University of the Philippines Manila
College of Arts and Sciences
Department of Physical Sciences and Mathematics

TAGR: Telehealth Automatically Generated Recommendations

A special problem in partial fulfillment
of the requirements for the degree of
Bachelor of Science in Computer Science

Submitted by:

Kryle Marxel E. Molina

June 2016

Permission is given for the following people to have access to this SP:

Available to the general public	Yes
Available only after consultation with author/SP adviser	No
Available only to those bound by confidentiality agreement	No

ACCEPTANCE SHEET

The Special Problem entitled "TAGR: Telehealth Automatically Generated Recommendations" prepared and submitted by Kryle Marxel E. Molina in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science has been examined and is recommended for acceptance.

____

Marvin John C. Ignacio, M.Sc.(*candidate*)
Adviser

EXAMINERS:

	Approved	Disapproved
1. Gregorio B. Baes, Ph.D. (*candidate*)	________	________
2. Avegail D. Carpio, M.Sc.	________	________
3. Richard Bryann L. Chua, Ph.D. (*candidate*)	________	________
4. Perlita E. Gasmen, M.Sc. (*candidate*)	________	________
5. Ma. Sheila A. Magboo, M.Sc.	________	________
6. Vincent Peter C. Magboo, M.D., M.Sc.	________	________

Accepted and approved as partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science.

____

Ma. Sheila A. Magboo, M.Sc.
Unit Head
Mathematical and Computing Sciences Unit
Department of Physical Sciences
and Mathematics

____

Marcelina B. Lirazan, Ph.D.
Chair
Department of Physical Sciences
and Mathematics

____

Leonardo R. Estacio Jr., Ph.D.
Dean
College of Arts and Sciences

Abstract

Telehealth Automatically Generated Recommendations (TAGR) is a module for automatically classifying SMS and e-mail messages with their appropriate tags as required by the National Telehealth Center for their National Telehealth Service Program (NTSP). It utilizes support vector machines in text classification which are configurable in a web-based interface. The default classifier trained on the base dataset gives an average accuracy of 73% via 10-fold cross validation.

Keywords: natural language processing, text classification, machine learning, support vector machine

Contents

List of Figures

List of Tables

I. Introduction

A. Background of the Study

In most countries, providing healthcare to the public is a responsibility of the government. As such, governments will have bodies and mechanisms in place for a working public health system. In the Philippines, the Department of Health (DOH) fulfills this role as the country's national health policy maker and regulatory institution [1]. Part of its mandate is attaining universal healthcare which translates to accessible and equitable quality healthcare for every Filipino. Its mission and vision places emphasis on guaranteeing health especially for the poor [2].

This raises the issue of ensuring that healthcare is accessible even to those living in remote and rural areas. There is a disparity of healthcare attention in rural and urban areas, leaning towards the latter. This is evidenced by the abundance of health centers and personnel near metropolitan areas in contrast with the areas far from urban centers. Hospitals in the National Capital Region (NCR) and Region IV-A make up 17% of all hospitals in all regions, while hospitals in the Autonomous Region of Muslim Mindanao (ARMM) and Region XIII together only amount to 7%. In perspective, Region IV-B has a slightly smaller population than that of ARMM, but it has almost twice the number of hospitals than the latter [3]. Consequently, health workers are more concentrated in Regions III, IV-A and the NCR than in Mindanao. However, the shortage of health workers does not stop at the level of urban-rural differences. The Philippines, in general, does not have the sufficient amount of health professionals to see to the needs of the population. One cited cause is the increased migration of health workers throughout the years [4].

Despite the shortage of health workers, the DOH still attempts to addresses the problem of distribution with programs like Doctors to the Barrios (DttB). The DttB program deploys physicians to underdeserved and difficult-to-access municipalities for at least two years [5]. The University of the Philippines College of Medicine (UPCM) has also looked into mandatory return of service programs as a way to bring more doctors to where they are needed the most [6]. While these programs may alleviate the problem of shortage and distribution of personnel, they do not involve the construction of more health centers to accommodate the communities. Rural physicians might not have all the facilities and resources that they may need during their deployment.

The University of the Philippines Manila - National Telehealth Center (NThC) performs research and development to improve healthcare delivery through the use of information and communication technologies (ICT) [7]. Their areas of focus include the provision of assistance to rural physicians through their telemedicine project, the National Telehealth Service Program (NTSP). This allows rural physicians access to vital support and consultation from clinical specialists which would otherwise be unavailable due to geographical barriers [8].

The NThC utilizes both e-mail and Short Message Service (SMS) for the NTSP [9]. E-mail has proven that it is a reliable messaging system by still being in widespread use for both business and personal transactions over the internet. However, SMS is more prevalent in the Philippines with its widespread use earning the country the title of SMS capital of the world [10]. This can be attributed to the low cost of sending SMS at 1 Philippine peso (roughly $0.02) per message. Devices that are internet-ready are also more expensive than those that are SMS-capable only. This makes SMS very accessible across all economic classes.

The NTSP works by allowing rural physicians to send consultations to its system via e-mail or SMS [11]. The system provides an interface for telehealth nurses, who are essential to the workflow. When a physician sends a message, a nurse receives and acknowledges it by replying to the referring physician. The messages content is then verified by the nurse. On verification, the case is considered opened and filed under a certain category. The message is then forwarded by the nurse to the appropriate clinical specialist or domain expert. The domain expert sends a response to the system, which is again received by the nurse. The response is forwarded back to the referring physician through the system [12]. Without the telehealth nurses, the system will not be able to accommodate the rural physicians.

Advances in health and technology are not exclusive to conventional ICTs. Natural language processing (NLP) deals with computer systems processing, ultimately extracting meaning from sentences in natural languages. NLP encompasses a wide range of sub-problems, some of which find applications in the clinical field [13]. One application of NLP is that of text classification, which can find use in electronic health records, among others.

The NTSP is continuously being developed, with the end goal of eventually expanding into a nationwide system with regional divisions. This study explores the applications of natural language processing in streamlining the telemedicine process for efficiency, and addresses existing problems that the system has encountered.

B. Statement of the Problem

Faster decision-making for rural physicians would lead to better health outcomes. To be able to assist rural physicians in attaining that goal, telehealth nurses must be able to perform their tasks efficiently and effectively. Therefore, speed and accuracy should be prioritized in the telemedicine process. Obstacles in these areas should be eliminated.

The ease of use of e-mails makes it easy to abuse. E-mail spam refers to abusive unsolicited bulk e-mails. Occasionally, they may also pose as security threats, as they may contain attachments or links to malware. It is estimated that more than half of all e-mail traffic is spam [14]. SMS spam also exists, albeit in smaller quantities. However, the smaller amount of SMS spam and the shortness of SMS makes it difficult to apply existing e-mail spam filtering techniques [15].

Due to the nature of the NTSP system and its reliance on e-mail and SMS, it is prone to spam in both platforms. While the occasional spam message may seem harmless, consistent spam may hinder the speed of the telemedicine process by taking up valuable time of the telehealth nurse assigned to the system. With the expansion and deployment of the system to more areas, the volume of incoming messages is expected to increase. Knowing this, the telehealth nurse should maximize efforts solely on the relevant messages and not be bothered by the spam messages.

Currently, the system requires referring physicians to label their messages accordingly with keywords, to help the telehealth nurse in classifying the request and forwarding to the appropriate domain expert. For e-mails, this is placed in the subject field, while SMS follows a certain format. At this point, the remaining task of the telehealth nurse is accurate verification of the messages. However, there are times that messages are mislabeled or not labeled at all, adding to the telehealth nurse's duties. Additionally, the development of the NTSP aims to eventually do away completely with the labeling requirement. This is to make the system flexible on the part of the rural physicians. To ensure accuracy, there should be a way to aid the telehealth nurse in the classification of messages.

C. Objectives of the Study

The study has developed a web-based system with the following components: a general spam blocker, an automated message tagger, and an interface for receiving, verifying, and managing messages.

The system includes the following functionalities:

1. Allow a telehealth nurse to:

    (a) Log-in to the system

    (b) View the messages
    (c) Manually mark a message as spam
    (d) Manually change a message tag
    (e) Manually remove a message tag
    (f) Verify that a message is ready to be forwarded
    (g) Remove verified status from a message
    (h) View the filtered messages
    (i) Unmark a message as spam

2. Allow an administrator to:
    (a) Function as a telehealth nurse
    (b) Configure the spam blocker
        i. Add an entry to the block list
        ii. Remove an entry from the block list
    (c) Configure the automated message tagger
        i. Train the message tagger using support vector machines
            A. Specify n-grams to be used in feature extraction
            B. Specify percentile of best features to be selected
        ii. Test the message tagger using support vector machines
        iii. Save the trained model
        iv. Load a previously saved model
        v. Choose a model that will be used by the message tagger

D. Significance of the Project

The development of the system addresses the previously discussed problems by making use of natural language processing techniques and machine learning methods in automated message tagging.

The creation of a rudimentary spam block list for both e-mails and SMS messages places a protective measure where the system previously had none. This ensures that spam entering the system is kept from zero to a bare minimum. This removes the time and effort that a telehealth nurse would otherwise be devoting to filtering spam messages.

The creation of an automatic message tagger eliminates the need for rural physicians to label their messages while still aiding the telehealth nurse by maintaining their task to that of verification. The automatic classification may also suggest a tag that the telehealth nurse might not have possibly considered.

However, the true strength of the system lies in its ability to learn through training and re-training. This makes sure that the system is flexible enough to be able to adapt to new tags, and improve performance over time.

E. Scope and Limitations

The following are the scope and limitations of the system:

1. The system simulates the arrival of messages through an input form.
2. The system can only be used to process messages in SMS and e-mail form.
3. The system does not handle the redirection or forwarding of messages.
4. Only messages in English, Filipino, and Taglish will be processed.

F. Assumptions

The following are the assumptions of the system:

1. The system has existing data that the message classifier can train on.

2. Incoming messages lack keywords indicating their correct tag.

3. The telehealth nurse is knowledgeable enough to handle the verification of tags.

4. The administrator is knowledgeable enough to be able to properly configure the spam blocker and the message tagger.

II. Review of Related Literature

Telemedicine is broadly defined by its literal meaning: healthcare at a distance. Its loose terminology has resulted in it being referred to as telehealth, online health, e-health, connected health, mobile health, and so on [16]. Regardless of the terms used, the idea stays the same. Information and communication technologies (ICT) are utilized to provide or support the delivery of healthcare remotely or at a distance. It is used in a very wide sense, from practical functions such as diagnosis and management of patients, to educational and administrative functions such as data analysis and videoconferencing.

The significant growth in the use of ICT around the world has been greatly influenced by similarly significant advances in technology. As ICT is integrated into everyday life, the awareness of their potential as social and economic tools for the public good grows [17]. This pattern is better seen in mobile technologies. The growth of mobile network coverage followed by increased use of mobile devices in developing countries has spawned respective initiatives into telemedicine [18].

In the Philippines, there has been a number of telemedicine projects implemented by both government and non-government organizations. The University of the Philippines Manila National Telehealth Center has produced, among others, the Community Health Information Tracking System (CHITS), the E-Learning for Health Project, and the SMS Telemedicine Project [19].

The use of Short Messaging Service (SMS) in the telemedicine perspective seems to have gained increased attention. Nisperos and Urano developed an SMS-based tool that allows consumers to receive information on medicine prices, allowing them to make informed decisions on drug purchases [20]. SMS has also been used to manage Diabetes Mellitus (DM), by sending text messages to patients about diet, exercise, and non-adherence to DM management. Patients were shown to have improved their adherence to diet and exercise over the course of 6 months [21]. In Rwanda, an SMS-based alert system was able to improve the response of community health workers to pregnancy cases, with the workers reported to be more pro-active [18].

A comprehensive review of studies on SMS for disease control in African countries noted that there was a lack of peer-reviewed literature that measured clinical outcomes. However, the studies reported that SMS intervention for disease prevention, surveillance, management and patient compliance were well accepted by the general population [22].

