Enhancing Cortana User Experience Using Machine Learning

Emad Elwany
elwany@stanford.edu
Microsoft Research

Siamak Shakeri
siamaks@stanford.edu
Microsoft

December 2014

Abstract

Voice enabled personal assistants like Microsoft Cortana are becoming better every day. As a result more users are relying on such software to accomplish more tasks. While these applications are significantly improving due to great advancements in the underlying technologies, there are still shortcomings in their performance resulting in a class of user queries that such assistants cannot yet handle with satisfactory results.

We analyze the data from millions of user queries, and build a machine learning system capable of classifying user queries into two classes; a class of queries that are addressable by Cortana with high user satisfaction, and a class of queries that are not. We then use unsupervised learning to cluster similar queries and assign them to human assistants who can complement Cortana functionality.

1 Introduction

Cortana is getting better at answering user queries for more and more scenarios every day, from adding reminders, to obtaining directions, tracking flights, and many more. However, the current technology does not allow addressing more advanced scenarios such as scheduling a doctor appointment or arranging dinner at a restaurant with a friend. These more complex tasks could however be broken into simpler tasks, some of which can be automated, while others require human intervention. Human intervention however is costly, and does not scale as well as automated task execution, and thus it should only be utilized as scarcely as possible.

In order to achieve that, we need to have a system that can analyze user queries and make a decision on whether a query should go through the normal Cortana flow, or the human enhanced one. In addition, the system also needs to identify what category of human assistant is best suited for this type of query, so that the query can be directed and handled efficiently.

2 Datasets

To train our models, we extracted hundreds of thousands of anonymized Cortana query log entries through a Microsoft proprietary internal data warehouse. Each log entry corresponds to a user query, along with relevant information like time, location, duration, as well as telemetry information indicating how the user interacted with the Cortana responses and the overall status of the query. In addition, we are able to group queries by user and by session using anonymous user and session ID's. This allows us to view the progression of a certain user's queries through a single search session.

While the query logs contain several types of data, we focused our attention on three:

- ***General Search***: Queries which Cortana redirects to Bing search.
- ***Command and Control (C&C)***: Queries where the user is trying to execute a task using voice command, such as adding an appointment to the calendar, or sending a text message.
- ***Enriched Search***: Queries where Cortana is able to provide enriched responses, such as weather conditions, current traffic, telling a joke and etc.

C&C and Enriched queries align very closely with the types of queries the Cortana-enhanced platform should target to provide an even better user experience. Due to computing constraints, we trained the model using a 1 day window of data(table 1)

Table 1: Summary of Datasets

Data Category	Class 1 to 0 ratio	Train size	Test size
General Data	0.304	302k	33k
C&C	0.596	223k	25k
Enriched	6.14	28k	3.1k

3 Data Labeling

We employed two different heuristics to label the data in a pre-processing step using the information

in the logs. This allowed us to avoid manual labeling and to be able to work with large amounts of data. Our heuristics separate the data into two classes:
0: Sufficiently Satisfactory Cortana response
1: Unsatisfactory Cortana response

3.1 Repetition Based Tagging: RBT

The simple heuristic works on the weak assumption that an unsatisfied user re-submits the query one or more times with no or minor tweaks within the same session. The labeling step goes through all the log entries and labels an entry '1' if a user submits a slightly tweaked version of the query multiple times. We detect such queries using Levenshtein distance of 5 [3].

3.2 Feedback Based Tagging: FBT

This heuristic takes into account more explicit signals from the logs. This information is only available for Enriched and C&C queries. After Cortana provides the user with answers or actions to these queries, it asks the user if s/he has been satisfied. We use this signal as the tagging in this heuristic. However an issue that might cause the labeling, especially with Enriched data to be less reliable, is that the users might just hit a button to move to another screen without providing feedback to Cortana. This will cause Cortana to tag that query as unsuccessful. By looking at the data, we noticed that for Enriched data, the provided feedback on the status of the query does not truly reflect the Cortana response satisfaction. We will discuss this later in models performance analysis.

