 1999).

A popular form of ANN is that of the multi-layered perceptron (MLP). Such networks consist of a series of input vectors, which all link in a fully meshed fashion to a series of hidden (processing) nodes. These hidden nodes, in turn, are fully meshed to one or more output nodes. A node in an MLP network may have multiple inputs, but a single output. Any incoming connection to a node has a weight w associated with it. The weight of every input is multiplied by the associated value a received on that input and then added to the results of performing the same calculation on all the other input value / weight pairs.

Table 1: Neural network factors

Description	Input or Output
Percentage of characters which are mathematical symbols.	Input
Percentage of characters which are letters.	Input
Percentage of characters which are numbers.	Input
Percentage of characters which are either numbers or mathematical symbols.	Input
Closest proximity between numeric and alphabetic characters, expressed as a percentage.	Input
Closest proximity between mathematical symbols and numeric characters, expressed as a percentage.	Input
The number of words in the entry, related to mathematics, expressed as a percentage.	Input
A low value indicating an entry which does not contain mathematics or a high value indicating an entry which contains mathematics	Output

The weights are derived from a training process, during which the network is presented with sample data. ANN training mainly takes the form of supervised or unsupervised training. Supervised training involves the application of training data in the form of artificially constructed or empirically observed input-output mappings, which are representative of the desired behaviour of the network (Jaeger 2005). The outputs to the data are recorded and the network then proceeds to work backwards from the output node to adjust the activation functions of the different nodes. This process is referred to as backwards propagation of the error or, simply, as back-propagation (BP). During unsupervised training the network is provided only with inputs. The network has to adjust automatically so that similar inputs provide similar outputs.

As the second categorisation vehicle, it was decided to create a BP MLP artificial neural network. In order to perform the necessary training of the network, use was made of supervised learning. An algorithm was devised to turn each data set entry into a set of input and output factors. The 7 input and single output factor chosen are shown in Table 1.

The neural network operates using input and output floating point values between 0 and 1. To train the network to output a candidate as a value close to 1 and a non-candidate as a value close to 0, all input and output factors were normalised. Any factor lower than the median is normalised to a value of 0.01. All factors with a value at the median or higher is normalised to a value of 0.99.

The network thus consists of 7 input nodes and 1 output node. In order to determine which number of hidden nodes would optimally fit this problem; tests were conducted with hidden nodes ranging in number from 1 to 14. In all cases the network was trained over a series of epochs. The training was performed on 200 entries from the training data set. During each epoch the network would undergo 10 iterations of adjustment, using the training data set, at a learning rate of 0.1. After the adjustments, the trained network was tested using 200 further entries from the training set. For each number of hidden nodes, the network was trained and tested for a total of 1000 epochs. The epoch during which each network variation reached its minimal local error rate and optimal accuracy was recorded during the training and validation.

All networks eventually reached the same level of precision on the validation set, identifying accurately whether an entry contains mathematics or not 88.5% of the time. The only differences were the number of epochs it took before the network became optimised and the lowest error rate of the network. To determine if there is any difference in the precision of the 14 different network variations, when applied to the test data set, each of the networks was retrained to their minimal local error rate. Table 2 lists the accuracy of the various configurations of the neural network.

Table 2: Configuration precision

Number of hidden nodes	Precision (%)
1 – 4, 6, 8, 8 – 10	93.48
5, 7, 12 – 13	93.28
14	93.16
11	80.64

In order to determine which of the configurations to use as the final version of the neural network, the processing speed of the networks with a precision of 93.48% was contrasted, as shown by the graph in Figure 1. From this it became clear that the fastest configuration uses 4 hidden nodes.

Although the actual speed in milliseconds of each of these processes is irrelevant as the tests were run on a test-bed system, they do serve to indicate the difference in time required by the various configurations to reach the same conclusion. As shown in Figure 2, the local minimal error rate for this network was set at $2.59015 \times = 10^{-8}$ after 92 epochs.

Thus, the final BP MLP neural network configuration for this study consists of 7 input, 4 hidden and 1 output nodes.