SMS-based systems are also built with the intention of collecting data for analysis and public health actions. U-report, an open-source SMS platform operated by UNICEF Uganda, gives the community a voice by conducting polls on issues, and receives both poll responses and unsolicited reports for other problems in the community [23]. With 200,000 participants and 10,000 text messages a week, manual inspection of messages is no longer sustainable. To be able to process messages that need response or action, UNICEF Uganda worked with IBM to develop an automated text classification system. At first, texts are filtered to identify poll responses. Unsolicited text messages are then classified. Initial attempts incorporated keyword matching, which worked for a few categories but was found to be insufficient overall. Machine learning techniques, specifically Nave Bayes Classifiers and Support Vector Machines, were used instead and with greatly improved results. This is an example of a single label classification problem.

In machine learning (ML), the goal of single label classification problems is to learn from a set of instances, the unique class label of an instance from a set of disjoint class labels. Depending on the number of disjoint class labels, the problem can either be binary classification (when there are only 2 labels) and multi-class classification (when there are more than 2 labels). The problem becomes a multi-label classification one, when instances are allowed to have more than one class [24].

Another NLP problem solved with a non-ML solution is that of the Patient-centered Automated SMS Tagging Engine (PASTE) [25]. PASTE uses NLP methods, lexicons, and knowledge sources to extract and tag medication information from patient text messages. Lexicons were built with textual ambiguity and SMS-specific abbreviations in mind. Tagging of terms was done by simple dictionary lookup.

One of the most popular approaches of solving multi-label problems is by implementing binary relevance (BR) [24]. In BR, the classification problem is treated as multiple binary classifiers, where the final result is a union of the result of those binary classifiers. While BR is easy to implement, it is not without disadvantages. There are proposed algorithms that improve upon the performance of BR by using correlation-based pruning [26], as well as to address cases where the classification has a large amount of labels [27].

There are alternative techniques in solving multi-label problems. One such approach makes use of an ensemble of Bayesian Networks, in contrast to the popular BR method and Chain Classifiers using Nave Bayes [28].

The MITRE system for clinical assertion status classification tackles a multiclass problem. It is different from the previous solutions in that it combines rule-based (pattern matching) techniques with ML (conditional random fields) techniques. The best submission obtained an overall micro-averaged F-score of 0.9343 [29].

Kim et al. showed the potential of conditional random fields (CRF) for learning over sequential data such as cohesive, structured text, or sentences. Using a set of pre-defined medical categories, sentences in medical abstracts are processed and then annotated. Their system performed very well, with micro averaged f-scores of 80.9% and 66.9% for structured and unstructured abstracts respectively [30]. A later improvement makes use of sentence ordering labels for unstructured abstracts. It was able to score as much as 22% higher than previous state-of-the-art systems[31].

Spat et al. developed a multi-label classification system for German-language clinical documents. The system was tested on 1,500 clinical documents, with and without text preprocessing, and on four different classification models. The training sample size for both cases was only limited to 150.F1-measure of all the classification results did not go below 0.8. Those with preprocessing applied performed slightly better than those without preprocessing [32]. The results show that classification models can be very effective when they are applied to a field with very specific terms used.

A generic strategy in text classification involves pre-processing, feature extraction and selection, model selection and classifier evaluation [33]. Pre-processing consists mostly of NLP tasks. Stop-words are words that are frequently occuring words in a text that are not useful for classification. Removing them reduces the vocabulary of the training text. Feature extraction and selection helps identify the important words in a text. This may include significant words, multi-words, or frequently occuring phrases.

Uysal and Gunal confirms that choosing appropriate pre-processing tasks significantly improve classification accuracy [34]. Their experiments show that appropriate combinations of pre-processing tasks provide signifcant improvements than just enabling or disabling all tasks. Performance will also depend largely on the domain and the language of the text.

A common issue in text classification is the high dimensionality of the feature space. This is addressed by feature selection. Experiments of Meesad et al. suggests that the determination of word importance using chi-square tests increases the speed of the classifier and significantly reduces resource usage while still preserving classifier performance [35]. Chen and Chen demonstrates how chi-square statistics improve a classifier by measuring similarities [36].

Support vector machines (SVM) tend to give good classification results with ample training data. SVM was used in classifying whether a document was a satirical or true news document [37]. In the study, the use of term frequency-inverse document frequency in feature extraction maximized the accuracy.

A study classifying Twitter trending topics into 18 classes achieved a classification accuracy of up to 65% using support vector machines [38]. In the same study, it was found that classification accuracy is a function of the number of teets and the frequent terms in them.

A binary Naive Bayes classifier was used to categorize messages in an online health message board [39]. Using unigrams and the chi-square-stat, it was able to achieve an F-score of 0.54, in contrast to the F-score of 0.4 without feature selection.

Argaw et al. explored the use of SVMs in categorizing medical free-texts and achieved an F-score of 0.83 [40]. The high score was reached by using unigrams and applying term frequency-inverse document frequency weights. In addition to that, a medical thesaurus was used as a knowledge base for stop-word removal. Using knowledge bases in the feature extraction and selection stages generally improves classifier performance as shown by Hassan [41]. The Naive Bayes classifier showed greater improvement with the knowledge base than the SVM classifier.

SVM has also been used in named entity recognition tasks. Zhenfei utilized SVM to recognize the name of the specified type from the collection of biomedical texts, getting a precision or 0.84 and recall of 0.80 [42]. It was also found that there is a relatively low recall rate with multiple categories. Increasing the test data increases the recall rate [43].

Combinations of unigrams and bigrams, and Support Vector Machines and Bayesian networs were used in a system designed to process clinical narratives and identify the patient's smoking status [44]. The task was divided into two binary classification tasks: first, whether the patient is a smoker and a non-smoker. Second, whether a patient is a current smoker or a past smoker. SVM together with bigram representation performed better in both tasks.

A comparison of semantic and statistical methods applied in medical entity recognition showed that a hybrid of both types of methods yielded the best performance [45]. Specifically, the methods involved the use of conditional random fields using features generated from semantic and rule-based methods.

III. Theoretical Framework

A. National Telehealth Center

The University of the Philippines Manila National Telehealth Center (NThC) was established in 1998 and is the leading research unit in the University of the Philippines with focus on making innovations in ICT to improve health care [46]. In line with its vision, the NThC is commited to engaging people to use available technologies in cost-effective ways to improve health care despite geographical barriers [7].

Some of its projects are: the RxBox, a multi-component device providing access to life-saving health care services to isolated and disadvantaged communities; and the National Telehealth Service Program, which allows for consutations between primary care physicians in geographically isolated and disadvantaged areas and specialists of the Philippine General Hospital using mobile and internet-based technologies [47].

B. Telemedicine

The American Telemedicine Association defines Telemedicine as "the use of medical information exchanged from one site to another via electronic communications to improve a patient's clinical health status" [48]. The definition is very broad, and can include many applications of information and communication technology (ICT) for facilitating healthcare from a distance.

Telemedicine can be grouped under two categories: asynchronous or store and forward, and synchronous or real-time. An example of asynchronous telemedicine is when data on a patient is gathered and then sent to another health professional for diagnosis. Synchronous telemedicine on the other hand involves live interaction such as videoconferencing for a visual consultation [49]. There is potential in implementing telemedicine in developing countries especially when in rural communities where people have less access to quallity healthcare. This applies to the Philippines, where geographical barriers present an opportunity for telemedicine to be utilized.

C. Doctors to the Barrios

The Department of Health (DOH) launched the Doctors to the Barrios (DttB) Program in 1993 to alleviate the lack of physicians serving in remote areas in the Philippines [5]. Under the program, volunteer physicians are deployed to depressed, unserved or underserved municipalities for a minimum of 2 years. Geographically hard to reach and critical 5th and 6th class (low-income) municipalities without doctors are included among the prioritized areas of deployment. The program also allocates volunteers to 3rd and 4th class municipalities that need more doctors to achieve the optimal doctor to population ratio of 1 to 2,000 [50].

In 2014, the program was reviewed with the aim of addressing issues and initiating reforms [51]. Among these issues are overdependence of local government units (LGU), recruitment, retention and eventual absorption of volunteer physicians. Leonardia et al. notes how support from both the DOH and LGU in terms of utilities, infrastructure and logistics is a crucial factor for physicians to leave or remain in their assignments. In their assessment, only 57.7% and 56.4% of DttBs are satisfied with the support that they receive from the municipal government and the DOH Central Office, respectively. Only 36% are satisfied with the quality of care that their health center can provide. Additionally, only 29.5% claim that their health unit has access to resources for health programs and projects, while a meager 18.3% have the equipment to do their job well and efficiently [5].

It is suggested that to enhance physician retention within the program, both the DOH and LGU must strengthen the support being given to DttBs. Physicians applying for the program should be motivated in helping rural populations, or have an interest in public health. Modern communication technology should also be provided to the health centers [5]. This will result in benefits not only for the rural physician, but also for the community being served.

D. Referrals

In the context of medicine, a referral occurs when a physician turns over the care of a patient to an appropriate specialist. For example, a general practitioner may redirect a patient to an ophthalmologist if deemed appropriate and necessary.

Under the NThC's National Telehealth Service Program, a referral can either be a teleconsultation, or a telereferral. Both terms are in the same vein as telemedicine, in that they are consultations and referrals made from a distance through the use of ICT. The NTSP assists rural health physicians or DttBs in making referrals by providing a system that will ensure that the physicians are connected to the appropriate domain experts (DE).

Fernandez and Gomez describes the typical referral workflow [12] as follows:

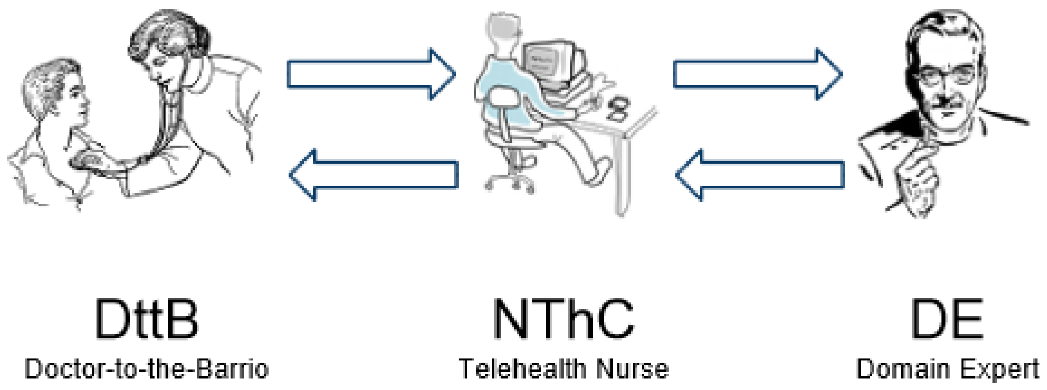


Figure 1: Referral workflow

1. Dttb sends referral to NThC.

2. NThC receives referral from DttB.

3. NThC forwards referral to the appropriate DE.

4. DE receives referral from NThC.
5. DE sends reply to NThC.
6. NThC receives reply from DE.
7. NThC forwards reply to DttB.
8. DttB receives reply from NThC.

However, the process is interrupted and prone to failure when the following occurs:

- Multiple DttBs are referring at the same time. This slows down the service.
- NThC personnel is not available. Forwarding is not facilitated at all.
- Dttb attempts to directly contact a DE. There is a possibility of contacting the inappropriate DE.

E. Machine Learning

Machine learning is a branch of computer science that is concerned with methods and algorithms to learn from and make predictions on data. A common definition states: a machine learns from experience E with respect to some class of tasks T and performance measure P, if its perforamnce at tasks in T, as measured by P, improves with experience E [52].

There are generally three types of learning differentiated by the available feedback [53].

