

Deep Learning Application: Understanding Neural Network Behaviour in Text Classification from Tagged Examples through Machine Learning for Kids

Jomari Manto
College of Computer Studies & Engineering
José Rizal University
Mandaluyong City, Philippines
jomari.manto@my.jru.edu

Abstract—This report documents an exercise in which a neural network was trained to classify customer feedback into three intent categories using only tagged examples rather than hand-coded rules. The classifier was built on the Machine Learning for Kids platform and trained on accounting software reviews drawn from a publicly available corpus. Three rounds of training were evaluated against a fixed set of test inputs covering clear cases, sarcastic phrasing, and administrative enquiries falling outside the defined categories. The first model, trained on five examples per class, classified 62.5 per cent of scored inputs correctly. Expanding the set to approximately eighty per class improved accuracy on unambiguous inputs but eliminated it entirely on sarcastic ones, reducing overall accuracy to 37.5 per cent. A third round, adding a small number of targeted examples to a single class, restored sarcasm recognition in full while degrading performance elsewhere. Accuracy therefore did not rise with data volume alone. Feature extraction is performed inside the platform and is not exposed to the user, so its position in the pipeline was inferred from these results, which indicate that the numerical representation is rebuilt from the tagged examples at each training run. Class balance, vocabulary distribution and representativeness each affected outcomes independently of example count.

Keywords—text classification, neural networks, tagged examples, training data quality, feature extraction, model confidence

I. INTRODUCTION

The automatic sorting of documents into categories is one of the older problems in computing. In its early days the task was approached with statistical counting methods and hand-built rules, in which a machine decided a document's category from the occurrence of selected clue words chosen in advance [1]. Text classification today is instead carried out by neural networks that learn their own numerical representation of the text from labelled examples, and such models have overtaken the earlier approaches across tasks including sentiment analysis and question answering [2].

One domain where this matters commercially is software as a service. Users of these products leave feedback in large volumes through app stores, review sites and support channels, and that feedback carries information the vendor needs. Separating it by intent allows each type to be routed to the people who can act on it [3].

Accounting software is a useful case within that domain because the consequences of misrouting are concrete. A user unable to run payroll on a filing deadline has a different problem from one who dislikes the interface, even though both

may write at length and in similar language. This exercise therefore takes Intuit QuickBooks as its subject and builds a classifier that separates customer feedback into praise, complaints and urgent support requests.

The classifier was trained on the Machine Learning for Kids platform using tagged examples alone, with no rules written by hand. Three models were trained on progressively different data and tested against the same inputs, so that the effect of the training data could be observed in isolation. This report describes that method and its results, then addresses the limitations of rule-based classification, the influence of dataset quality and volume, and the point at which feature extraction occurs.

II. METHODOLOGY

The dataset was drawn from a publicly released collection of Amazon product reviews covering the software category, made available for academic use by the McAuley Lab at the University of California San Diego [4]. Reviews mentioning Intuit QuickBooks, Quicken or Intuit by name were selected from the collection, and short entries were excluded so that the material retained the length and irregularity of genuine customer writing. The review text was used as written, with only line breaks and repeated spaces normalised so that each review occupied a single line on import.

Three classes were defined for the classifier: Positive_Praise, Negative_Complaint and Urgent_Support. Because the source reviews carry star ratings rather than intent labels, each selected review was assigned to a class using the rating together with the presence of language indicating a service failure in progress. Every assignment was reviewed before the example was used.

The classifier was built on Machine Learning for Kids, a browser-based teaching platform that trains a text model from tagged examples without requiring any code. Examples were imported into the three classes and the model was trained through the platform interface. Three models were produced in sequence, each trained on a larger or differently composed set of examples than the last, so that the effect of changing the training data could be observed while everything else was held constant.