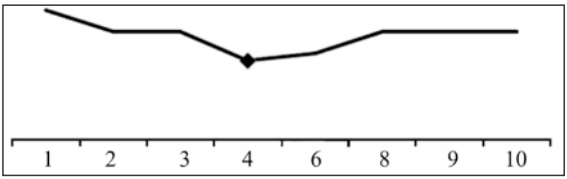

Figure 1: Neural network processing speed in relation to the number of hidden nodes

4.3. Decision Trees

Decision trees are representations of a series of logical decisions. They may be used to demonstrate and/or predict how a set of input variables may result in a pre-set conclusion. Decision trees have been used to perform facial recognition (Maturana, Mery & Soto, 2011), part-of-speech tagging (Schmid, 1994) and text categorization (Remeikis, Skucas & Melninkaite, 2005). A decision tree is constructed as a series of decision nodes. Each node corresponds to an input variable from a data set. A decision is made at each node in order to create a branching structure, representing the results of each decision. Each branch of the structure should end at a final, labelled decision. These end-nodes are referred to as leaf nodes.

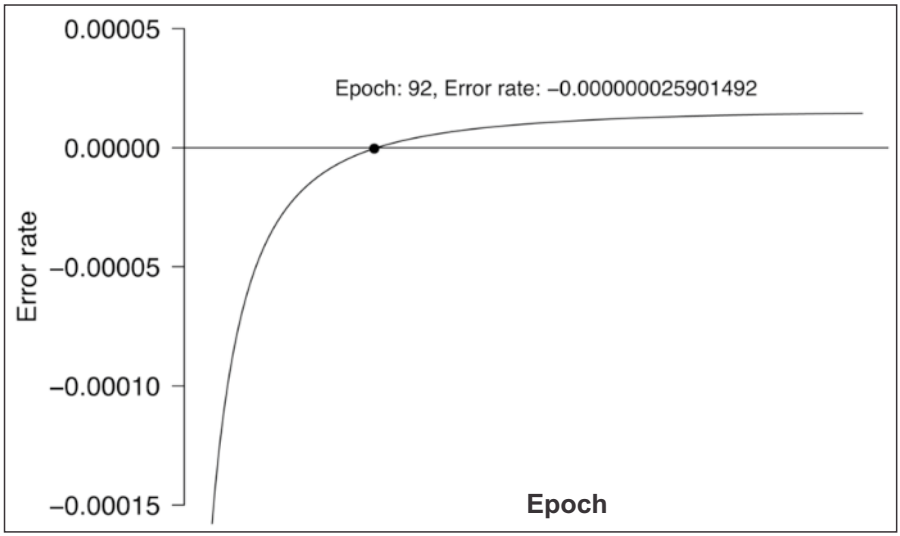

Figure 2: Changes in error rate for a network with 4 hidden nodes

There are various techniques for constructing a decision tree. The form used for this study is referred to as a binary decision tree (BDT). These trees require that a mutually exclusive decision be made at each junction, resulting in one of two outcomes, e.g. yes/no, true/false or greater/smaller than. Decision trees do not operate only on numeric values, but are applicable to categorical values as well.

The BDT uses the same normalised factors as the ANN, devised in the previous section. A tree structure learning algorithm was devised specifically to cater to the content of the queries used in this study, called the GPM (Grow, Prune and Modify) tree. The GPM tree was implemented as a binary decision tree, which initially makes decisions based on the median value of the overall data set. In the case of the input data, this median value was 0.5.

The growth phase of the decision tree was performed by processing iteratively each of the entries contained in the training logs. The nodes of the tree were created by following the branches from an initial root node. At every decision node, one of the 7 input, comparison values was compared to the median of 0.5. If the value was greater than or equal to or less than the median value, a new branch in the tree was created (if no branch to this end already exists) and the following branch node was set as the current node. Every set of comparison values was followed through to its conclusion as either a candidate statement or not.

The construction of the tree was conducted in an unbalanced manner, so that only the ultimate decision paths contained in the data set were mapped. No alternate branches were created to indicate decisions for which the result could not be proven from the data set. Once the tree has passed its growth phase, a recursive pruning algorithm removes or trims any unnecessary or lengthy branches. The pruning continues until no further changes can be made to the tree.

An iterative process was then embarked upon whereby every entry in the training set was fed to the decision tree. If the entry was incorrectly classified, the values of each of the nodes of the tree were modified iteratively by varying them individually from a low value of 0.01 to a high of 0.99, while keeping the values of all the other nodes constant. After each variation, the overall accuracy of the tree was again tested against the entire training log. If the change in the node value increased the overall accuracy, the change was retained; otherwise it was discarded.