- Supervised Learning - the machine is given an example input paired with the desired output, and will generalize rules in mapping inputs to outputs.
- Unsupervised Learning - the machine is given no hints and must process unlabelled examples to discover stuctures in the input.

- Reinforcement Learning - the machine learns through interactions with a dynamic enviornment, receiving an evaluation of its actions without being given the correct ones, all while maximizing the accumulated reward over time.

In a classification task where labeled data is available, it is only appropriate to use a supervised learning algorithm. Here, a model is trained based on the labeled data containing paired inputs and outputs. The model will then be used on new data composed of inputs and predict the respective outputs.

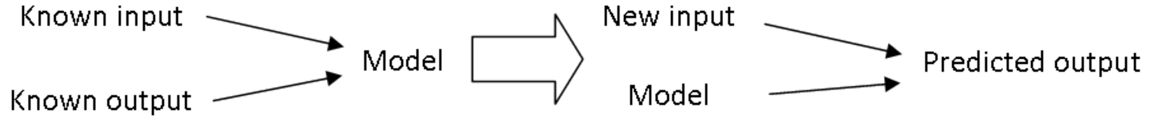


Figure 2: Supervised learning process

F. Natural Language Processing

Natural language processing (NLP) has its roots in the 1950s and from the fields of artificial intelligence and linguistics. The end goal of NLP is to be able to extract meaning, or semantics, from plain text. NLP itself is a broad field consisting of numerous tasks. Some low-level NLP tasks include sentence boundary detection, tokenization, part-of-speech tagging and shallow parsing. Higher level tasks tackle more specific problems built on the low-level tasks. Some of these tasks are named entity recognition, negation identification and information extraction [13].

Machine learning and NLP are related in the sense that they reside under the same problem space as the broader field of artificial intelligence. NLP deals with the problem of understanding language while ML focuses on learning through experience. Combining the two results into machine learning techniques being applied to solve NLP problems.

G. N-gram

An n-gram is a sequence of n items taken from a longer sequence. In the context of natural language processing, items are usually either characters or words. For example, in the phrase, “hoping to hear from you soon”, the extracted n-grams as words are:

- n=1 (unigram): “hoping”, “to”, “hear”, “from”, “you”, “soon”
- n=2 (bigram): “hoping to”, “to hear”, “hear from”, “from you”, “you soon”
- n=3 (trigram): “hoping to hear”, “to hear from”, “hear from you”, “from you soon”
- and so on for n

A common application of n-grams is in building language models and utilizing their frequencies to solve classification problems [54].

H. Term Frequency-Inverse Document Frequency (TF-IDF)

In natural language text, some words appear very frequently while not carrying a lot of useful information with regard to the content of the text. In English, some of these words include “the”, “a”, and “is”. A classifier dependent on term frequencies would give more weight to these words than the less frequent, but more interesting terms. TF-IDF is a weight that reflects how imporant a term is to a document in a collection by considering term frequency in both the document and the whole collection [55].

TF-IDF is composed of two terms, Term Frequency (TF) and Inverse Document Frequency (IDF).

$$TF(t) = \frac{\text{Number of times term } t \text{ appears in a document}}{\text{Total number of terms in the document}}$$

$$IDF(t) = \log \frac{\text{Total number of documents}}{\text{Number of documents with term } t \text{ in it}}$$

The TF-IDF weight is the product of TF(t) and IDF(t).

I. Chi Square (χ^2) Feature Selection

Feature selection is the process of selecting a subset of features and using only this subset in the construction of a machine learning model. In text classification, feature selection is mainly used to increase a classifier's efficiency by decreasing the size of the effective vocabulary. It is also used to improve classification accuracy by eliminating noise features, or features that increase classification error on new data [56].

A common feature selection method is using the Chi Square (χ^2) test. In statistics, the χ^2 test is used to test the independence of two events. In feature selection, we use it to test whether the occurence of a specific term and the occurence of a specific class are independent [57]. Features that are found to be most likely independent of class are considered irrelevant for classification.

J. Support Vector Machine (SVM)

A Support Vector Machine is a supervised learning model commonly used for classification and regression. A popular rationale for using SVMs is its promising empirical performance [58].

In a classification problem, an SVM represents the data as points in space. The SVM then finds the optimal hyperplane that separates the data into their respective classes. The optimal separating heyperplane maximizes the margin between the data. Instances of a class that are the most difficult to classify due to similarity are called support vectors. Consequently, support vectors are the data points closest to the separating hyperplane. If a support vector is moved, the separating hyperplane is changed accordingly.

A binary (two-class) classification problem is shown in Figure 3. The labeled data are represented as squares and circles. The support vectors are the points nearest to the other class (they lie on the broken lines.) New unlabeled data will be labeled accordingly on what side of the optimal hyperplane they fall in. This assumes that the data are linearly separable.

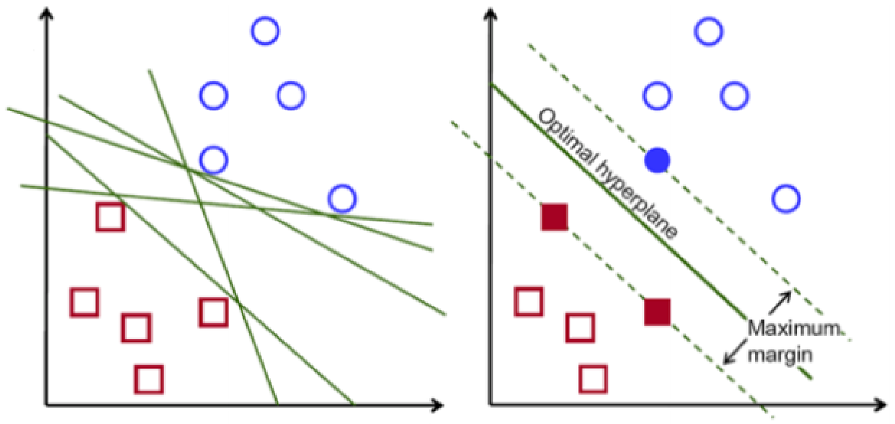


Figure 3: Support vector machine; generated optimal hyperplane

In the event that the data is not linearly separable, SVMs can still perform non-linear classification through the use of the kernel trick. This maps the data into a higher dimensional space, where a separating hyperplane can be generated. This is demonstrated in Figure 4

If the number of features is large, the data may not be needed to be mapped to a higher dimensional space. In this case, the nonlinear mapping does not improve the performance, making the linear kernel suitable for the task [59].

SVMs can be applied in a multi-class classification problem by combining multiple binary classifiers in a "one versus all" scheme. For each k class, a binary classifier is learned against the other k-1 class. The class with the highest score is then chosen.

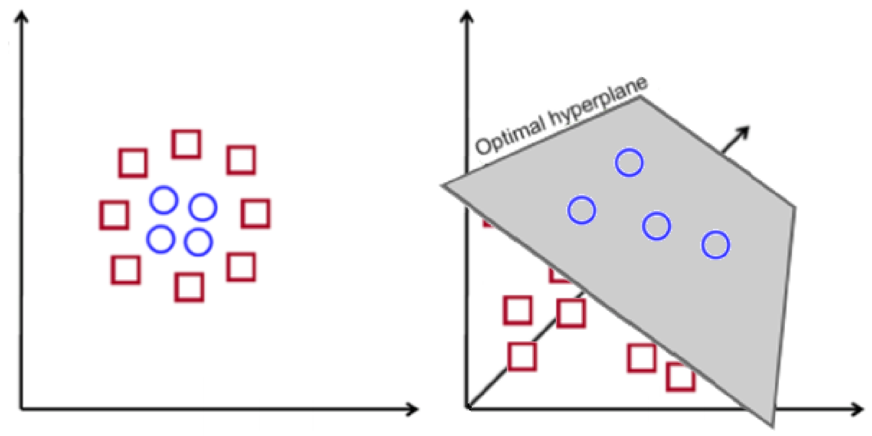


Figure 4: Support vector machine with kernel trick

K. web2py

web2py is a free and open source web framework for rapid development of fast, scalable, secure and portable database-driven web-based applications. It is written and programmable in Python. web2py is a full-stack framework, which means that it contains all the components needed to build functional web applications (web server, SQL database, integrated development environment) out of the box. It follows the Model View Controller (MVC) pattern[60].

L. scikit-learn

scikit-learn is a free machine learning library for Python. It integrates a wide range of state-of-the-art machine learning algorithms for supervised and unsupervised problems. It focuses on ease of use, performance, documentation and API consistency. Being written in a general-purpose high-level language makes it accessible even to non-specialists in the field of machine learning [61].

IV. Design and Implementation

A. Use Case Diagram

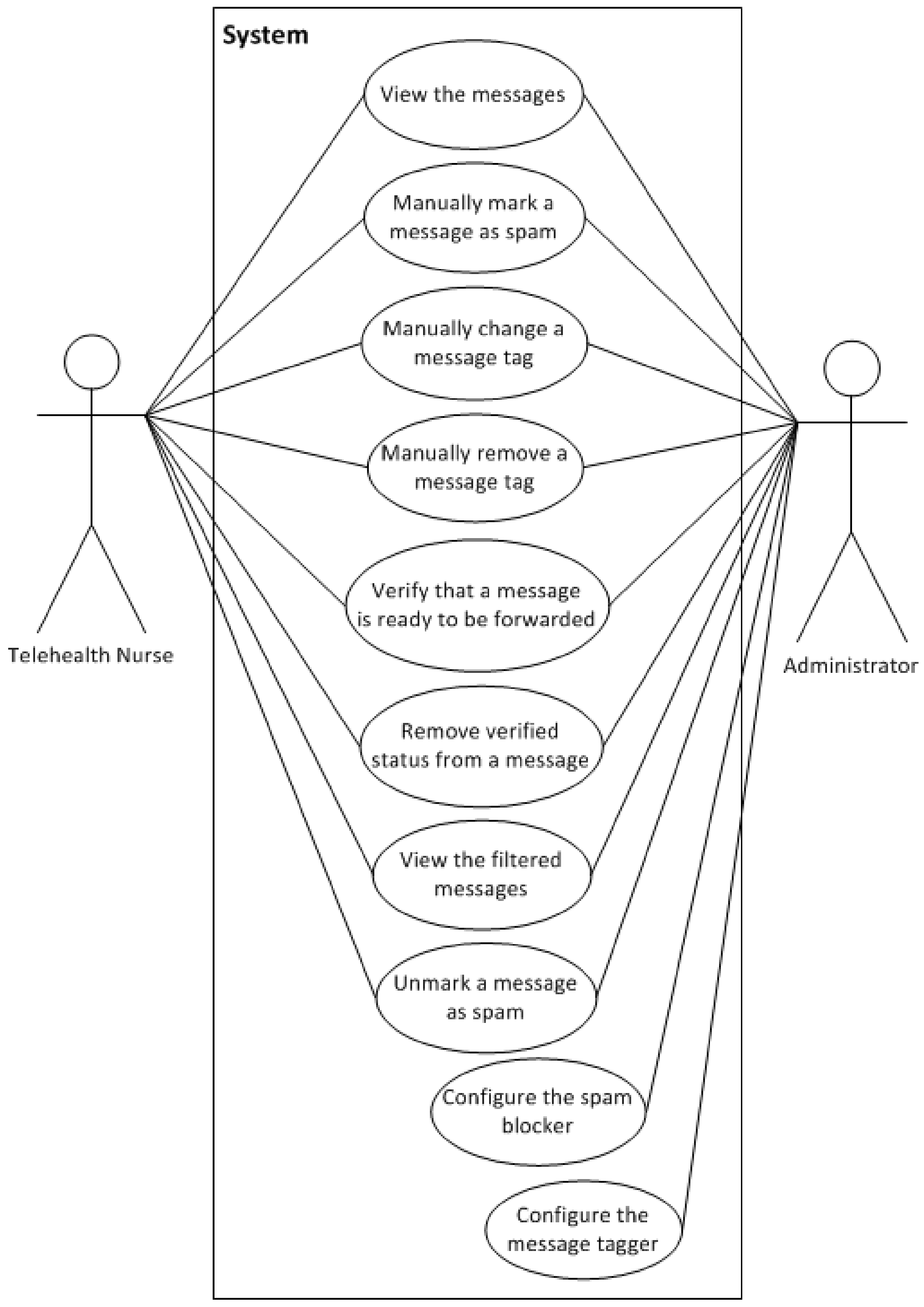

System
View the messages
Manually mark a message as spam
Manually change a message tag
Manually remove a message tag
Verify that a message is ready to be forwarded
Remove verified status from a message
View the filtered messages
Unmark a message as spam
Configure the spam blocker
Configure the message tagger
Telehealth Nurse
Administrator

Figure 5: Overview use case diagram

There will be two types of users: the telehealth nurse, and the administrator. The system will automatically tag any messages received.