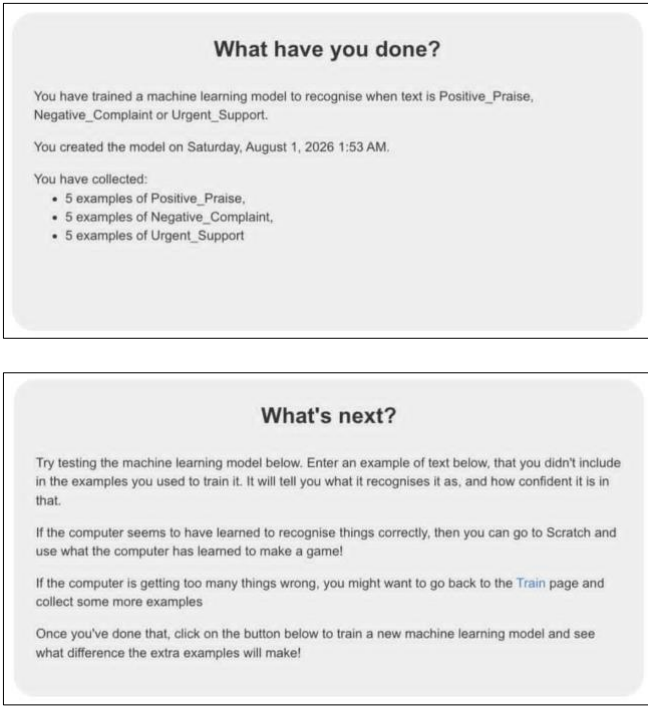


Fig. 1. Model information reported by the platform after training, showing the creation time and the number of tagged examples held in each class.

Evaluation used a fixed set of twelve short messages written by the author rather than drawn from the corpus, so that no test item had appeared in training. The set was divided into three groups covering unambiguous cases, sarcastic phrasing, and administrative enquiries falling outside all three classes. The same twelve inputs were submitted to each of the three models, and the predicted class and reported confidence were recorded for every result.

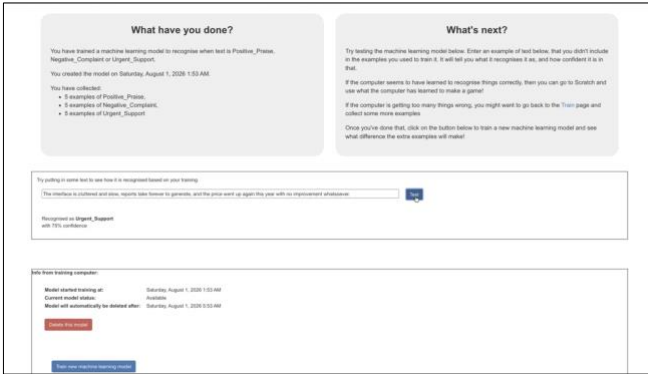


Fig. 2. The platform testing interface, showing a submitted test input together with the predicted class and the reported confidence.

III. RESULTS

The first model was trained on five examples per class. This was not the intended starting point; the exercise specifies a comparison between three examples per class and ten, but the platform refuses to train a model on fewer than five examples in any class. The floor is itself a result, in that the system declines to operate below a minimum quantity of evidence.

Against the twelve test inputs, this model classified two of the four unambiguous cases correctly and three of the four sarcastic ones, giving an overall accuracy of 62.5 per cent

across the eight scored items. The two failures in the first group were a direct exchange: a complaint about a slow interface was labelled Urgent_Support at 75 per cent confidence, while an account lockout during a payroll deadline was labelled Negative_Complaint at 61 per cent. On the four administrative enquiries, which belong to none of the three classes, the mean reported confidence was 1.25 per cent.

Expanding the training set to roughly eighty examples per class produced a mixed outcome. Accuracy on the unambiguous group rose to 75 per cent, and the payroll lockout was now labelled correctly. Accuracy on the sarcastic group fell to zero. All four sarcastic complaints were assigned to a positive or urgent class, and one of them, previously wrong at 26 per cent confidence, was now wrong at 96 per cent. Overall accuracy therefore fell to 37.5 per cent.

The third model added eighteen further examples to Negative_Complaint alone, each written to place positive vocabulary inside a complaint. Accuracy on the sarcastic group rose to 100 per cent, and the slow-interface complaint that had been misclassified in both earlier rounds was now correct. Accuracy on the unambiguous group fell to 25 per cent, and eleven of the twelve predictions returned Negative_Complaint. Overall accuracy returned to 62.5 per cent, the same figure as the first round but produced by a different distribution of errors.

TABLE I. CLASSIFICATION ACCURACY ACROSS THE THREE TRAINING ROUNDS

Measure	Round 1	Round 2	Round 3
Examples per class	5 / 5 / 5	79 / 80 / 80	79 / 98 / 80
Group A (clear)	50.0%	75.0%	25.0%
Group B (sarcastic)	75.0%	0.0%	100.0%
Overall (A + B)	62.5%	37.5%	62.5%
Group C mean confidence	1.25%	2.25%	3.00%

^a. Accuracy is defined only for Groups A and B, where a correct class exists. Group C inputs belong to none of the three classes, so mean confidence is reported instead.

