Improving Government Response to Citizen Requests Online

Garren Gaut
University of California Irvine, CA
ggaut@uci.edu

Andrea Navarrete
University of Chicago, IL
andrean@uchicago.edu

Laila Wahedi
Georgetown University, DC
law98@georgetown.edu

Paul van der Boor
University of Chicago, IL
pvboor@gmail.com

Adolfo De Unánue
ITAM, Mexico
adolfo.deunanue@itam.mx

Jorge Díaz
National Digital Strategy, Mexico
jorge.diaz@presidencia.gob.mx

Eduardo Clark
National Digital Strategy, Mexico
eclarkgd@gmail.com

Rayid Ghani
University of Chicago, IL
rayid@uchicago.edu

ABSTRACT

The Mexican constitution guarantees its citizens the right to submit individual requests to the government. Public officials are obligated to read and respond to citizen requests in a timely manner. Each request goes through three processing steps during which human employees read, analyze, and route requests to the appropriate federal agency depending on its content. The Mexican government recently created a centralized online submission system. In the year following the release of the online system, the number of submitted requests doubled. With limited resources to manually process each request, the Sistema Atención Ciudadana (SAC) office in charge of handling requests has struggled to keep up with the increasing volume, resulting in longer processing time. Our goal is to build a machine learning system to process requests in order to allow the government to respond to citizen requests more efficiently.

We develop models to help at each stage: to accept or reject incoming requests, distinguish between technical support questions to a help desk and substantive requests to the SAC, and send substantive requests to the appropriate federal agency. Given the office's needs, our system can automatically accept or reject 85% of all requests, automatically send 39% of requests to a technical helpdesk or to the SAC, and automatically routes 49% of all requests to the most appropriate federal agency. These models each operate at a different step in the manual request sorting process. Incorporation of our models into the request system would save the office of the president 3.8 man hours per day accepting and rejecting requests, 3.1 man hours per day deciding whether requests should go to the helpdesk, and 4.8 man hours per day routing requests to federal agencies. Together, this would allow the government to process 24.4% more petitions, allowing them to better address citizen needs with limited resources.

COMPASS '18, June 20–22, 2018, Menlo Park and San Jose, CA, USA

ACM ISBN 978-1-4503-5816-3/18/06…$15.00
https://doi.org/10.1145/3209811.3209872

KEYWORDS

natural language processing, machine learning, user experience design, applied data science

ACM Reference Format:
Garren Gaut, Andrea Navarrete, Laila Wahedi, Paul van der Boor, Adolfo De Unánue, Jorge Díaz, Eduardo Clark, and Rayid Ghani. 2018. Improving Government Response to Citizen Requests Online. In *COMPASS '18: ACM SIGCAS Conference on Computing and Sustainable Societies (COMPASS), June 20–22, 2018, Menlo Park and San Jose, CA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3209811.3209872

1 INTRODUCTION

In 1917, the Mexican government ratified Article 8 of the Constitution [1], which gives citizens the right to written request. Article 8 states:

> *Public officials and employees shall respect the exercise of the right of request, provided it is made in writing and in a peaceful and respectful manner; but this right may only be exercised in political matters by citizens of the Republic.*
>
> *Every request shall be replied to in writing by the official to whom it is addressed, and said official is bound to inform the requester of the decision taken within a brief period.*

The request system provides the citizens of Mexico with a crucial conduit for interfacing with their government. The majority of requests come from poor states (at least 46% of population below poverty line). Requests range from questions on how to enroll in social services, assistance in accessing a pension fund, and requests for medical coverage. In some requests, it is obvious that citizens turned to this system when they didn't know where else to go. One extreme request was a plea for help; the citizen had been assaulted but the charges were dropped because the lawyer and witnesses had been physically intimidated. Given the gravity of some of these requests, it is important that there exist a system to process and respond to as many requests as quickly as possible. To satisfy this need, the government of Mexico created the *Sistema Atención Ciudadana* (SAC), whose sole responsibility is to receive requests and direct them to the federal agencies that ultimately provide responses.

Traditionally, citizens submitted handwritten requests to a brick-and-mortar location. However, in 2015 the Mexican government launched an online request submission portal to respond more efficiently to citizens. The submission process consists of the following steps (see Figure 1):

Figure 1: A step by step diagram of online request submission.

(1) **Step 1**: A citizen visits the online portal and lodges his or her request (www.gob.mx/atencion/). They receive a confirmation email and must verify their email address. Once the citizen has verified their email address, the request enters the system. Unverified requests are automatically rejected.
(2) **Step 2**: After email verification, a human receptionist reads the request and decides whether to reject it, or accept it for further processing. Requests can be rejected for having inappropriate or disrespectful language or for being unintelligible.
(3) **Step 3**: The receptionist decides to send the request to the SAC office or to a help desk. The help desk provides technical support for the website, and can help citizens find information or forms. All remaining requests are sent to the SAC office.
(4) **Step 4**: SAC analysts send requests to the appropriate federal agency via the agency's contact person. Requests are routed to the different federal agencies according to the subject and the knowledge the analyst has of the federal agencies. If the agency can handle the request, they write a response. Analysts in the SAC coordinate with the federal agency to ensure that the request is resolved. If the federal agency cannot handle the request, they send it back to a SAC analyst to send to a different federal agency. Once the request is approved by the analysts, an answer is sent to the citizen.