The telehealth nurse performs the manual corrections to individual messages. These include changing tags, verifying that a message is ready to be forwarded, removing the verification status, marking and unmarking spam.

The administrator can perform the same functions as the telehealth nurse. In addition to that, the administrator can configure the blocker and the tagger.

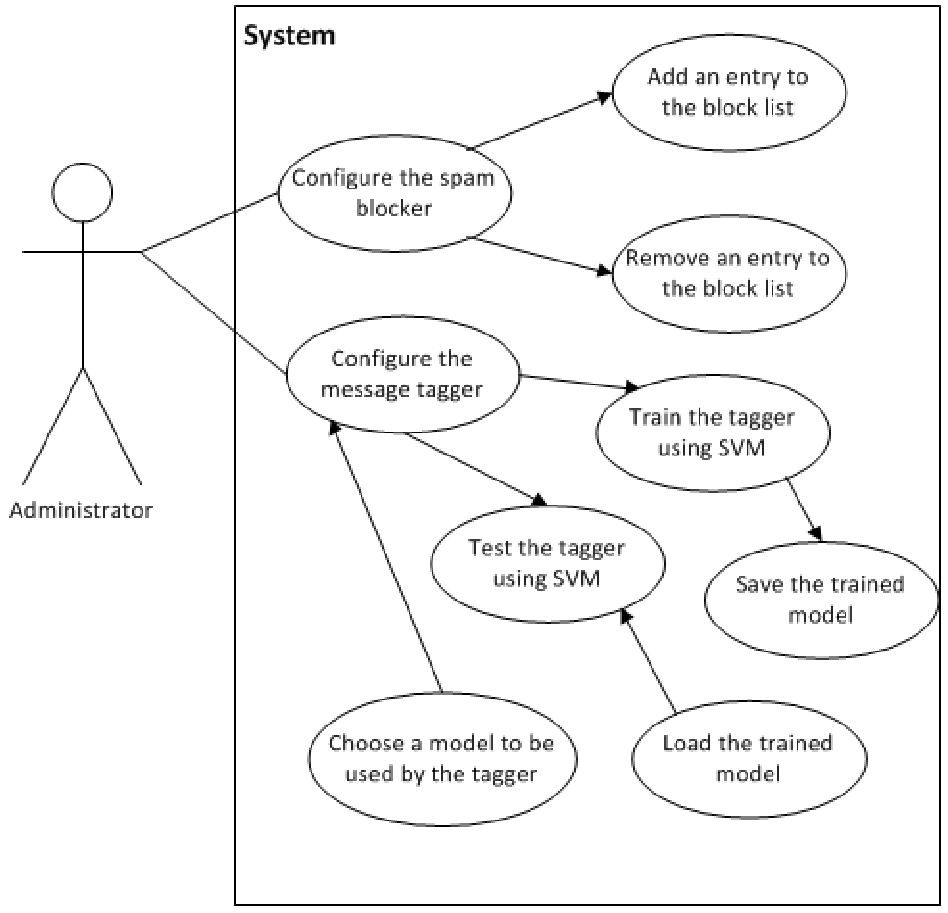

System
Add an entry to the block list
Configure the spam blocker
Remove an entry to the block list
Configure the message tagger
Train the tagger using SVM
Test the tagger using SVM
Save the trained model
Choose a model to be used by the tagger
Load the trained model
Administrator

Figure 6: Administrator use case diagram

Configuring the spam blocker consists of adding and removing entries to the block list. Configuring the message tagger consists of training and testing the SVM. It is also possible to save and load trained models. Lastly, the administrator can choose which model to use in deployment.

B. Data Flow Diagram

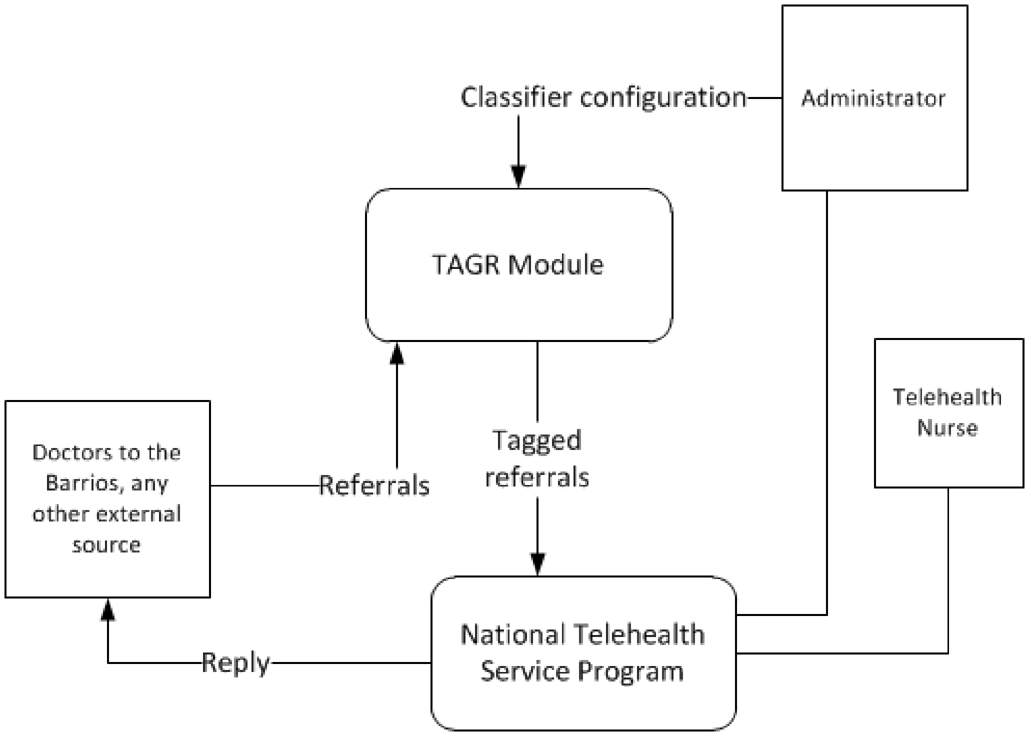
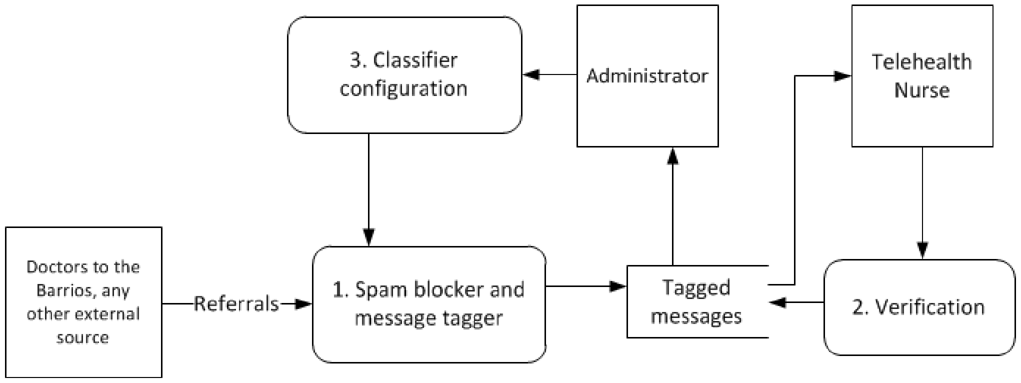


Figure 7: Top level data flow diagram

Messages, in the form of e-mail or SMS referrals, will be coming from outside the system. The messages will be automatically processed by the system, and then manually verified by the telehealth nurses. Administrators shall oversee the configuration of the classifier used by the system.



3. Classifier configuration
Administrator
Telehealth Nurse
Doctors to the Barrios, any other external source
Referrals
1. Spam blocker and message tagger
Tagged messages
2. Verification

Figure 8: Explosion of data flow diagram

Figure 9 exposes the three main functions of the system: automated tagging, manual verification, and classifier configuration.

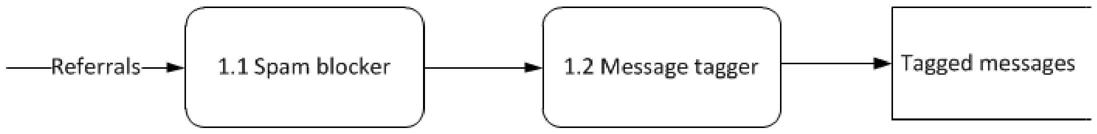


Figure 9: Sub-explosion for 1. Spam blocker and message tagger

Messages will first go through the spam blocker before anything else. This process is detailed in Figure 10. If a message is determined to be spam, it is filtered out but still saved in the database.

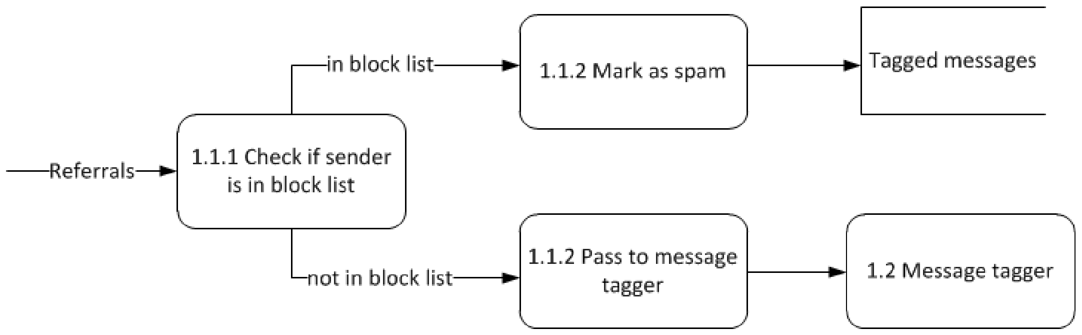


Figure 10: Sub-explosion for 1.1 Spam blocker

If the message is not filtered out, it is then passed to the message tagger. Figure 11 shows the step-by-step process of classification. The tokenization, vectorization, and TF-IDF transformation is implemented as one function in TAGR.

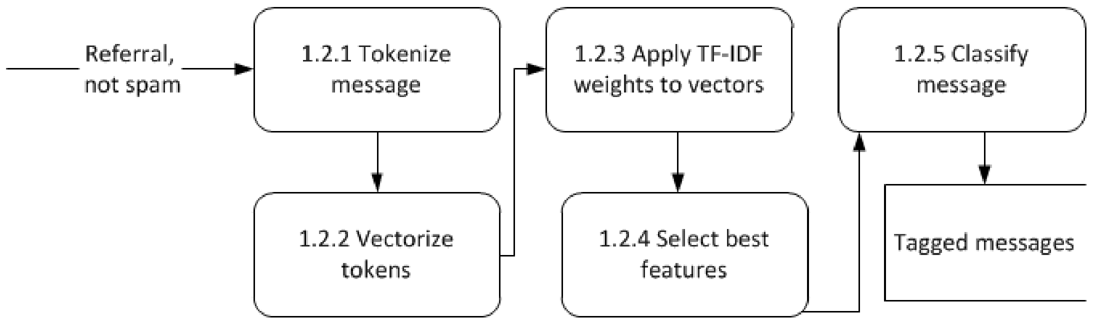


Figure 11: Sub-explosion for 1.2 Message tagger

The next three figures 12, 13, 14 demonstrate the manual verification process that can be done by both the telehealth nurse and the administrator. Note that only messages that are not marked as spam can be tagged and verified. Consequently, only unverified messages can be marked as spam.