4 Feature Extraction

We use the raw query as our primary feature, and represent queries using the event model. we construct a dictionary of words by pre-processing all the queries, eliminating noise, non-alphabetic and non-English words, as well as applying basic stemming to the dictionary terms. We then encode every query into a bit vector of size k, where k is the number of terms in our vocabulary. The i^{th} bit of each vector represents the number of occurrences of the i^{th} term in our vocabulary. The number of features after all the query modifications turned up to be 10,414.

5 Classification Models

5.1 Naive Bayes

We start by using a Bayesian classifier with the multinomial event model for text classification and Laplace smoothing. We test the classifier performance by using simple cross validation. We use General Search data(table 1) for training and testing. The results are summarized in the following table:[1] As it can be seen, the error rates are high

Table 2: Naive Bayes with Repetition Based Tagging

Test error	Training error	Precision	Recall
55	54.61	24.24	67.87

and precision is low, which is not promising. There could be two issues causing this:

- NB is not a proper modeling for this problem. Our intuition here is that the Naive Bayes assumption does not work well. For example a word could more likely cause a 0 classification if used with question words. This sort of information is ignored in NB by assuming the features are independent give the output.
- The tagging heuristic used here does not truly reflect if the user has been satisfied with the Cortana response or not.

Before resorting to other models, we try NB with FBT, and see if there are any performance enhancements. As the table 3 shows, the errors are still high

Table 3: Naive Bayes with Feedback Based Tagging

Data Category	Test error	Train error	Precision	Recall
C&C	60.59	60.57	37.49	93.31
Enriched	15	15	86.74	97.62

for C&C data, even worse than General data tagged with RBT. For Enriched data, as explained previously, due to high ratio of class 1 to class 0 samples(table 1) as well as unreliable feedback used by FBT, algorithm tends to predict everything as class 1. That is why the errors are low, however this does not mean the algorithm is performing well. We noticed in the predicted test data, only 10 out of 420 class 0 samples are truly predicted. We will later design a test to show this flaw with General data which also suffers from this issue. From now on, we will focus on C&C data which has more reliable query feedbacks.

Since using FBT did not produce any tangible performance enhancements, we will use other models to train and classify the data.

5.2 Support Vector Machine Classifier

We now focus our attention on using SVM with FBT and see how the results compare to NB. We

[1] The training and test error, precision and recall values in tables are all in percent.

used SVM with linear kernel implemented in LIBLINEAR library [2]. Since both our test and training data are unbalanced, we also experimented with changing the weights in SVM. Here is the C-SVM problem used by LIBLINEAR to solve this problem:

$$\begin{aligned} \min_{\omega,b,\xi} \quad & \frac{1}{2}\omega^T\omega + C^+\sum_{y_i=1}\xi_i + C^-\sum_{y_i=-1}\xi_i \\ \text{subject to} \quad & y_i(\omega^T\phi(x_i)+b)+b) \geq 1-\xi_i \\ & \xi_i \geq 0. \end{aligned} \tag{1}$$

We used C&C data for train and testing. Figures 1 and 2 show the results.