A striking problem with the current system is that the supply of labor for processing requests is struggling to accommodate an increasing number of requests submitted. The number of online requests submitted daily doubled from ≈100 requests in 2015 to ≈ 200 requests in 2016, and the government anticipates further increases in request submissions as the website is publicized. The SAC processes requests manually, and cannot afford to hire additional staff. To keep up with the increasing volume of requests, the SAC needs a scalable approach to processing requests.

The goal of this project is to improve the lives of the Mexican population by increasing the number of requests that the request system can efficiently process. We focus on 1) increasing the number of, and speed at which, citizen requests are addressed and 2) improving the scalability of the online request response system by developing automated methods for request processing. This automation will enable some percentage of requests to be processed and routed automatically while the rest will be handled manually (based on the system's confidence). Each step in this process has specific bottlenecks that we highlight in the following sections.

2 STEP 1: EMAIL CONFIRMATION

2.1 Problem

During initial data exploration, we discovered that many requesters fail to verify their contact via the confirmation email: 32% of all incoming requests remain unconfirmed. The purpose of this step is to filter spam, but we found that many unconfirmed requests contain legitimate requests from citizens. Neither the email nor the web portal clearly expressed to the user that he or she needed to take action on the confirmation email in order to have his or her request read (See Figures 2 and 3).

The email contained a text hyperlink that was easily overlooked, an email header that translates to *'Your request is being Processed'* (when in reality the request will only be processed after email confirmation), an email subject that suggests that the request has been submitted (rather than requesting confirmation from the citizen), and an informal sender address (*sac@presidencia.gob.mx*) that does not convey the official nature of the email.

The website also failed to communicate that a citizen would be receiving an email that must be responded to. The site has a misleading title (*'Your request is Being Processed'*), and only at the bottom of the page informs the user that a confirmation email was sent and contains a link that needs to be clicked on. Furthermore, the hyperlink text is not highlighted or made visually distinct from other text on the page.

2.2 Solution

We provided a new version of the confirmation email that clearly expressed to the user that he or she needed to take action (see figures 3 and 2). We changed the body of the email to explain that email will be the primary form of contact between requester and respondent, and thus must be confirmed in order to proceed. We used strategies from behavioral science research and widely used in digital marketing to encourage citizens to confirm their email, including providing a large red button linking to the confirmation webpage (rather than a hypertext link), changing the sender email account to reflect a formal sender, and changing the email subject to indicate that action is necessary.

In the new version of the website, we changed the title to indicate that the user has not yet finished submitting a request (*'You Have One More Step'*), updated content to inform the user that a confirmation email was sent, and added step by step instructions for verifying the user's confirmation email.

Figure 2: On the left is the old confirmation email. Highlighted are misleading email titles and headers, informal sender address, and the use of a text hyperlink rather than a button. On the right the new confirmation email. We changed misleading email titles, sender address, and used a button rather than text hyperlink to link to confirmation page.

Figure 3: On the left the old website with Original Spanish version and English translation. On the right is the new website with Spanish version and translated version.

2.3 Evaluation

We used A/B testing to evaluate whether the new email and web portal would lead to an increase in confirmation rate. For a two-week period (July 27th to August 8th, 2016), we collected new requests (≈2500). For each new request, we randomly and uniformly gave the requester either A) the new versions of the website and email or B) the old versions of the website and email. We performed a two-way t-test to test the difference in confirmation rates between the two versions.

2.4 Results

We found that the new website and confirmation email resulted in a 39% ($p < 1 \times 10^{-33}$) increase in the rate of confirmed requests, compared to the old website and confirmation email. We believe that the large emphasis placed on advising the citizen that they had one more step to complete is the driving force behind this increase.

It's important to note here that simple behavioral science techniques and A/B testing can have a big impact on response rates in these types of problems.

Our partners at the Office of National Digital Strategy in México [2] have implemented the changes to the email and website changes, and the rate of unconfirmed requests has decreased from 32% to 19%. We also provided recommendations for further improvements, and advised them to continue running A/B tests to test the effects of minor changes to citizen communications.

3 STEPS 2-4: REQUEST AUTOMATION

After a requester confirms his or her email, the SAC reads the corresponding request. Independent subdivisions of the SAC (2) accept or reject a request, (3) send a request to a technical help desk or to another analyst for agency routing, and (4) route requests to the appropriate federal agency for response. We implement an automatic routing system to reduce man hours spent reading requests and increase the scalability of the entire request system.