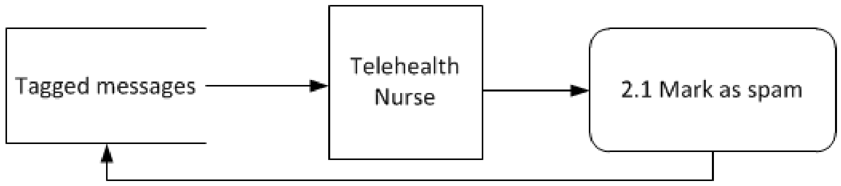


Figure 12: Sub-explosion for 2. Verification, Mark as spam

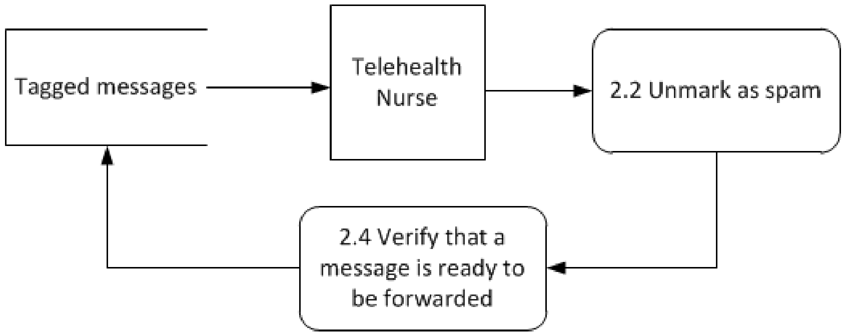


Figure 13: Sub-explosion for 2. Verification, Unmark as spam

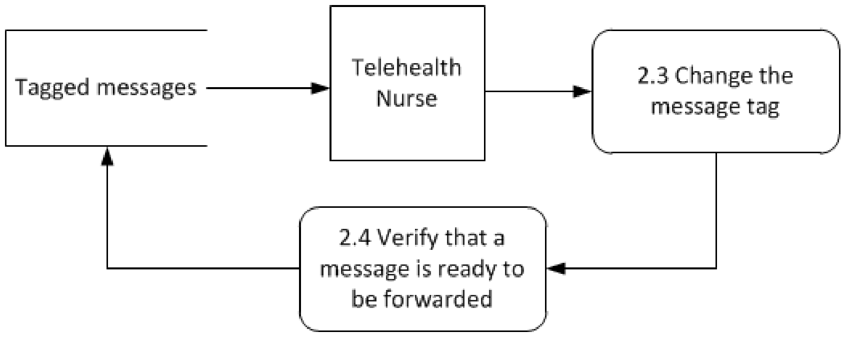

Tagged messages
Telehealth Nurse
2.3 Change the message tag
2.4 Verify that a message is ready to be forwarded

Figure 14: Sub-explosion for 2. Verification, Change message tag

Classifier configuration is split into configuration for both the blocker and the tagger as shown in Figure 15.

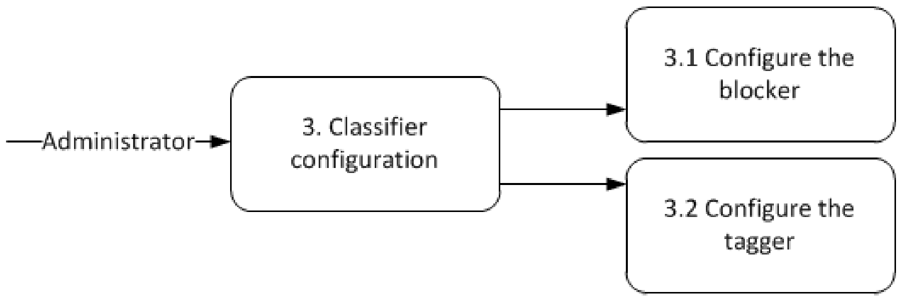


Figure 15: Sub-explosion for 3. Classifier configuration

Configuring the blocker is as simple as adding and removing entries from the block list.

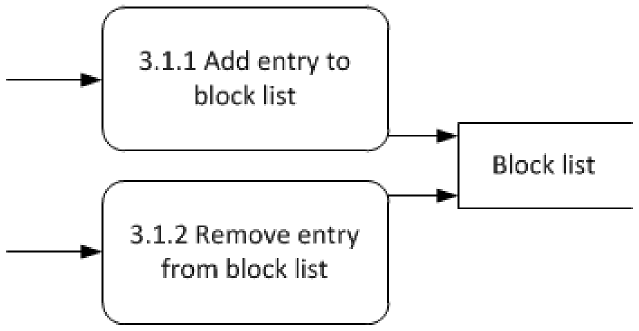

3.1.1 Add entry to block list
3.1.2 Remove entry from block list
Block list

Figure 16: Sub-explosion for 3.1 Configure the blocker

Training a new model involves the similar process of tokenization, vectorization, TF-IDF weighting, and feature selection of process 1.2 (Figure 13). The difference is that instead of applying it into a single message instance, the process is applied onto the existing database of messages.

After processing all of the messages from the database, cross validation is performed using an instance of the classifier. During cross validation (Figure 17), the processed messages are split into 10 folds, or sets. The validation is then done for 10 iterations. For each iteration, a fold is selected, while the classifier is trained on the remaining 9 folds. It is then tested on the selected fold. The collated results of the 10 iterations is shown as an evaluation report containing Precision, Recall, F-score, and a confusion matrix.

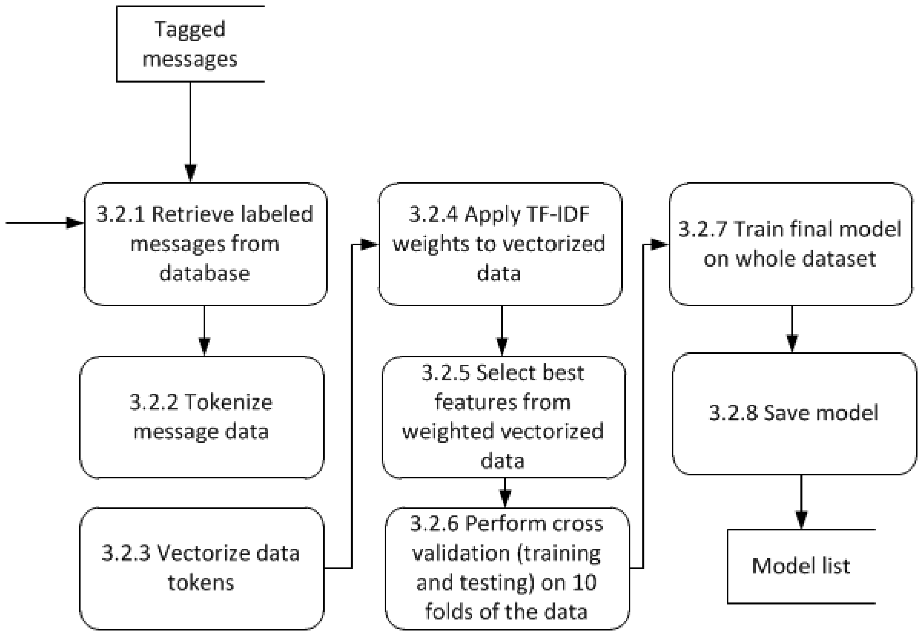

Tagged messages
3.2.1 Retrieve labeled messages from database
3.2.2 Tokenize message data
3.2.3 Vectorize data tokens
3.2.4 Apply TF-IDF weights to vectorized data
3.2.5 Select best features from weighted vectorized data
3.2.6 Perform cross validation (training and testing) on 10 folds of the data
3.2.7 Train final model on whole dataset
3.2.8 Save model
Model list

Figure 17: Sub-explosion for 3.2 Configure the tagger, Train a new model

If the evaluation is satisfactory to the administrator, the trained model can be saved to the database where it can then be loaded for later use. The loading process repeats most of the steps in the training of a new model.

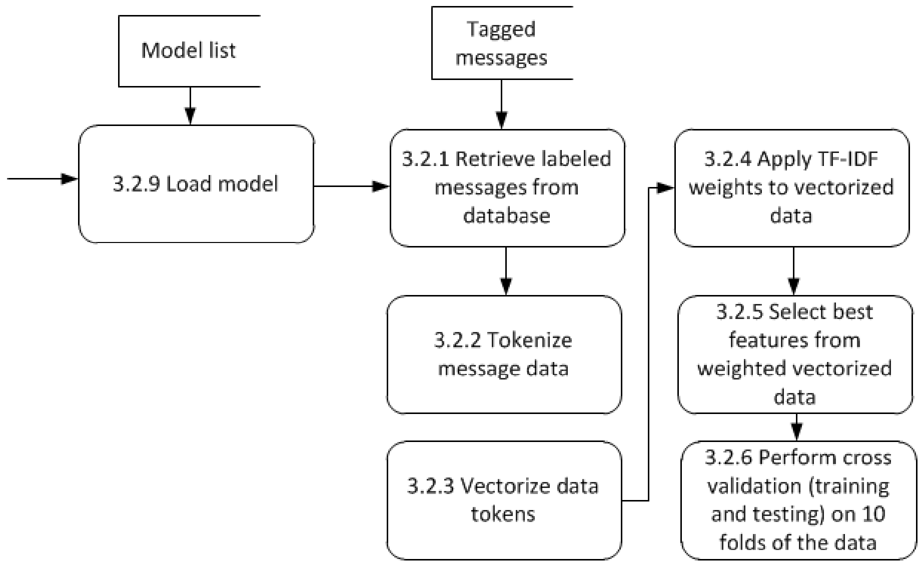

Model list
Tagged messages
3.2.9 Load model
3.2.1 Retrieve labeled messages from database
3.2.2 Tokenize message data
3.2.3 Vectorize data tokens
3.2.4 Apply TF-IDF weights to vectorized data
3.2.5 Select best features from weighted vectorized data
3.2.6 Perform cross validation (training and testing) on 10 folds of the data

Figure 18: Sub-explosion for 3.2 Configure the tagger, Load a model

Choosing a model to be used in deployment simply sets a boolean switch in the

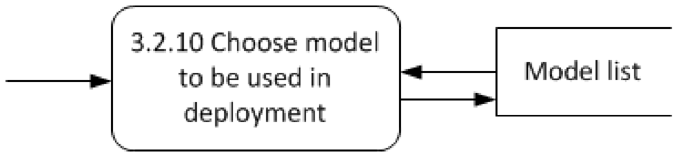


Figure 19: Sub-explosion for 3.2 Configure the tagger, Choose a model

list of models. A different instance of the classifier will be loaded with the chosen model.