As it can be seen the result are more promising

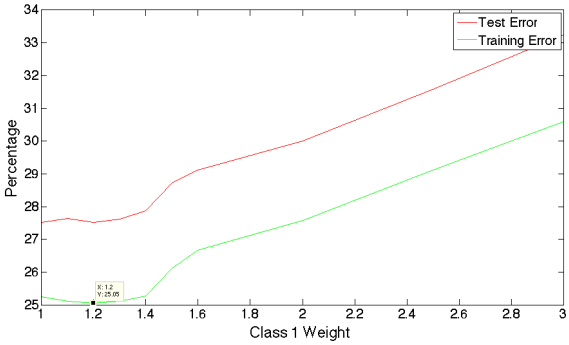

Figure 1: Test and Training Errors for Linear Kernel SVM with C&C Data using FBT

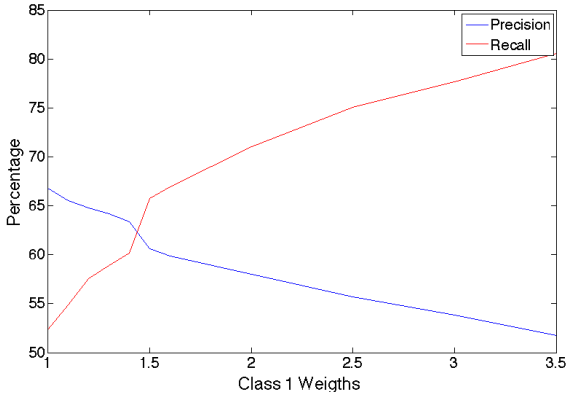

Figure 2: Precision and Recall for Linear Kernel SVM with C&C Data using FBT

with SVM. We can get as low as 25.7% test error with weight = 1.2 for class 1 with reasonable precision and recall. The optimal result considering precision and recall besides the test error, is with weight 1.5 for class 1: test error: 28.7%, recall: 60% and precision: 65.8%. Intuitively, this value for weight is close to the ratio of class 0 to class 1, which is 1.69(table 1). Here are some observations:

- As the class 1 weight is increased over the optimal point(∼1.4), both the training and test errors also increase. This is expected, as assigning higher weights to class 1 will cause the classifier to be more biased towards class 1. This will end up increasing the number of wrongly tagged class 1's(false positives).

- Precision and Recall have opposite trajectories in the figure. With increasing the weight, a greater percentage of actual class 1 samples are classified as class 1, however this will cause a higher percentage of class 0 samples to be wrongly also tagged as class 1. Thus we will have higher recall, but with less precision.

5.2.1 Bias vs Variance

We will analyze the test and training error versus training set size change to see whether we are running into over or under fitting issues.

The number of training samples is more than 20 times the number of features for C&C and General. For Enriched data, this value is 2.8. Considering the test and train error rates are very close, we are not facing variance or bias. Figure 3 shows this for C&C with Linear SVM. The values of training and

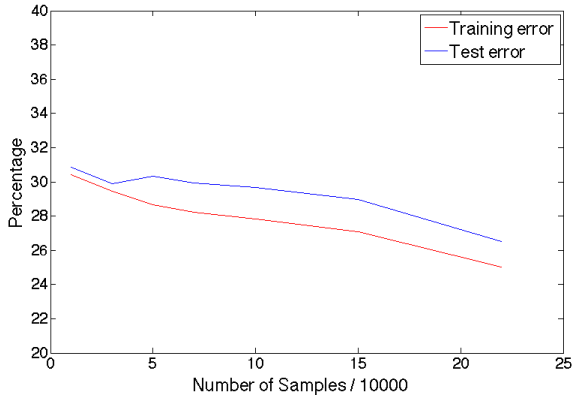

Figure 3: Training and Test error Vs. Training set size for Linear Kernel SVM with C&C Data using FBT

test error are staying very close to each other with changes in the training size. Also, the amount of the change is near 5% when changing the size from 10,000 to 200,000. Thus both our data sets and number of features are large enough so that are models are not suffering from either bias or variance.

5.3 Logistic Regression

As another popular classifier, we will now use logistic regression model to classify the C&C data. We will be using L1 and L2 norm logistic regression implemented in LIBLINEAR [2].