Automatic routing of requests can be viewed as a multi-label document classification problem [5, 7, 13] where each request can have up to two classes (i.e., accepted/rejected, sent to helpdesk or sent to a given federal agency). Generally, there are three ways to handle a multi-label classification: convert into a multi-class classification problem using power-sets, converted into multiple binary problems (i.e., the Binary Relevance (BR) method), or adapt the learning algorithm, [5]. Since each stage of our problem is performed by independent agents, we use the BR method to mirror the independent nature of our problem domain and allow careful control of performance at each stage. Additionally, the BR method allows for problem scalability, reduces overfitting of infrequent labels, and allows for flexible inclusion and exclusion of labels in a potentially changing government environment [11].

Another important modeling choice in text classification is whether to use count or vector based embeddings [4, 6, 8]. We chose to use a count embedding approach rather than vector space encodings to easily interpret the relationship between features and model parameters and to reduce computational demand.

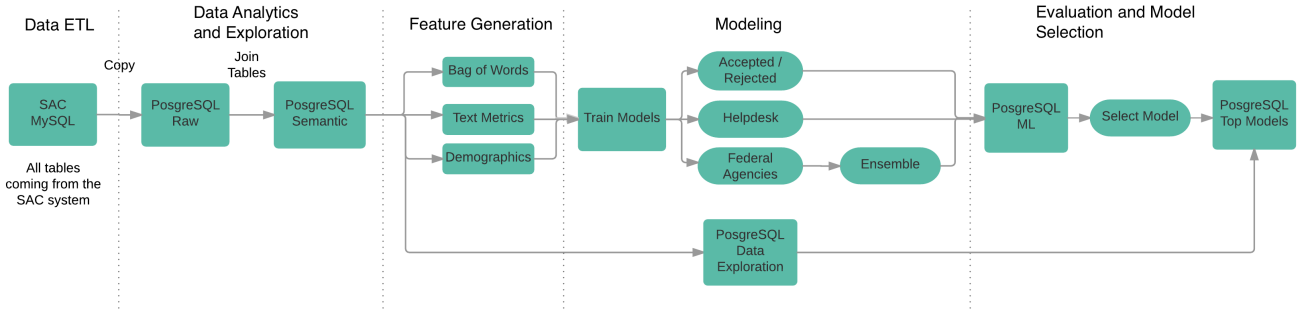

Figure 4: Computational pipeline.

We implemented machine learning models for each step in the request process (2-4) (see figure 1). We built a machine learning pipeline to run all sequential tasks: 1) Data ETL, 2) Data Analytics and Exploration, 3) Feature Generation, 4) Modeling, and 5) Evaluation and model selection (see figure 4).

3.1 Data

Our data comes from the SAC and contains information about each request, the citizen who submitted the request, and how the request was routed.

The data set consists of 69,402 online submissions from October 2014 to May 2016. Each online submission contains the text of the request, demographic information about the requester, and details from each processing step performed within the SAC, including who read the request, where the request was sent, and the final government agency that responded to the request.

The age of requesters ranges from 17 to 67 years, but half of all requesters are under 37 years old. The skew toward lower ages is most likely due to increased facility with computers among younger populations, increasing their ability to submit online requests. Men submitted more requests than women (56%, and 44% of all requests, respectively). Almost all requests (98%) were submitted in Mexico. The states that submitted the largest percent of all requests are also the states with the largest populations: Mexico City (19%, 8 million people), the State of Mexico (16%, 16mil), Jalisco (7%, 7mil), Veracruz (7%, 8mil) and Guanajuato (4%, 5mil). Out of the 47% of respondents that provided occupational status 28% were employed, 9% unemployed, 5% students, and 5% housewives.

We also computed descriptive statistics about the text of the requests to provide insight into feature generation for the machine learning models. Rejected requests and requests sent to Helpdesk are shorter than those sent to SAC or federal agencies (see Figure 5). Since, requests may be rejected if they contain curse words, we computed how many curse words each request contained. Less than 1% of requests contained curse words those that did contain an average of 1.47 curse words. A single request might mention one or more agencies, so we computed how often each federal agency was mentioned (either its full name or abbreviation). From the entire corpus of requests, 44 agencies were mentioned at least once. The top mentioned agencies were IMSS (1724), ISSSTE (1157), INFONAVIT (795), SEP (703), and SAGARPA (380).

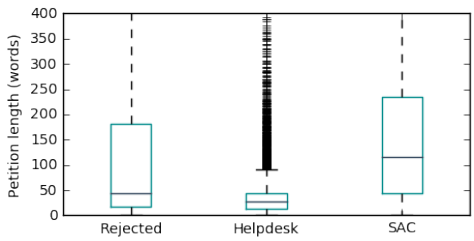

Figure 5: The length of requests for each step of the request process. Requests that were rejected or sent to the help desk are shorter than accepted requests.