C. Database Design

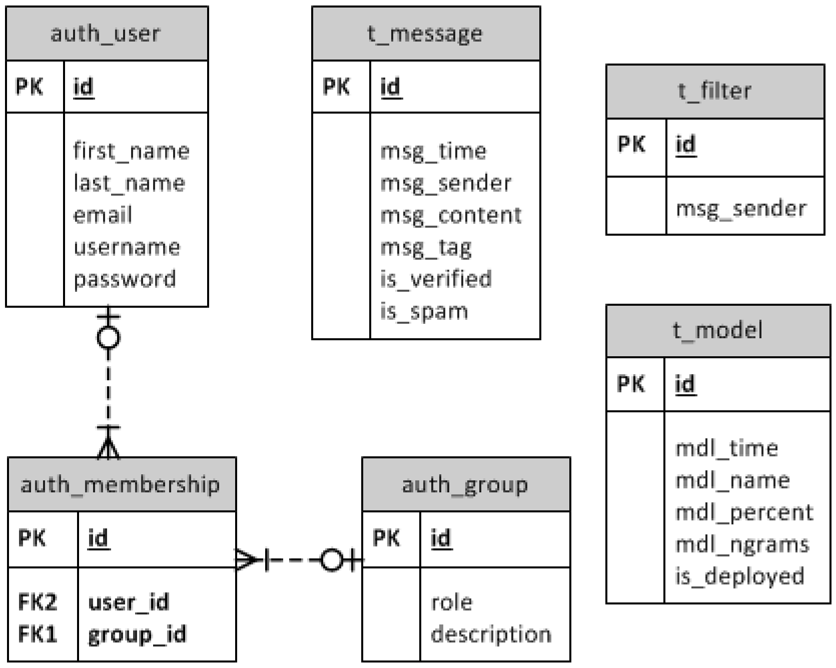


Figure 20: Entity relationship diagram

Figure 20 shows the entity relationship diagram of TAGR's web component. The *t_message* table stores the messages and its details. The *t_filter* table maintains a list of message senders with which new messages are checked to be filtered. The *t_model* table stores saved models and their respective configurations. Only one model can be set to be deployed at a time. This is enforced through controller logic.

D. Data Dictionary

Field	Type	Description
id	integer	Unique identifier of the user
first_name	string	First name of user
last_name	string	Last name of user
email	string	E-mail address of user
username	string	Username of user
password	string	Password of user

Table 1: auth_user table

Field	Type	Description
id	integer	Unique identifier of the user membership
user_id	integer	Unique identifier of the user in the user table
group_id	integer	Unique identifier of the group in the group table

Table 2: auth_membership table

Field	Type	Description
id	integer	Unique identifier of the group
role	string	Role assigned to the group
description	string	Description of the role

Table 3: auth_group table

Field	Type	Description
id	integer	Unique identifier of the message
msg_time	datetime	Date and time that the message was received
msg_sender	string	Contact of message sender
msg_content	text	Text content of the message
msg_tag	string	Assigned tag of the message; hardcoded list of possible tags: Unassigned, Dermatology, Family Medicine, General Question, Internal Medicine, Medico Legal, Neurology, Obgyn, Ophthalmology, Otorhinolaryngology, Pediatrics, Radiology, Surgery
is_verified	boolean	Indicates if a message is verified
is_spam	boolean	Indicates if a message is spam

Table 4: t_message table

Field	Type	Description
id	integer	Unique identifier of the filter
msg_sender	string	Contact of message sender to be filtered

Table 5: t_filter table

Field	Type	Description
id	integer	Unique identifier of the model
mdl_time	datetime	Date and time that the model was created
mdl_name	string	Name of the model
mdl_percent	integer	Selected feature percentile of the model
mdl_ngrams	integer	Extracted n-grams of the model
is_deployed	boolean	Indicates if a model is deployed

Table 6: t_model table

E. System Architecture

The system makes use of the web2py framework for its web-based component, and scikit-learn for the machine learning component. Both of this requires Python 2.7. While web2py does not require any other dependency, scikit-learn requires the installation of a number of additional Python modules. The system was developed with 32-bit Python 2.7. To ensure compatibility, care should be taken to install the exact 32-bit versions of all dependencies as they were during the development phase. They are:

- Python 2.7
- web2py 2.14.5 (source code version)
- numpy-1.10.2
- scipy-0.16.1
- scikit-learn-0.17.1

F. Technical Architecture

The server shall host the web-based interface of the system, as well as the machine learning components of the system. To accommodate this, the server must have the minimum requirements:

- Two gigabytes of memory
- Two-gigahertz processor
- Python 2.7 (32-bit), and dependencies listed above

The client only needs access to the web-based interface of the system. The client is recommended to have the following for smooth performance:

- Two gigabytes of memory
- Two-gigahertz processor
- Modern web browser

The system's server-side and client-side components were developed and tested on two machines. Their performance differences were negligible. Their specifications are as follows:

Role	Development, Testing	Testing
Machine	Custom-built desktop	Lenovo ThinkPad R400
Operating System	Windows 7 Professional SP1, 64-bit	Windows 7 Professional SP1, 32-bit
Processor	AMD Athlon II X2 240 2.8 GHz	Intel Core2 Duo P8400 2.26 GHz
RAM	4.00 GB	2.00 GB
GPU	NVIDIA GeForce 9600 GT 512 MB	Intel Integrated Graphics 4500MHD
Display Resolution	1600 x 900	1280 x 800

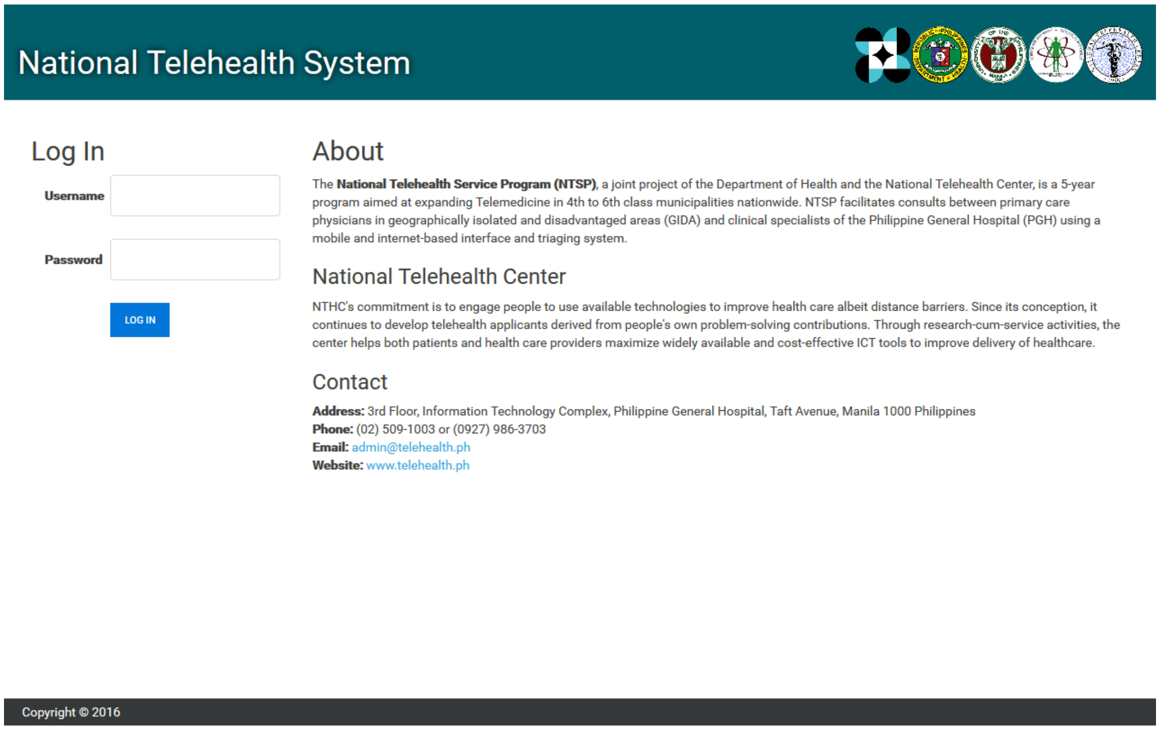
Table 7: Specifications of the machines used in development and testing

V. Results

Telehealth Automatically Generated Recommendations (TAGR) is a module for automatically classifying SMS and e-mail messages. Built around it is a web-based system that demonstrates how TAGR can be used. For the purpose of discussion, the web-based component, the National Telehealth System, shall be addressed as part of TAGR itself. The web-based component was built on the web2py framework while the message tagger utilized scikit-learn library.

A. General View

Login Page



National Telehealth System

Log In

Username

Password

LOG IN

About

The **National Telehealth Service Program (NTSP)**, a joint project of the Department of Health and the National Telehealth Center, is a 5-year program aimed at expanding Telemedicine in 4th to 6th class municipalities nationwide. NTSP facilitates consults between primary care physicians in geographically isolated and disadvantaged areas (GIDA) and clinical specialists of the Philippine General Hospital (PGH) using a mobile and internet-based interface and triaging system.

National Telehealth Center

NTHC's commitment is to engage people to use available technologies to improve health care albeit distance barriers. Since its conception, it continues to develop telehealth applicants derived from people's own problem-solving contributions. Through research-cum-service activities, the center helps both patients and health care providers maximize widely available and cost-effective ICT tools to improve delivery of healthcare.

Contact

Address: 3rd Floor, Information Technology Complex, Philippine General Hospital, Taft Avenue, Manila 1000 Philippines
Phone: (02) 509-1003 or (0927) 986-3703
Email: admin@telehealth.ph
Website: www.telehealth.ph

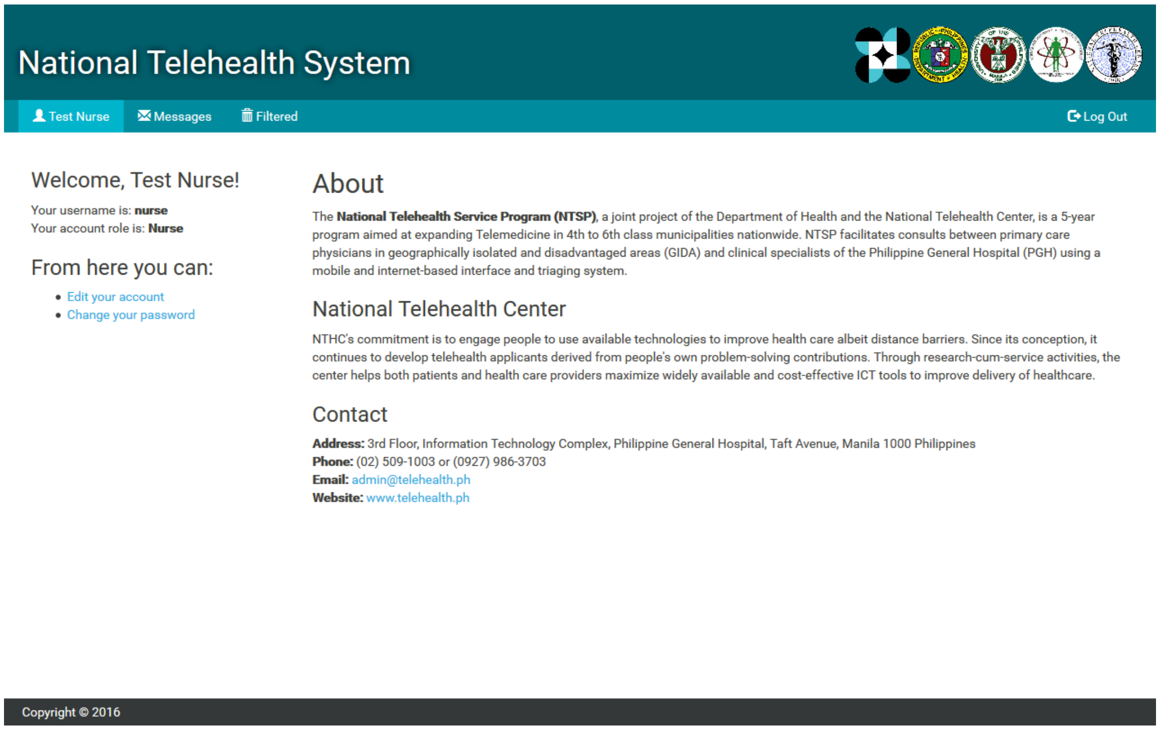
Copyright © 2016

Figure 21: Log in page, *TAGR*

Figure 21 shows the home page of TAGR. It contains a login form that allows users with the proper credentials access to the rest of the system. It also contains a brief description of the system, and contact information of the National Telehealth Center.

B. Telehealth Nurse View

Welcome Page



Welcome, Test Nurse!