These modifications continued until it was no longer possible to increase the overall accuracy of the tree on the training data set. This entire process is illustrated in Figure 3. After this phase was completed, the final tree had the same structure as at the end of the pruning phase, but with different node values. The tree was converted to a series of branching selection (if - else) statements, to avoid having to retrain the tree before each use.

4.4. Bayesian Classifier

The Bayesian classifier is based on the Bayesian theorem, which suggests how to combine statistical evidence. The classifiers are also referred to as belief networks as they use probabilities to determine an outcome or classification. The contents of individual Dr Math learner queries may contain too much variance to perform classification adequately according to probability. A pre-processor removes the variance of numbers, mathematical symbols and variables found in the text, by converting all instances of each of the aforementioned to a single occurrence of the word number, symbol or variable. A prior study identified specific mathematical tag words, which are repeatedly encountered in the Dr Math query logs (Haskins & Botha 2013). Occurrences of any of these tag words are replaced with the word tag.

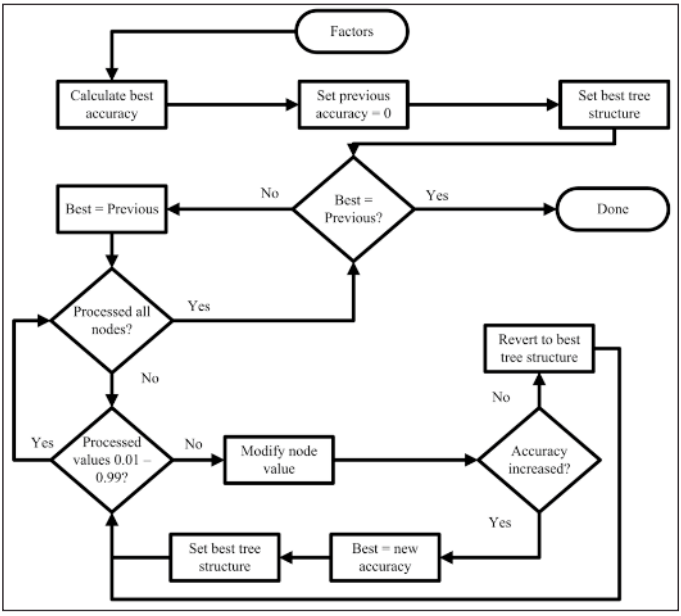

Figure 3: The decision tree modification phase process

Graham devised a means for identifying spam in emails, by providing two separate input lists (Graham, 2004). One represents possible spam emails and the other represents valid emails. The two sets of emails were then processed separately to create two hash tables mapping individual words (tokens) in the sets to the number of times they occur in the corpus of the given input set. A third hash table was then created from the two tables, mapping every entry to a probability that an email containing it is spam. This technique was applied to identify candidate queries.

The training set was split into two separate lists, one containing only valid candidate entries and the other non-candidate entries. These non-candidate entries are akin to the spam entries in the email filter. Equation 1, for mapping every entry to a probability that an email containing it is spam, was applied to the entries in the lists (Graham, 2004).

$$\begin{aligned} r_g &= \min\left(1, 2\left(\frac{good(w)}{G}\right)\right) \\ r_b &= \min\left(1, \frac{bad(w)}{B}\right) \\ P_{non|w} &= \max\left(.01, \min\left(.99, \frac{r_b}{r_g + r_b}\right)\right) \end{aligned}$$

Equation 1 was used as follows in this study. Pnon|w is the probability that a query is not a valid candidate entry. w is the token (word) of which the probability is being calculated. good and bad refer to the previously calculated occurrence rate saved in the hash tables and G and B are the total word counts in each table respectively. This calculation was done for every word in the training corpus. This list of word and probability associations was saved to serve as input data during the training and testing phases of the study.