3.2 Feature Generation

Our set of generated features consists of text statistics, document attributes, and demographics (see Table 1).

We created multiple Bags-of-words (BOWs) by varying 1) whether to apply a term frequency-inverse document frequency (TFIDF) transformation, 2) whether to tokenize documents into unigrams or unigrams+bigrams, and 3) whether to stem tokens. Request preprocessing resulted in 8 different BOWs: one for each combination of parameter options. We removed a custom set of Spanish stopwords manually selected to have little semantic value. When fitting our models, we consider each BOW a separate corpus representation and test which representation leads to better performance.

We generated document attribute features by computing document level statistics such as the number of times curse words were used in a request, the length of the request, the number of words removed during pre-processing (as a marker of how different requesters expressed themselves), and the number of times each federal agency was mentioned in request text.

We used demographic information to generate categorical features such age, sex, location, and occupation. We also included the time of day when the request was sent. Since most of the demographic fields were not mandatory, we added additional binary features to flag whether the requester provided each demographic variable.

Table 1: Set of all features used in all models.

Set name	Features
BOW	8 BOWs
Document Statistics	request length (words/characters), request length after processing, number of words removed during preprocessing, number of curse words used, whether each agency was mentioned (one feature per agency), number of times each agency was mentioned (one feature per agency)
Demographics	gender, occupation, nationality, whether requesteris Mexican, time of day submitted

3.3 Modeling

We built machine learning models for automating steps 2-4 of the request submission process (accept/reject, help desk/SAC, routing to federal agencies). Each model takes features computed from one submission (request and requester) as input and returns a class label as output.

We fit our models on different subsets of the dataset. The accept/reject model was fit on all requests in the dataset. The help desk model was fit on all the accepted requests since its creation in January 2016. The federal agency model was fit on all requests sent to the SAC. By using all applicable data to train each model, we treat each step as independent. This gives the best possible estimates of how much time we could save at each step, but may not accurately reflect system performance across steps.

For all steps, we performed a grid search over model types and parameters to optimize performance (See table 2 for a list of all model and parameter combinations). For step 2, we fit binary classifiers to predict whether a request would be accepted or rejected. For step 3, we fit binary classifiers to predict whether a request would be sent to the help desk.

For step 4, we fit independent binary classifiers for sending a request to each of the top five most solicited agencies (see Table 3). We focus on the top five most solicited agencies because they make up a plurality of request submissions. Other agencies had too few submissions to reliably classify. Focusing on the top five allowed us to create a classifier that could reduce the load on the SAC by handling the more common requests while minimizing error.

We compare two types of decision frameworks: one in which a request can be sent to one and only one agency (multi-class classification), and another in which a request may be sent to multiple agencies (multi-label classification). While the former approach fits more closely with current SAC practices and is therefore easier to implement, we find significant performance improvements in the latter, potentially allowing the SAC to process more petitions faster.

For the single-label case (a request can be sent to only one agency), we compare our threshold optimization routine and decision rule to a decision tree, and find that the threshold optimization routine performs better.

Threshold Optimization. Binary models traditionally produce classifications by defining a score threshold; examples with scores above the threshold are classified as 1 and examples with scores below the threshold are classified as 0. To produce a classification from a set of scores, we must find a set of thresholds (one per agency) and a decision rule (i.e., how to classify a request with multiple scores above the respective thresholds) that produce optimal classifications. Our threshold optimization routine searches over all possible combinations of score thresholds and selects the set of thresholds that gives the highest precision given a minimum recall of 0.15 (see Algorithm 1). In order to reduce computational demands, we run two iterations of the optimization procedure. On the first, we use larger-spaced thresholds. In the second, we manually select high-performing threshold ranges in consultation with the office on acceptable failure rates given the error rate of human classification.

Decision Tree. We compared our threshold optimization routine to a decision tree that selects an agency from the binary classification scores. For a single request, the tree takes scores from the top models for each agency as input and produces a categorical label. The label indicates which agency to send the request to or, alternatively, if the request should be sent to another federal agency.

Multi-label Agency Classifications. We also explore a decision framework that allows a request to be sent to multiple agencies. The intuition behind this approach has two parts. First, if a request is sent to more than one agency at the same time, this in effect parallelizes the current process, allowing for faster processing time overall. Second, we could provide a list of possible classifications to analysts to reduce classification errors, and speed up the time it takes to manually classify a request. We expect automation to allow the SAC to handle more requests. However, if too many redundant requests are sent to federal agencies, then any time-saving benefit will decrease as they divert energy toward handling misclassified or duplicate requests. Whether this approach is effective will depend on emerging bottlenecks as the system is deployed, and optimizing between the two approaches will require continued testing of the deployed system.