5.3.1 L1 Norm

The unconstrained primal optimization problem solved for this model is:

$$\min_{\omega} \quad \|\omega\|_1 + C\sum_{i=1}^{l} log(1 + e^{-y_i\omega^T x_i}) \qquad (2)$$

Where ω is the the vector of parameters. We will also experiment with weights as with SVM. Figures 4 and 5 show the results: The model demonstrates

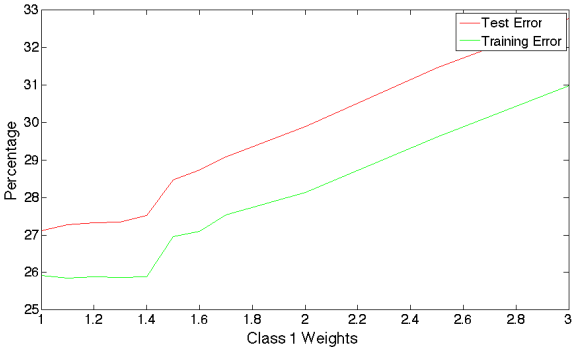

Figure 4: Test and Training Errors for L1 Logistic Regression with C&C Data using FBT

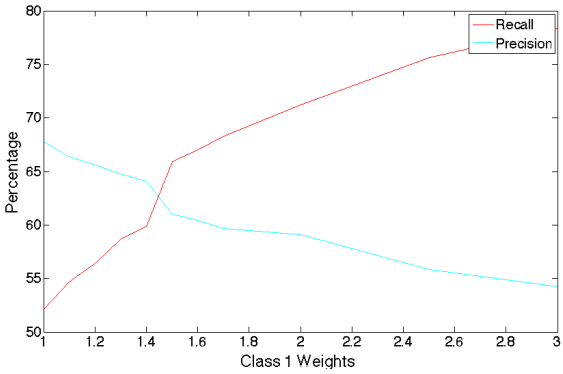

Figure 5: Precision and Recall for L1 Logistic Regression with C&C Data using FBT

very close performance to Linear SVM. Here are some notes:

- Weights of 1.4 to 1.6 yield the best results considering test and training errors as well as recall and precision.
- The similar trends in terms of training and test error as well as precision and recall is seen compared to SVM.

5.3.2 L2 Norm

$$\min_{\omega} \quad \frac{1}{2}\omega^T\omega + C\sum_{i=1}^{l} log(1 + e^{-y_i\omega^T x_i}) \qquad (3)$$

We experimented with L2 norm logistic regression, and the results did not show any tangible differences with L1 norm. We used 1.5 as weight for class 1, and ran the model over C&C dataset. As it can be seen from table 4, L2 norm model slightly outperforms L1 norm in terms of test error, however the difference is very marginal.

Table 4: L2 Vs. L1 norm Logistic Regression with C&C dataset

Model	Training error	Test error	Precision	Recall
L1	26.96	28.46	61	65.92
L2	26.27	28.3	60.62	65.79

5.3.3 Logistic Regression Vs. SVM

Both models perform very closely. Our intuition is that they both are solving very similar optimization problems, so it is expected that they perform closely. The regularization done in Logistic Regression models were not very useful, as we saw that due to the large size of our dataset, the algorithms are not experience high variance.

5.4 Testing queries with aggressive class ratio datasets

In order to see how well our model performs against naive algorithms, as well as how resilient it is with respect to the changes in the input query class ratio, we choose three different set of test samples:

- Test set containing of all 0's: This is to test against an algorithm(AlgBrute) that always predicts 1. In this case AlgBrute will have 0% accuracy.
- Test set containing of all 1's: This is to see how well our models perform agains an all positive data set. AlgBrute will always predict 100% here.
- Test set containing of 50% 1's and 50% 0's. AlgBrute will have 50% accuracy.