For the purposes of this study, it was decided to use the overall probability calculation as provided by Graham (2004) in Equation 2. The equation used by Graham (2004), proposed the use of (word) probabilities of 15 interesting tokens to calculate the overall probability. The study used a naïve Bayes implementation, using a default probability of 0.5 (50%) that an email is spam or not. A naïve Bayes implementation does not make any predetermined assumptions about the data being processed, thus a mid-range value such as 50% is used as an initial probability. Interesting tokens are those with their probabilities set the furthest from 0.5. Words that have not been encountered before are assigned a default probability of 0.4.

$$P_{non} = \frac{\prod_{i=1}^{15} P_{non|w_i}}{\prod_{i=1}^{15} P_{non|w_i} + \prod_{i=1}^{15} (1 - P_{non|w_i})}$$

For our study, this method was used as is. The average number of tokens in a given Dr Math learner query was calculated as 10. It was attempted to adjust the calculation to use the probabilities of the 10 most interesting tokens, but this did not provide a better result than using the original 15 tokens proposed by (Graham 2004).

This calculation returns a combined probability in the range of 0 to 1, with values approaching 0 having a greater probability of being valid candidates and values approaching 1 having a higher probability of not being valid candidates. Graham considered an email as being spam if the combined probability is higher than 0.9 (Graham 2004).

This works well for the purpose of spam filtering, as spam is supposed to be the exception rather than the norm. The Dr Math learner queries seem to suggest the opposite, with the actual entries containing mathematics being the exception.

In order to determine the optimal threshold for accurate classification, the training data set was processed repeatedly using a varying threshold while comparing the classification results with the pre-classified labels of the training set. The lowest deviation of 0.47% was found at a probability threshold of 0.0012. This threshold serves as the cut-off value between valid and invalid candidates.

4.5. k-means Clustering

Another method that has been applied to the categorisation process is that of k-means clustering. Clustering algorithms place series of values into groups (clusters), so that those with similar values end up in the same groups. For creating and testing the k-means algorithm, the normalised training sets created for the Bayesian filter were further processed. They were converted to series of input vectors, by substituting each entry with a list consisting of the probabilities of each token (word), contained in the entry. An output vector was added for confirmation purposes. This vector is set to 1 for a candidate entry and to 0 for a non-candidate entry.

Initially, a version of a standard k-means clustering algorithm was implemented. The results were discouraging as it was, on average, only able to categorise 39.93% of the entries accurately. The reason why the accuracy rating is an average, is that because of the nature of the clustering algorithm there is no guarantee that the same clusters would be created on consecutive runs. This is problematic if the process were to be employed in a real-world candidate identification scenario.

An alternate version of k-means, with manual intervention, was then attempted. The training data sets were grouped into two separate sets in preparation for this study, providing two neat clusters of tuples. One cluster represents entries containing mathematics and the other represents entries which do not contain mathematics. For each of the entries in these clusters, their mean squared value was calculated. From this an average value was calculated to represent the mean square value of each cluster.

The two calculated mean values of 1.1245 and 1.2246 achieved a relatively high rate of accuracy on the training data set. This approach was able to categorise correctly 82.20% of the training data. To determine if this result could be improved upon, an algorithm was implemented which adjusted iteratively the mean value of the valid cluster by 0.0001, from 0 to 3. At every iteration the precision of the categorisation process was recorded.

This was a very lengthy process, but eventually the values of 0.9180 and 1.2246 were selected to represent the clusters of valid and invalid translation candidates. These values resulted in a precision of 84.88% on the training data. The two means were used as pre-calculated centroid values to perform Euclidean distance calculations for each further input entry.

5. AN INITIAL COMPARISON

After training the different techniques for selecting candidates, their results were contrasted using the test data set. The test data set contains 327 entries containing mathematics and 2173 entries which do not contain mathematics. The metrics chosen for contrasting the various classifiers are those of precision and recall. Recall may be described as the number of positive categorisations made by the automated process divided by the total number of positive categorisations in the reference data set, i.e. the rate of true positives. Precision may be described as a reference to how many of the positive classifications made by the automated process are indeed true positives. For each of the techniques, the labels of the dataset serve as a reference and the results from the automated process serve as the candidate.