As in the first decision framework, we use a score threshold for classification; however, here we use a cost matrix to optimize the set of recommended classifications. The cost matrix allows us to weigh the consequences of false positives differently than false negatives. A false negative is very costly because eventually the request will have to be sent through the manual system. A false positive is less costly. As long as the correct federal agency is among the set of all agencies a request is sent to, that request will not have to re-enter the manual system. The set of possible classifications and corresponding costs for a request are as follows:

- **Cost = 1**: A request is classified as not belonging to any of the top five agencies, and we send it through the manual system. This case does not require any additional work over the current manual system, so we assign it a low cost.
- **Cost = 0.5 $\times$(N − 1)**: N classifications to agencies are produced, of which one classification is correct. Since we sent the requests to $N - 1$ incorrect agencies, we penalize by the number of

Table 2: Models and parameter values iterated over in the pipeline.

Model	Parameters	Range
Random Forest	estimators, max depth, min sample split penalty	[100-3000], [10-500], [5,10] [0.001,0.1,1,10]
Logistic Regression	penalty	[0.00001,0.001,0.1,10]
Naive Bayes	-	-
Linear Support Vector	penalty	[0.001,0.1,10]

Table 3: The top five most solicited federal agencies and number of requests sent to each agency between October 2014 and May 2016.

Agency	Abbr.	#Pet
Secretaría de Educación Pública	SEP	4,112
Dirección General de Atención Ciudadana	DGAC	3,980
Secretaría de la Función Pública	SFP	2,197
Secretaría de Salud	SS	1,492
Secretaría de Economía	SE	1,315

incorrect agencies we send to. If we set the cost for sending a request to the incorrect agency as 1 or higher, then this analysis is unnecessary, since it would be less costly to send any request with more than a single classification through the manual system. This simplifies to the case in the single classification framework that we described above.

Cost = 2×N: N classifications to agencies are produced, and they are all incorrect. We penalize this case heavily because it wastes time, and we will have to send the request through the manual system after agencies deny the request.

We then use a threshold optimization routine for classification. Given a set of score thresholds and model scores for a request, the threshold routine classifies a request as follows:

(1) If zero model scores are above their respective score thresholds, we classify the request as belonging to an agency outside the set of agencies we are considering.
(2) If two or more agency scores are above their respective score thresholds, we send the request to all agencies with scores above threshold.

Similar to the threshold optimization routine for the single-classification framework, we iterate over all possible threshold sets. We compute a total cost for each threshold, and find the optimal threshold by choosing the threshold set with minimum cost that sends at least 15% of all requests to one or more federal agencies.

3.4 Evaluation and Model Selection

The general objective of our models (accept/reject, helpdesk, federal agency) is to automate as many requests as possible while maintaining a low error rate, where error rate is defined separately for each step in the request process. A key aspect of our problem scope is that our models do not have to automate *all* requests, since any requests we are not confident about can be sent through the existing manual system. Therefore, we only automate the requests that we are most confident about, and optimize the number of requests processed automatically at a given error rate. We worked with the National Digital Strategy office to decide acceptable error rates and found that the maximum error rate allowed at any step is 6%. Thus

input : requests (P), class labels (C), set of classification models M, $n = |M|$, integer k
output: best threshold set $t_{\max}$

for *iteration = 1:2* **do**
  if *iteration == 1* **then**
    // evenly-spaced threshold generating set;
    $h_m = [0, \frac{1}{k}, \frac{2}{k}, ..., \frac{k-1}{k}]\ \ \forall m \in \{1, 2, \ldots, n\}$
  end
  // generate threshold superset, all combinations of thresholds for each model;
  T = $\{[h_1(j_2), h_2(j_2), \ldots, h_n(j_n)] \mid j_i \in \{1, \ldots, k\}$ and $\forall i = 1, \ldots, n\}$;
  for *threshold set* $t \in T$ **do**
    for *request* $p \in P$ **do**
      $s = [M_1(p), M_2(p), ..., M_n(p)]$ // model scores;
      if $sum(s > t) == 1$ **then**
        $C_{\text{out}}(p) = c : s_c(p) > t_c$ // predict class
      else
        $C_{\text{out}}(p) = -1$ // send to SAC
      end
    end
    $PREC(t) = \text{precision}(C_{\text{out}}, C)$;
    $REC(t) = \text{recall}(C_{\text{out}}, C)$;
    // correct prediction if the correct class or we send to SAC and correct class is not among candidate models
  end
  $t_{\max} = \text{argmax}_{t \in T: REC(t) > 0.15}\, PREC(t)$;
  // set threshold generating sets for next iteration;
  for $m \in \{1, ..., n\}$ **do**
    $h_m = [a_m : \frac{b_m - a_m}{k} : b_m]$ // a_m, b_m chosen by visually inspecting range of thresholds that gave highest precision for each model
  end
end

Algorithm 1: Threshold Optimization Routine

in general, the metric we will optimize is the maximum number of requests we can classify at an error rate of 6%. We also report the number of additional man hours our models would save.