Two observations apply across all three rounds. Reported confidence on the out-of-vocabulary group never exceeded six per cent, rising only from a mean of 1.25 to 3.00 per cent as the training set grew. Separately, the entire first round was repeated in a second project created on a different day, and every one of the twelve results was reproduced exactly, including the confidence percentages. The training procedure is therefore deterministic, and the differences recorded between rounds are attributable to the training data rather than to variation between runs.

IV. EXPLICIT RULES AND TAGGED LEARNING

Writing explicit conditional statements for every possible customer complaint would encounter several difficulties. The most obvious is coverage. Customers describe the same problem in different words, and a rule matching one phrasing will not fire on another, so the rule set must grow with every variant encountered. Maintaining that set is costly, and every addition risks contradicting a rule already present.

A second difficulty is that words do not carry fixed meanings. A rule triggering on refund would classify a request for a refund and a compliment about how quickly a refund was processed identically, because the rule can see the word but not the sentence around it.

The testing carried out here, however, shows that the comparison runs in both directions. The phrase asking where support could be obtained contains the word support, which is the name of the target class. A keyword rule would have classified it immediately and correctly. The trained model placed it in Negative_Complaint at six per cent confidence. The model had no notion that the word support relates to the class Urgent_Support, because it does not learn what words mean; it learns which documents were placed in which class. Tagged learning avoids the maintenance burden of explicit rules, but it acquires no knowledge that was not present in the examples supplied.

V. DATA QUALITY, VOLUME AND DIVERSITY

The results do not support the expectation that accuracy rises with the quantity of training data. Increasing the set from five examples per class to approximately eighty lowered overall accuracy from 62.5 per cent to 37.5 per cent.

The mechanism is visible in which items failed. The enlarged Positive_Praise class contained seventy-nine long reviews written by satisfied customers, and those reviews are dense with words such as love, great, fantastic and wonderful. Training on them strengthened the association between those words and the positive class. Sarcastic complaints open with exactly those words, and at five examples per class the association was weak enough that the surrounding complaint vocabulary still prevailed. At eighty it was not. Additional data of the same kind did not teach the model more about the problem; it reinforced a pattern that was already misleading.

The third round demonstrates the converse. Eighteen examples, selected specifically to place positive vocabulary inside complaints, recovered the sarcastic group completely and also corrected an error that had persisted through both earlier rounds. Eighteen well-chosen examples achieved what two hundred and twenty-five generic ones had not.

That round also introduced a distortion which should be recorded. The eighteen added examples averaged ninety-seven characters, whereas every other training example ranged between seven hundred and seventy-six and just under fifteen thousand. Sarcastic framing and short length were therefore introduced together, and since the test inputs are also short, part of the resulting shift toward Negative_Complaint is attributable to length rather than to sentiment. The refinement changed two variables where it should have changed one.

Class balance is a further factor. Negative_Complaint grew to ninety-eight examples against seventy-nine and eighty in the other two classes, and eleven of the twelve predictions in that round returned Negative_Complaint. Volume, balance, vocabulary distribution and the format of the examples each affected the outcome, and they did so independently of one another.

VI. FEATURE EXTRACTION IN THE PIPELINE

Feature extraction occurs during training, at the point where the platform builds the model from the tagged examples. It is not an action the user performs. The exercise material describes the step without naming it, in stating that words are converted into numerical tokens and mathematical associations are found between word patterns and labels.

The platform does not expose its architecture, so the exact scheme cannot be observed directly and the account below is

inferred from the behaviour recorded during testing. The text of each example is separated into words, a vocabulary is assembled from every word appearing across the tagged examples, and each document is represented as numerical values derived from which of those vocabulary items it contains. The classifier then operates on those values alone and never sees the text.

Three observations support this account. First, the four administrative enquiries returned confidence between zero and six per cent in every round. Words such as discount and weekends appear rarely or not at all in accounting software reviews, so few features could be derived from those inputs and the classifier had little to score. The vocabulary available to the model is therefore bounded by the examples supplied in the collection stage.

Second, the same input produced different classifications after the training data changed, while the input itself was unaltered. One sarcastic complaint moved from Positive_Praise at 26 per cent, to Urgent_Support at 96 per cent, to Negative_Complaint at 56 per cent across the three rounds. The numerical representation is therefore derived from the training examples and is rebuilt at each training run rather than carried forward.

Third, the repetition of the first round in a separate project returned identical results, which indicates that the representation is a deterministic function of the training data. Taken together, these observations place feature extraction inside the training step, constrain it to the vocabulary present in the tagged examples, and show it to be reconstructed whenever a new model is trained.