We will be using Linear SVM with class 1 wight of 1.5. This model's performance on the normal test set has been: Test error: 28.72%, Recall:65.78% Precision: 60.64%. Tables 5 shows the results with C&C data. As it can be seen, the classifier maintains its accuracy well. The precision and recall are by definition expected to be 0 and inf when there are no 1's, and precision is 100% when there

Table 5: Linear SVM with aggressive C&C test data

Test set	Test error	Precision	Recall
All 1's	33.96	100	66.03
All 0's	25.36	0	inf
50-50	30.03	71.88	65.79

are no 0's. For the other cases, both precision and recall maintain very close to normal training set performance with aggressive test data. This has the important implication that our model has not merged to become naive algorithms where one class is predicted all the time, or choosing one class with a certain probability every time. These two naive algorithms will perform very poorly with the aggressive tests.

We performed this experiment with General search data, which as mentioned before has the problem of unreliable tagging. We used 3 as the weight for class 1. Table 6 shows the results. As it can be seen, specifically with all 1's data, the algorithm seems to be doing a coin flip to predict the samples. Thus the training labels have not provided any net valuable information to the model. This proves our previous claim that the RBT is not reliable. We saw similar results with Enriched queries tagged by FBT.

Table 6: Linear SVM with aggressive General search test data

Test set	Test error	Precision	Recall
All 1's	44.9	100	46.57
All 0's	30.72	0	inf
50-50	37.83	56.04	46.57

6 Query Clustering

We start by using k-means [4] to cluster queries together. We used our own implementation of the k-means algorithm and then compared it to the Matlab build-in version for validation. Both implementation yielded very similar results for the same value of k and the same similarity metric. We used cosine similarity [5] as a similarity metric when performing the clustering step. We tried to use the EM algorithm as well, but it was a challenge to use our sparse bit-vector data representations to estimate Gaussian distributions, we plan on investigating using factor analysis to work around this in the future.

7 Conclusion

The purpose of this project was to predict the queries that existing Cortana system will not be able to handle with satisfactory results. These unsatisfied queries could later be handled by humans as a premium service. We started by Naive Bayes model and a repetition based tagging of General search queries. Our results showed poor performance with errors more than 55%. With improved tagging of data by using FBT, and better models such as SVM and Logistic Regression, we could reduce the test errors to as low as 28%. We also saw improvements in precision from 24% to 60%. Besides that, we noticed that SVM and regularized logistic regression models have very close performances. In addition, we showed that our SVM model with proper tagging(FBT) is resilient with respect to various ratios of class 1 to class 0 in the test samples, different from training set.

8 Future work

One of the main obstacles that we encountered was how to properly tag the data for test and training of the model. Inferring user satisfaction is very complex, and needs a more complex mechanism to detect. A more advanced way is to follow the actions that user takes after the Cortana interaction is over. This can give more accurate information to see whether user has been satisfied or not.

Another path to expand this work is to run the models over a larger set of data. We used only a day of Cortana data in this project. Our main obstacle was limited computation resources as well as inability of the implemented models to digest data sets larger than the available virtual memory of the machine. For example, the SVM we used requires loading the whole training sparse matrix, which becomes very large considering our large number of features and queries.

One area that we touched slightly and could be well investigated further is query clustering. We did clustering on the data, however we did not see meaningful clusters. Meaning that we were expecting to see clusters of reminders, meetings, etc. This was not observed by using k-means. More advanced algorithms such as EM could yield better results.

References

[1] Chang, Chih-Chung and Lin, Chih-Jen, *LIBSVM: A library for support vector machines*. ACM Transactions on Intelligent Systems and Technology, volume 2, issue 3, 2011.

[2] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang and Chih-Jen Lin *LIBLINEAR: A Library for Large Linear Classications*. Journal of Machine Learning Research, 2008.

[3] Levenshtein, Vladimir I *Binary codes capable of correcting deletions, insertions, and reversals*. Soviet Physics Doklady, 1966.

[4] MacQueen, J. B *Some Methods for classification and Analysis of Multivariate Observations.*. Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability : University of California Press 1967.

[5] Singhal, Amit *Modern Information Retrieval: A Brief Overview*. Bulletin of the IEEE Computer Society Technical Committee on Data Engineering. 2001.