All model evaluation for single binary models is performed in a 10-fold cross validation routine. We use a stratified (by federal agency) 5-fold cross validation routine for training and testing multi-class federal agency classification.

3.4.1 Step 2 Accept/Reject. We evaluated our accept/reject model according to the maximum number of requests we can accept while guaranteeing that 94% of accepted requests are true positives. Specifically, we maximize the percent of requests that we can classify with a precision of 94%. Our modeling pipeline automates this process, and allows the SAC to choose any precision threshold. Importantly, we only automate requests that our model is highly confident will be accepted. This avoids the ethical problem of automatically rejecting requests, since rejecting a request that should have been accepted denies a citizen their constitutional right to request.

3.4.2 Step 3 Helpdesk/SAC. We evaluated our help desk model according to the maximum number of requests we can automate by funneling accepted requests to either the SAC or the help desk. For this model, the cost of misclassification is not as high, since misclassified requests remain inside the request system and are manually re-routed to the correct location. Specifically, we take the model that maximizes

$$\max_m p_m^{0.94} + 1 - r_m^{0.975},$$

where $p_m^{0.94}$ is the percentage of requests classified when help desk precision for model m is 0.94 and $r_m^{0.975}$ is the percentage of requests classified as help desk when help desk recall is 0.975. Intuitively, maximizing $p_m^{0.94}$ gives the most requests we can send to the help desk with 6% error. Maximizing $1 - r_m^{0.975}$ gives the minimum percentile at which we attain a recall of 0.975, representing the a high separability between help desk and SAC scores.

3.4.3 Step 4 Federal Agency. For each federal agency, we searched for the binary classifier that allowed us to route the most requests with the minimum error rate. For each agency, we selected the model that optimizes

$$\max_m p_m^{0.94}$$

where $p_m^{0.94}$ is the percentage of requests classified as being sent to an agency when precision for model m is 0.94.

Using the best models for each federal agency, we compute a decision rule, either by decision tree or threshold optimization. For the framework where a request can be sent to just one agency, we evaluate models (decision tree vs. threshold optimization) by searching for the model with highest overall precision. For the best model, we report the precision, recall, and the number of requests we could automate at that precision. We evaluate the multi-label classification decision framework by comparing the total cost associated with our model to the total cost of the current manual system.

3.5 Results

The best accept/reject model is a Random Forest with 1000 trees, 500 maximum depth, 5 minimum sample split and log_2 maximum features using document statistics and bags of words as features (See figure 6 for a histogram of scores and true classes). The model can classify accepted requests with an error rate of 8.8%, but this is a small improvement upon the baseline error rate of 9.2% obtained by classifying all requests as accepted.

If we take only the requests that we are most confident in accepting, we can automate 85% of all requests with a 6% error rate, which would automate 170 requests and save 3.8 man hours per day (at a rate of 200 requests in an 8 hour work day).

We looked into the data for reasons for this low performance, and found that content was not greatly discriminative for accepted and rejected requests. We believe that incorporation of new features and manual pruning of the dataset may lead to improved performance, but we leave this to future work.

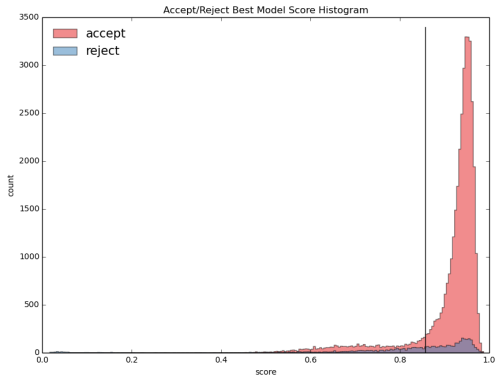

Figure 6: Density of random forest scores for accepting and rejecting requests. The threshold is drawn at a 6% error rate, which allows us to automate 85% of requests.

The best performing help desk/SAC model is a Random Forest with 1000 trees, 100 maximum depth, 5 minimum sample split and log2 maximum features using the demographic, bag of words, and text metric features (See figure 7 for a histogram of scores and true class). The vertical lines on the figure show score cutoffs that define score ranges that correspond to where to send requests: scores above the upper threshold are sent directly to the SAC, scores below the lower threshold are sent directly to the help desk, and scores in between are sent to a receptionist to process. The thresholds were chosen so that of the automated requests, there is only a 6% error rate. We found that using these thresholds we can automate 39% of requests at this step, saving approximately 3.12 man hours per day (at a processing rate of 200 requests in an 8 hour workday).