VII. PROGRAMMATIC ACCESS TO THE TRAINED MODEL

The platform exposes each trained model through a web interface for classifying individual inputs. It also issues a project key which allows the same model to be queried programmatically, and this was used to build a small interface in Google Colab. A single function submits text to the model and returns the predicted class with its confidence, and a selection control allows any of the twelve test inputs to be chosen and submitted without retyping.

The value of this for the exercise is that it separates the model from the interface used to reach it. The same three-class model that was evaluated through the browser returns identical results when queried from code, which confirms that the behaviour recorded in the preceding sections belongs to the trained model rather than to the platform interface.

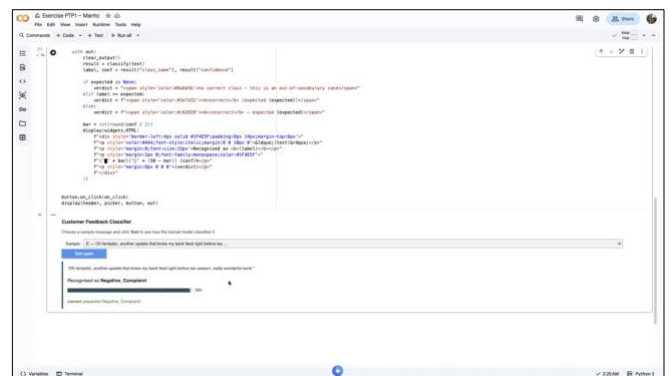


Fig. 3. Interactive classifier built in Google Colab, showing a selected test input, the class returned by the trained model and the reported confidence.

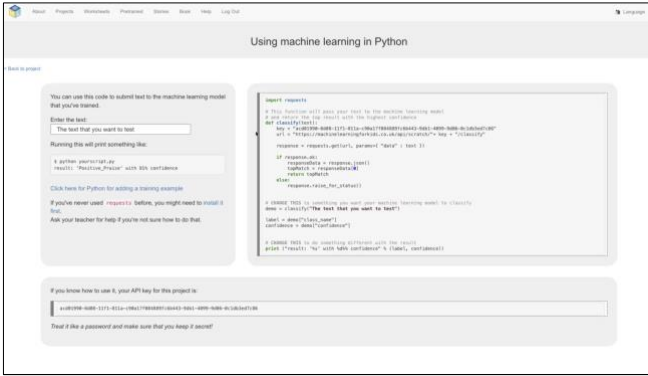


Fig. 4. Python code used to query the trained model through the platform project key.

VIII. CONCLUSION

A three-class text classifier was built from tagged examples alone, without hand-written rules, and its behaviour was observed across three training rounds. Overall accuracy moved from 62.5 per cent, to 37.5 per cent, and back to 62.5 per cent, with the errors falling in different places each time. Accuracy did not increase with the volume of training data.

The exercise showed that the composition of a training set matters more than its size. Adding seventy-five further examples per class degraded performance on the harder inputs, while eighteen examples chosen to address a specific failure repaired it. It also showed that reported confidence should not be read as a measure of correctness: the model was wrong at 96 per cent confidence on one input and correctly uncertain at one per cent on another.

Both findings follow from where feature extraction sits. Because the numerical representation is built solely from the tagged examples at training time, the model can only distinguish what the examples distinguish. Improving a classifier of this kind is therefore a question of what is placed in the training set, not how much.

ACKNOWLEDGMENT

The author thanks Dr. Rodolfo C. Raga Jr. for the design of this exercise. The review data used for training was made available for academic use by the McAuley Lab, University of California San Diego.

REFERENCES

- [1] M. E. Maron, "Automatic indexing: An experimental inquiry," *J. ACM*, vol. 8, no. 3, pp. 404–417, Jul. 1961.
- [2] S. Minaee, N. Kalchbrenner, E. Cambria, N. Nikzad, M. Chenaghlu, and J. Gao, "Deep learning-based text classification: A comprehensive review," *ACM Comput. Surv.*, vol. 54, no. 3, Art. 62, Apr. 2021.
- [3] W. Maalej and H. Nabil, "Bug report, feature request, or simply praise? On automatically classifying app reviews," in *Proc. IEEE 23rd Int. Requirements Eng. Conf. (RE)*, 2015, pp. 116–125.
- [4] Y. Hou, J. Li, Z. He, A. Yan, X. Chen, and J. McAuley, "Bridging language and items for retrieval and recommendation," *arXiv:2403.03952*, 2024.