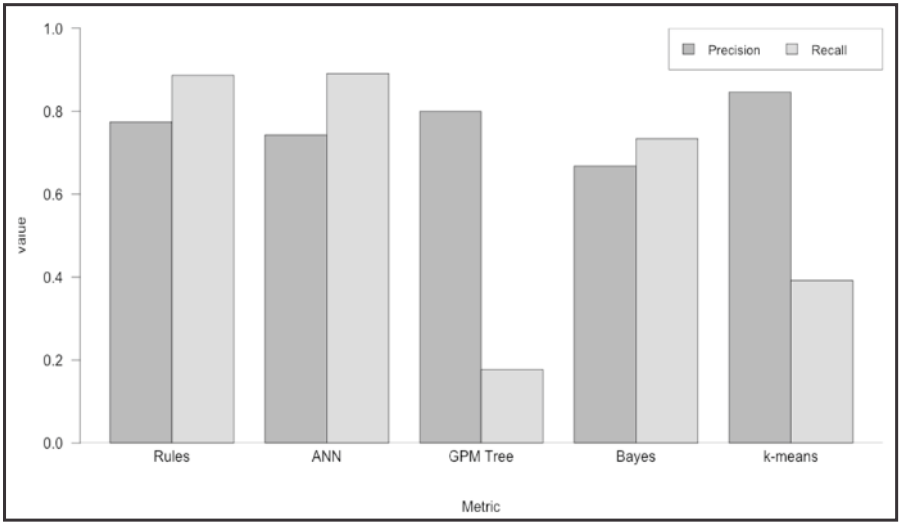

Figure 4: A comparison of recall and precision on the test data set

As Figure 4 indicates, the rule-based filter and the neural network provide very similar results, with regard to precision and recall. Both techniques provide a very high level of recall, i.e. they provide a high rate of true positives. Furthermore, the precision of both techniques approaches 0.8, which means that the majority of positive classifications made by the techniques are, in fact, true positives.

Although both the GPM tree and k-means classifier provide a higher precision than both the rule-based filter and neural network; their low levels of recall mean that they may miss too many positive classifications. The log entries in the test set were taken from the same time frame as those of the training set, namely 2010 to 2011; therefore it may be expected that their content and language may be similarly structured. To ensure that the classification techniques provide useful results, the validation of the techniques was achieved using the validation data set, which contains log set entries from 2013. This is discussed in the following section.

6. VALIDATING THE TECHNIQUES

The training and test data sets were classified by one of the authors of this paper. The validation data set was classified by three coders, all of whom have no further stake in this study. Figure 5 illustrates the results of the precision and recall measurements for the automated techniques on the validation data set. The results show a similar pattern to the results achieved on the test set, with the rule-based filter and the neural network providing the closest matching, high precision and recall measures. The GPM tree and k-means classifier again show very high levels of recall, but low levels of precision. Specifically, the GPM tree has a precision of nearly 1.0, i.e. nearly 100%. The Bayesian classifier consistently performs at a level slightly lower that the rule-based filter and neural network. This technique may hold future promise if re-trained with larger training data sets.

In order to perform a final comparison of the various techniques, it may be simpler to com- bine the precision and recall metrics into a single metric, called F-measure. Makhoul, Kubala, Schwartz, Weischedel et al. (1999) describe F-measure as representing the harmonic mean between precision and recall. A high F-measure value ensures a high level of both precision and recall, as the harmonic mean of two numbers is normally closer to the lower of the two input values. As the purpose of the automated techniques is to approximate the classification judgement of a human tutor, it was decided to determine how closely the classifications made by the automated techniques correlate with those of the human coders. Correlation may be defined as a mutual relationship of interdependence between two or more things (Soanes, Stevenson & Hawker 2004). For this study, it was decided to employ the widely used Pearson correlation coefficient.

To serve as basis for comparison, the level of correlation was calculated among the classification results of the individual human coders. For this purpose, the correlation was calculated for the three coder pairings Coder1-Coder2, Coder2-Coder3 and Coder1-Coder3 and then averaged. This average correlation coefficient was calculated as 0.774.