For federal agency routing, we took the models that best automate requests by looking at the percent we could automate at an error rate of 6%, i.e., the percent of requests we could automate with precision of 0.94 (See Table 4). The model for routing requests to Secretaría de Educación Pública (SEP) provided the best individual level of automation and was able to automate 3.6% of requests at a 6% error rate. The next best models were for automating to Secretaría de Salud (SS,.09%), the Dirección General de Atención Ciudadana (DGAC,.06%), Secretaría de la Función Pública (SFP,.02%),

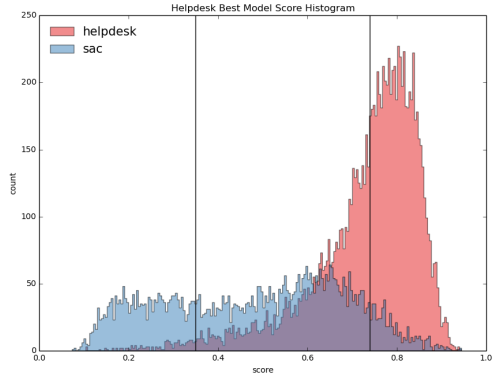

Figure 7: Density of random forest scores for sending requests to the help desk or the SAC. The thresholds are drawn at the positions where we can automate the most requests with an 6% error rate, allowing 39% of all requests to be automated.

Table 4: Performance, parameters, and features used in our best performing models. LR=Logistic Regression, RF = Random Forest. Perc. at 0.94 is the percent of all requests we can classify with a precision of 0.94. This is the metric we maximize to choose the best model.

Agency	Mod.	Parameters	Corpus	Perc. @ .94
SEP	RF	max_depth=100 max_feat=log2 n_estim=3000 min_samp_split=10	tfidf=0 1-2gram stem=0	3.6
SS	RF	max_depth=500 max_feat=log2 n_estim=1000 min_samp_split=10	tfidf=0 1-2gram stem = 0	0.09
DGAC	LR	C=1 norm=L1	tfidf=0 1gram stem = 1	0.08
SFP	LR	C=10 norm=L1	tfidf=1 1-2gram stem=0	0.02
SE	LR	C=1 norm=L1	tfidf=0 1gram stem=0	0.01

and Secretaría de la Economía (SEP, .01%). Were we to incorporate a single model for agency routing, in this case the model for routing to SEP, we would be able to automatically route approximately 2-3 requests per day and save the SAC 1.14 man hours per day (3 analysts and one supervisor processing 200 total requests per 3.53 hours, given time spent on handwritten requests).

We evaluated our threshold optimization routine for producing single classifications from all independent models, and compared it to a decision tree in order to provide an option that could fit into the current SAC process. We also evaluated the performance of the threshold optimization routine in a multi-label agency classifications setting.

Threshold Optimization. The threshold optimization routine performed better than the decision tree, resulting in a precision of 0.84, recall of 0.25. The threshold optimization routine would send 14% of all requests to one of the top five dependencies, which would save 2.03 man hours per day. However, the precision for the threshold optimization routine is still too low to satisfy our partners requirements. We believe that this poor performance is in part because in the available data there are relatively few requests sent to each agency. As the online request system matures and more requests are submitted to the top agencies, we hope that precision will increase enough for the system to be implemented. Because of this poor performance, we also presented the multi-label classification framework.

Decision Tree. For the single classification federal agency decision framework, the decision tree resulted in a precision of 0.203, recall of 0.438, and accuracy of 0.55. The model had trouble distinguishing requests that should be sent to DGAC from requests that should be sent through the manual system; 47% of requests sent to the DGAC were classified by the model as manual system, and 14% of requests sent through the manual system were classified by the model as DGAC. This result is not surprising since DGAC, the office that processes requests, is the final agency to send out a response, and most likely receives many requests inquiring about responses to other requests. The threshold optimization routine did a better job of distinguishing between agencies with the training data available so far.

Multi-label Classification. The threshold optimization routine for multi-label classification performed better than an all-manual baseline, on average resulting in $\approx 66\%$ of the cost of running the manual system (18470.5 versus 27546). A cost of 1 corresponds to sending the request through the manual system, which is equivalent to 0.2748 man hours (three analysts and one supervisor processed on average 51.36 online requests in 3.528 hours, given handwritten request processing time). Thus, the multi-label classification system would have saved the SAC 4.7987 man hours a day over the 18 month period over which the data was collected, and to process an additional 17.46 petitions per day.

3.6 Technical Implementation

The pipeline is implemented using the pipeline manager tool Luigi [3] that allows the system to be deployed at scale. Luigi helps build complex pipelines of batch jobs by automatically handling dependency resolution. For example, for each task in our pipeline, we specify three functions:

(1) **requires**: returns the output that must exist for the task to run
(2) **run**: runs the task
(3) **output**: returns the output of the task.

Luigi automatically checks whether the required input exists for each task, and only runs a task if its output does not exist or if its input has changed. We avoid redundant calculations by separating each calculation into modular tasks. This enables the SAC office to both automatically retrain the models as new data becomes available, and avoid time-intensive recalculations when adjusting score thresholds and other model specifications.

All computation was run using Amazon Cloud Computing, databases were hosted on Amazon Cloud Computing servers, and output was stored on Amazon S3. All code is available on github [9].