Your username is: **nurse**
Your account role is: **Nurse**

From here you can:

- Edit your account
- Change your password

About

The **National Telehealth Service Program (NTSP)**, a joint project of the Department of Health and the National Telehealth Center, is a 5-year program aimed at expanding Telemedicine in 4th to 6th class municipalities nationwide. NTSP facilitates consults between primary care physicians in geographically isolated and disadvantaged areas (GIDA) and clinical specialists of the Philippine General Hospital (PGH) using a mobile and internet-based interface and triaging system.

National Telehealth Center

NTHC's commitment is to engage people to use available technologies to improve health care albeit distance barriers. Since its conception, it continues to develop telehealth applicants derived from people's own problem-solving contributions. Through research-cum-service activities, the center helps both patients and health care providers maximize widely available and cost-effective ICT tools to improve delivery of healthcare.

Contact

Address: 3rd Floor, Information Technology Complex, Philippine General Hospital, Taft Avenue, Manila 1000 Philippines
Phone: (02) 509-1003 or (0927) 986-3703
Email: admin@telehealth.ph
Website: www.telehealth.ph

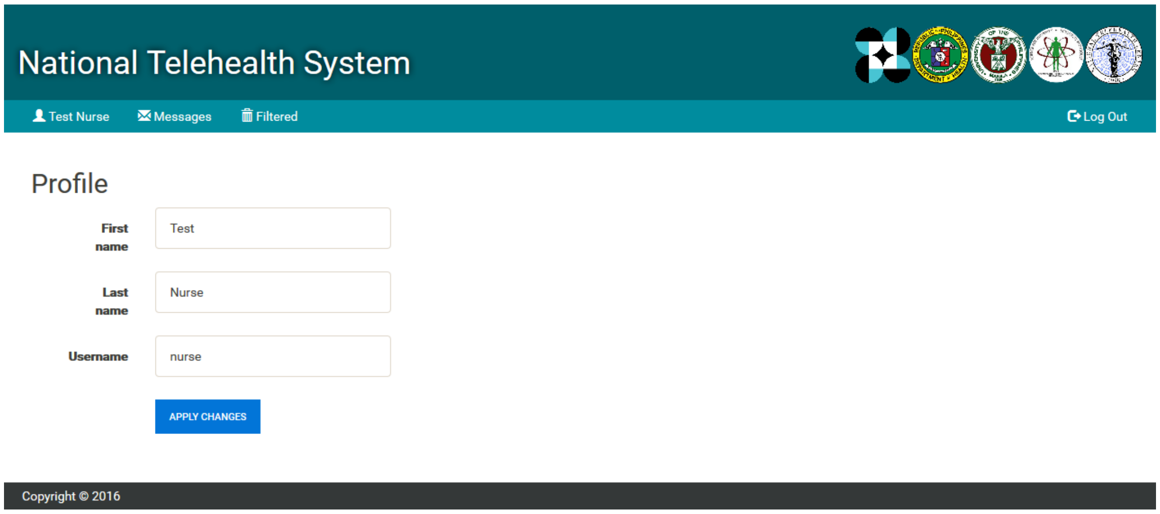
Copyright © 2016

Figure 22: Nurse welcome page, *TAGR*

After logging in, the telehealth nurse is greeted by a welcome page. As in Figure 22, it is similar to the login page, except the login form has been replaced with a description of the user and user-related links. Also, the navigation menu is now visible. The nurse has access to the Messages and Filtered pages.

The edit account page in Figure 23 lets the user edit his/her first name, last name, and username.

The change password in Figure 24 lets the user change his/her password. The user must provide their old password before entering the new password twice for


National Telehealth System

Test Nurse | Messages | Filtered | Log Out

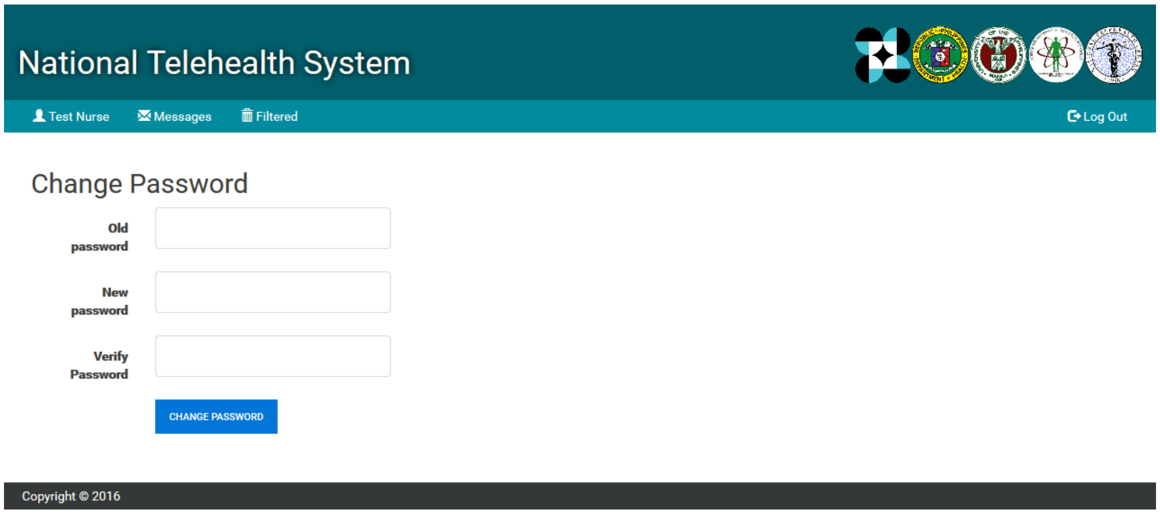
Profile

First name	Test
Last name	Nurse
Username	nurse

APPLY CHANGES

Copyright © 2016

Figure 23: Edit profile details, *TAGR*


National Telehealth System

Test Nurse | Messages | Filtered | Log Out

Change Password

Old password	
New password	
Verify Password	

CHANGE PASSWORD

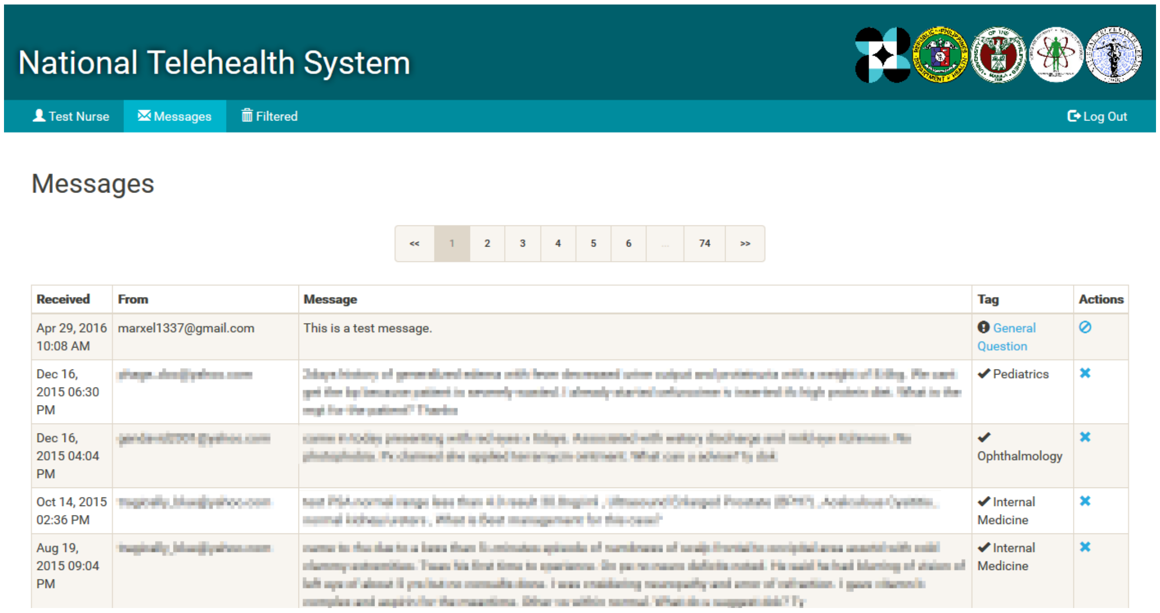
Copyright © 2016

Figure 24: Edit password, *TAGR*

verification. Passwords must have a minimum length of 8 characters and must consist of alphanumeric characters only.

Messages

The Messages page show the messages stored in the system in tabular format. As in Figure 25, the following details are shown: date and time that the messages was received, who sent the message, the message contents, the message tag, and the available actions. The messages are sorted in a descending order from time of receipt

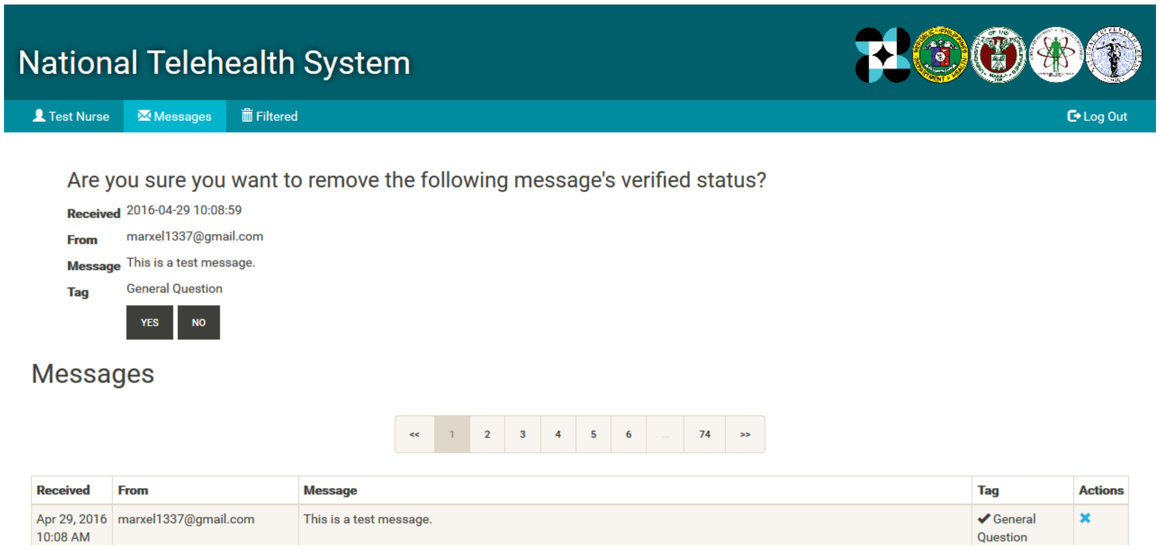


Messages

Received	From	Message	Tag	Actions
Apr 29, 2016 10:08 AM	marxel1337@gmail.com	This is a test message.	General Question	⊘
Dec 16, 2015 06:30 PM	[redacted]	[redacted]	✓ Pediatrics	✖
Dec 16, 2015 04:04 PM	[redacted]	[redacted]	✓ Ophthalmology	✖
Oct 14, 2015 02:36 PM	[redacted]	[redacted]	✓ Internal Medicine	✖
Aug 19, 2015 09:04 PM	[redacted]	[redacted]	✓ Internal Medicine	✖

Figure 25: View Messages, *TAGR*

(newest first). Messages that are verified have check marks in the tag column. The only available action for them is to remove the verification. For unverified messages, they can either be marked as spam, or edited and verified.



Are you sure you want to remove the following message's verified status?