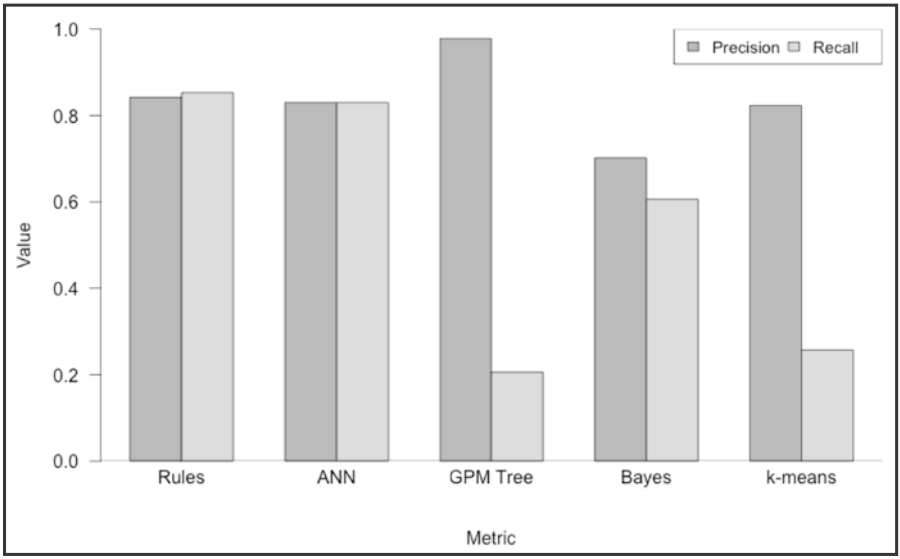

Figure 5: A comparison of precision and recall on the validation data set

Figure 6 illustrates graphically both the F-measure calculated for each technique, as well as the correlation coefficient calculated between the classifications of the given automated technique and the human coders. From this figure several conclusions may be drawn. The first is that for all of these techniques, their F-measure and correlation values are very similar. In future studies, it might therefore suffice to use only one of these values as a metric. The second conclusion is that the rule-based filter (0.804) and the neural network (0.783), with regard to the results of the human coders, show a strong correlation.

Strong (or high) correlations are defined as being in the range of 0.68 to 0.89 (Taylor 1990). This means that the classifications made by these techniques align relatively closely to those made by the human coders. Furthermore, this value is comparable to the level of correlation found when comparing the classifications of the individual human participants to one another. The final conclusion is that the rule-based technique and the neural network are very similar in performance. Although the rule-based process provides slightly better results, it is less future-proof and flexible than the neural network, as it is based on hard-coded conditional logic statements. As microtext is a fluid language, these logic statements may only be applicable for a limited time period. The neural network may more easily be kept relevant by re-training it periodically.

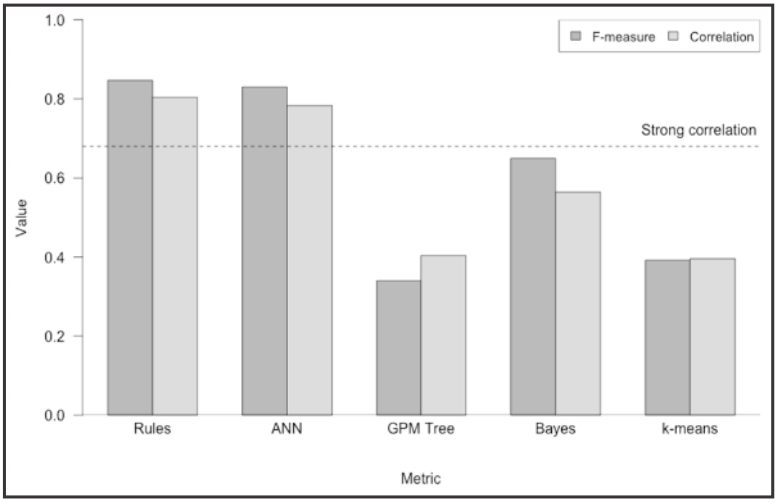

Figure 6: A comparison of the F-measure calculation for each technique and the Pearson's correlation coefficient between each technique and the classifications made by the human coders

7. CONCLUSION AND FUTURE WORK

This study set out to address two secondary objectives. The first of these secondary objectives, which was to identify and implement various techniques for classifying queries automatically on the Dr Math service, was addressed by identifying different techniques for evaluating whether an individual entry contains mathematics. The second of the secondary objectives, which was to contrast the results of the automated classifier with those of human participants, was addressed by contrasting the results of the techniques with those of three human coders. From these tests, two techniques were identified as delivering results similar to that of the human coders. These techniques were a rule-based filter and a back-propagation neural network.

The completion of the secondary objectives supports the main objective, which was to determine whether an automated process could classify microtext statements at a level which simulates human judgement. As the process is automated, it is able to perform the classification task at a much higher rate than is possible by any human tutor. Such a process may, therefore, be used as a means to focus the attention of the tutors on entries more relevant to the tutoring service. If there is a backlog on the service, such a process may be used as a threshold-based filter to warn the learners and tutors automatically to keep to the intended topic.