To have access to a larger suite of analytical tools and database resources, we migrated the MySQL database into a PostgreSQL database. We then created a non-normalized schema for feature generation and model building. Feature generation was done using PostgreSQL, sci-kit learn, and gensim [12]. Modeling was implemented in sci-kit learn [10] and we used luigi to perform a grid search over all models and parameters.

To easily deploy our system on the SAC's servers, we wrote Docker containers to initialize and run the pipeline. Docker containers are virtual environments that wrap software in a complete file system that contains everything needed to run: code, runtime, system tools, system libraries. Docker guarantees that software will run consistently, regardless of its environment. Our Docker files contained all the necessary code for setting up a Luigi server that runs all pipeline tasks.

3.7 System Integration

To seamlessly integrate our automated system, models will be initially tested side-by-side with the receptionist and analysts. For each model that accurately automates a step of the process, that model may be deployed. This testing period will build trust in the system as users (SAC employees) see how it works. It will also be an opportunity to get user feedback and determine acceptable error rates.

To ensure that our models remain robust to changing request topics, we recommend continually generating unbiased labeled data; a percentage of requests that were confidently classified should still be sent to the receptionist or the SAC. These labels should be used to re-train, re-test, and re-select top models at regular time intervals to ensure that models change over time with the data and continue to automate the maximum number of requests possible at acceptable error rates.

It is extremely important that the accept/reject model only automates high-confidence accepted requests. If a request is rejected that should not be rejected, we are denying a citizen access to his/her constitutional right and undermining the goals of the request system.

4 FUTURE WORK

A limitation of our work is that our models at each step were fit independently on perfect data from the proceeding step. For example, we did not test the case where our accept/reject model erroneously accepts a request and then sends that request to our help desk model. Since the accept/reject model automates the first step in the processing system, our results are accurate for this step and the model will still save 3.8 hours per day of manual processing time. However, we expect the help/desk and federal agency routing error to be slightly larger in practice. Future work will involve integrating models into a single system, optimizing for system-wide performance, and analyzing system-wide error propagation.

Our evaluation framework can be improved to account for changing topics in the submitted requests. Since the data came from a 1.5 year time period, we don't think that this greatly impacted results. However, as the system matures, incorporating temporal changes in the data will become increasingly important. We plan to do this via temporal cross validation and periodically refitting our classifiers as additional data are generated.

Another method by which the evaluation framework should be improved is to further investigate the biases in our dataset. During data exploration, we looked at the demographics most likely to submit a request, but should expand these analyses to whether certain demographics are associated with accepting/rejecting requests or to which agencies requests are sent. In the case that systematic biases exist, further analysis should be conducted to determine whether biases are a result of request content or the decision making process. Together with the SAC, any decision-making biases should be addressed in incoming data streams. Importantly, the causal mechanisms of these biases need to be studied in order to understand how to correct them. We cannot correct artificial intelligence biases without correcting our own biases.

5 SUMMARY

The presented work improves the request processing system of the government of Mexico by increasing the number of requests read by the government and developing automated methods for request processing. Using our work, the SAC will be able to process and respond to more requests in a timely manner. The government of Mexico will be able to fulfill the guaranteed right to request and the lives of Mexican citizens will improve.

6 ACKNOWLEDGMENTS

We thank the Eric & Wendy Schmidt Data Science for Social Good Fellowship for generously supporting this work, our partners at the Office of National Digital Strategy for sharing data, expertise, and feedback for this project, and all staff at Sistema Atención Ciudadana for their help and support.

REFERENCES

[1] Constitución política' de los estados unidos mexicanos (artículo 8). http://www.diputados.gob.mx/LeyesBiblio/htm/1.htm, 1917.
[2] Mèxico digital. https://www.gob.mx/mexicodigital/, 2015.
[3] E. Bernhardsson and E. Freider. Luigi [computer software]. https://github.com/spotify/luigi, 2016.
[4] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391, 1990.
[5] T. G. and K. I. Multi-label classification: An overview. *International Journal of Data Warehousing and Mining (IJDWM)*, 3.3:1–13, 2007.
[6] C. D. Manning, P. Raghavan, H. Schütze, et al. *Introduction to information retrieval*, volume 1. Cambridge university press Cambridge, 2008.
[7] C. D. Manning, H. Schütze, et al. *Foundations of statistical natural language processing*, volume 999. MIT Press, 1999.
[8] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.
[9] A. Navarrette, L. Wahedi, G. Gaut, A. D. Unanùe, E. Potash, and R. Ghani. Improving government response to citizen requests online. https://github.com/

dssg/atencion, 2016.

[10] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[11] J. Read, B. Pfahringer, G. Holmes, and E. Frank. Classifier chains for multi-label classification. *Machine learning*, 85(3):333, 2011.

[12] R. Řehůřek and P. Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA. http://is.muni.cz/publication/884893/en.

[13] F. Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys (CSUR)*, 34(1):1–47, 2002.