In future, the classification process will be refined and tested on the intended server hardware as part of a larger web service. Some of the identification techniques, such as the Bayesian filter, may be re-addressed by the application of more training data. Others, such as the rule-based technique, may have to be revised in future owing to the transitive nature of the microtext language on which it is based. As the authors of this study are not directly involved with the administration of the Dr Math service, the process devised in this study has not yet been implemented in the live Dr Math service. A future study may report on the utility of an automated classifier when applied to the live service.

8. REFERENCES

Butgereit, L. & Botha, R.A. 2010a. C3TO: An Architecture for Implementing a Chat Based Call Centre and Tutoring Online. In Proceedings of the IST-Africa 2010 Conference. Durban, South Africa: IST Africa pp. 1–6.

Butgereit, L & Botha, R.A. 2010b. Scaling a Mobile Tutoring Project: Strategic Interventions in C3TO. In Proceedings of the 12th Annual (South African) Conference on World Wide Web Applications. Durban, South Africa.

Clark, J., Koprinska, I. & Poon, J. 2003. A Neural Network Based Approach to Automated E-mail Classification. In Proceedings of the 2003 IEEE/WIC International Conference on Web Intelligence. pp. 702–705.
Dayhoff, J.E. & DeLeo. J.M. 2001. Artificial Neural Networks. Cancer 91(8):1615–1635.

Graham, P. 2004. Chapter 8: A Plan for Spam. O'Reilly Media Inc. pp. 121–129.

Haskins, BP & Botha, R.A. 2012. Identifying Suitable Mathematical Translation Candidates From the Logs of Dr Math. In Proceedings of the 23rd Annual Symposium of the Pattern Recognition Association of South Africa, ed. A de Waal. Pretoria, South Africa: PRASA.

Haskins, BP & Botha, R.A. 2013. Identifying Tag Word Counterparts for Dr Math. In 2013 AAAI Spring Symposium on Analyzing Microtext, ed. Eduard Hovy, Vita Markman, Craig Martell & David Uthus. Stanford, California: AAAI.

Hovy, E, Markman, V., Martell, C., & Uthus, D. 2013. Preface. In Proceedings of the 2013 AAAI Spring Symposium on Analyzing Microtext. AAAI p. vii.

Jaeger, H.. 2005. A Tutorial on Training Recurrent Neural Networks, Covering BPPT, RTRL, EKF and the 'echo State Network' Approach. ReVision 2002:1–46.

Krippendorff, K. 2004. Content Analysis: An Introduction to Its Methodology. 2nd edn. SAGE Publications, Inc.

Lam, S.L.Y. & Lee, D.L. 1999. Feature Reduction for Neural Network Based Text Cat- egorization. In Proceedings of the 6th International Conference on Database Systems for Advanced Applications. DASFAA '99 Washington, DC, USA: IEEE Computer Society pp. 195–202.

Makhoul, J., Kubala, F., Schwartz, R., Weischedel, R. et al. 1999. Performance measures for information extraction. In Proceedings of the DARPA Broadcast News Work-shop. San Francisco: Morgan Kaufmann Publishers pp. 249–252.

Maturana, D., Mery, D. & Soto, A. 2011. Face Recognition With Decision Tree-Based Local Binary Patterns. In Computer Vision–ACCV 2010, ed. Ron Kimmel, Reinhard Klette & Akihiro Sugimoto. Vol. 6495 of Lecture Notes in Computer Science Springer Berlin Heidelberg pp. 618–629.

Remeikis, N., Skucas, I. & Melninkaite, V. 2005. A Combined Neural Network and Decision Tree Approach for Text Categorization. In Information Systems Development, ed. Olegas Vasilecas, Wita Wojtkowski, Jouˆe Zupancic, Albertas Caplinskas, W. Gregory Wojtkowski & Stanislaw Wrycza. Springer US pp. 173–184.

Schmid, H. 1994. Probabilistic Part-of-Speech Tagging Using Decision Trees. Vol. 12 New Methods pp. 44–49.

Soanes, C., Stevenson, A. & Hawker, S. eds. 2004. Concise Oxford English Dictionary. 11th edn. Oxford University Press.

Taylor, R. 1990. Interpretation of the Correlation Coefficient: a Basic Review. Journal of Diagnostic Medical Sonography 6(1):35–39.