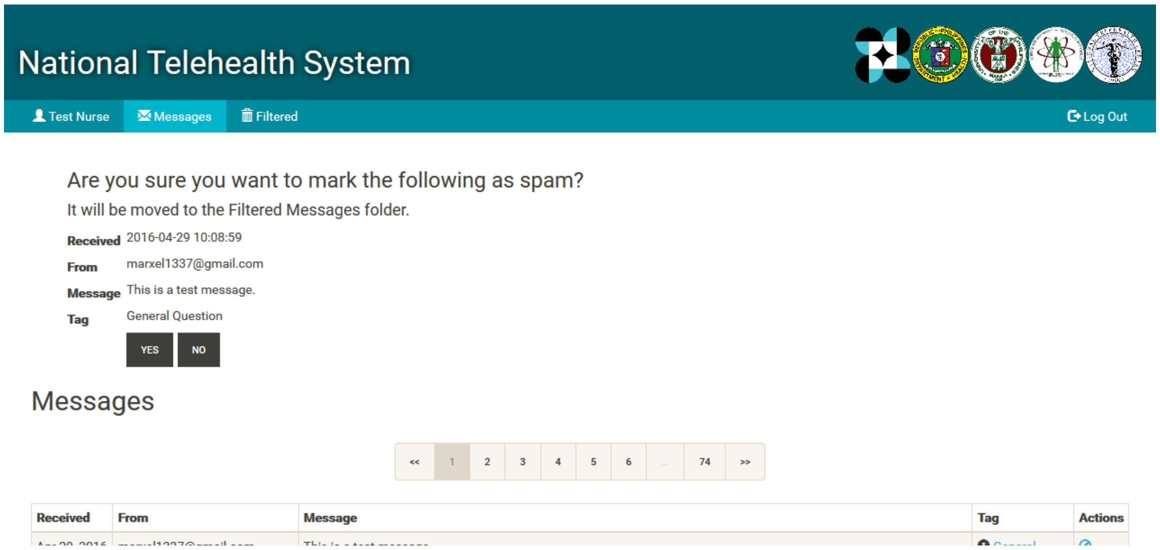
Received	2016-04-29 10:08:59
From	marxel1337@gmail.com
Message	This is a test message.
Tag	General Question

Messages

Received	From	Message	Tag	Actions
Apr 29, 2016 10:08 AM	marxel1337@gmail.com	This is a test message.	✓ General Question	✖

Figure 26: Confirm removal of verified status, *TAGR*

Removing a message's verification status directs to a confirmation page as shown in Figure 26. Confirming that a message is spam is shown in Figure 27.


National Telehealth System

Test Nurse | Messages | Filtered | Log Out

Are you sure you want to mark the following as spam?

It will be moved to the Filtered Messages folder.

Received	2016-04-29 10:08:59
From	marxel1337@gmail.com
Message	This is a test message.
Tag	General Question

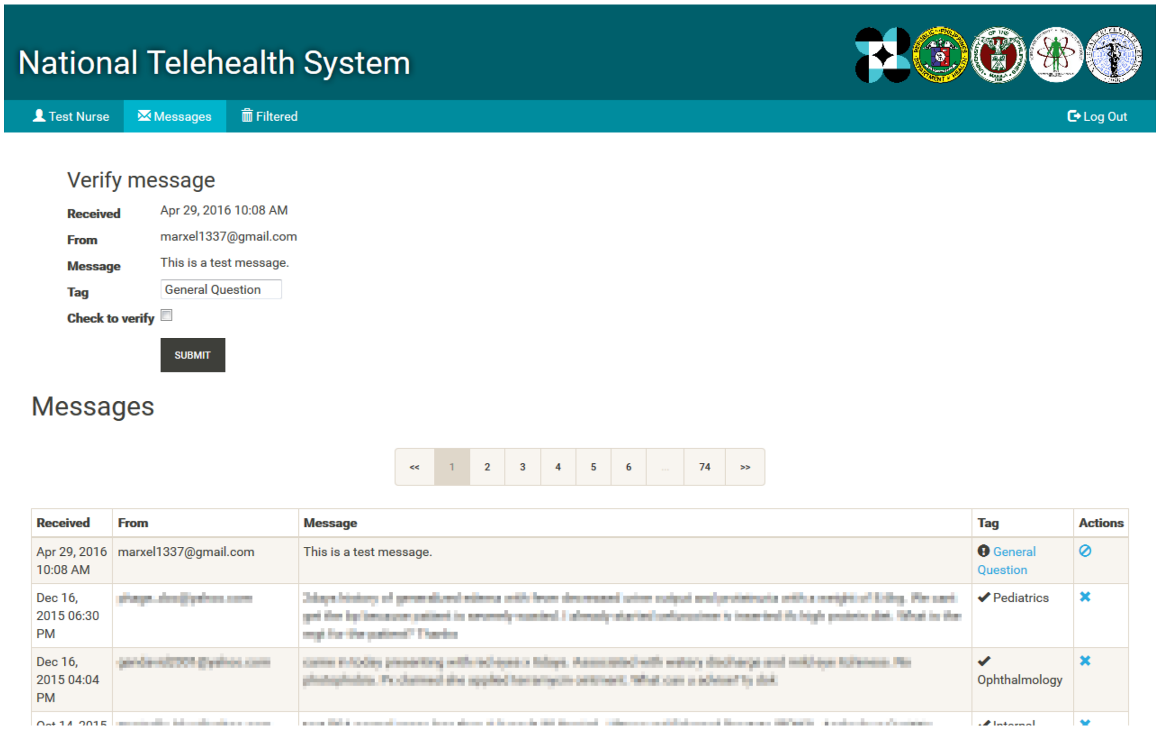
YES NO

Messages

<< 1 2 3 4 5 6 ... 74 >>

Received	From	Message	Tag	Actions

Figure 27: Confirm marking a message as spam, *TAGR*


National Telehealth System

Test Nurse | Messages | Filtered | Log Out

Verify message

Received	Apr 29, 2016 10:08 AM
From	marxel1337@gmail.com
Message	This is a test message.
Tag	General Question
Check to verify	☐

SUBMIT

Messages

<< 1 2 3 4 5 6 ... 74 >>

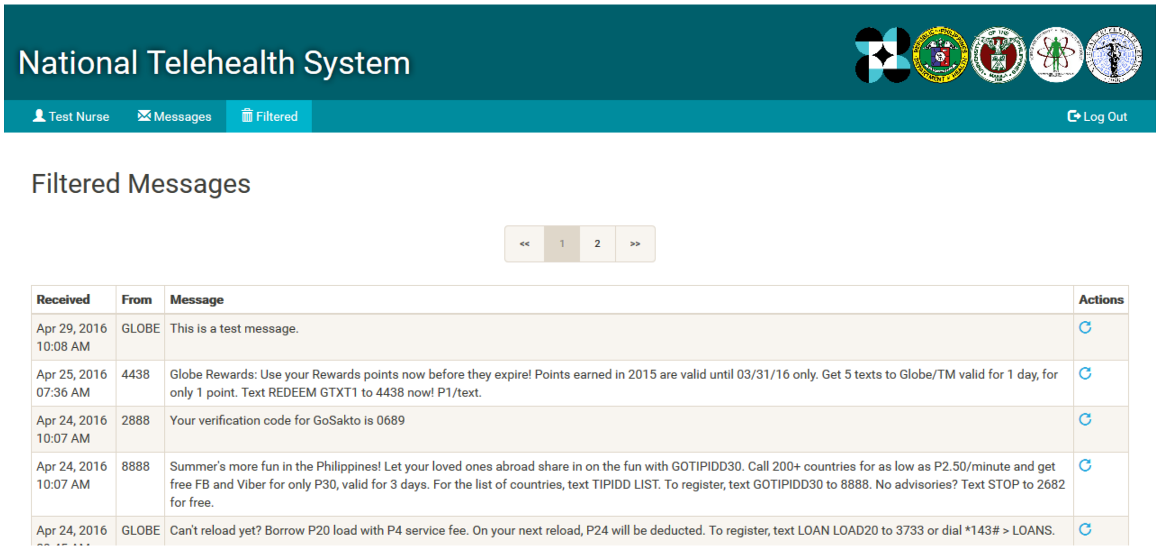
Received	From	Message	Tag	Actions
Apr 29, 2016 10:08 AM	marxel1337@gmail.com	This is a test message.	General Question	⊘
Dec 16, 2015 06:30 PM	[redacted]	[redacted]	✓ Pediatrics	✖
Dec 16, 2015 04:04 PM	[redacted]	[redacted]	✓ Ophthalmology	✖

Figure 28: Verify a message, *TAGR*

Clicking on an unverified message's tag brings the verification page. Aside from verifying the message, the user can change the message's assigned tag. This is shown in Figure 28

Filtered Messages

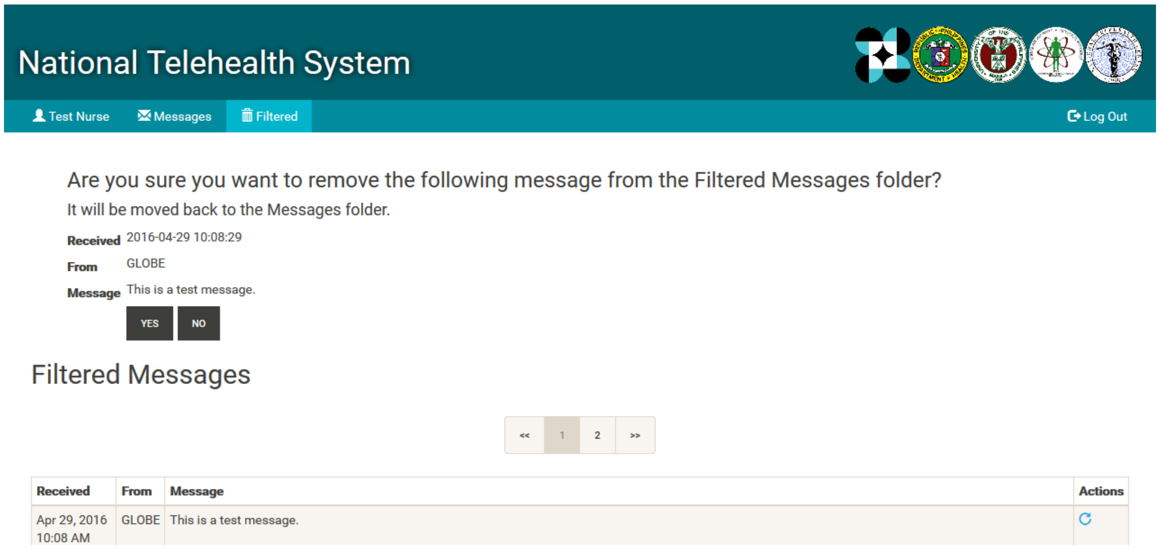
The Filtered Messages is virtually similar to the Messages except the messages displayed are either automatically filtered messages or messages marked as spam. The only available action for the user here is to unmark a message as spam.



Received	From	Message	Actions
Apr 29, 2016 10:08 AM	GLOBE	This is a test message.	
Apr 25, 2016 07:36 AM	4438	Globe Rewards: Use your Rewards points now before they expire! Points earned in 2015 are valid until 03/31/16 only. Get 5 texts to Globe/TM valid for 1 day, for only 1 point. Text REDEEM GTXT1 to 4438 now! P1/text.	
Apr 24, 2016 10:07 AM	2888	Your verification code for GoSakto is 0689	
Apr 24, 2016 10:07 AM	8888	Summer's more fun in the Philippines! Let your loved ones abroad share in on the fun with GOTIPIDD30. Call 200+ countries for as low as P2.50/minute and get free FB and Viber for only P30, valid for 3 days. For the list of countries, text TIPIDD LIST. To register, text GOTIPIDD30 to 8888. No advisories? Text STOP to 2682 for free.	
Apr 24, 2016	GLOBE	Can't reload yet? Borrow P20 load with P4 service fee. On your next reload, P24 will be deducted. To register, text LOAN LOAD20 to 3733 or dial *143# > LOANS.	

Figure 29: View Filtered Messages, *TAGR*

Figure 29 shows the Filtered Messages page while Figure 30 shows the confirmation page when a message is unmarked as spam.



Are you sure you want to remove the following message from the Filtered Messages folder?

It will be moved back to the Messages folder.

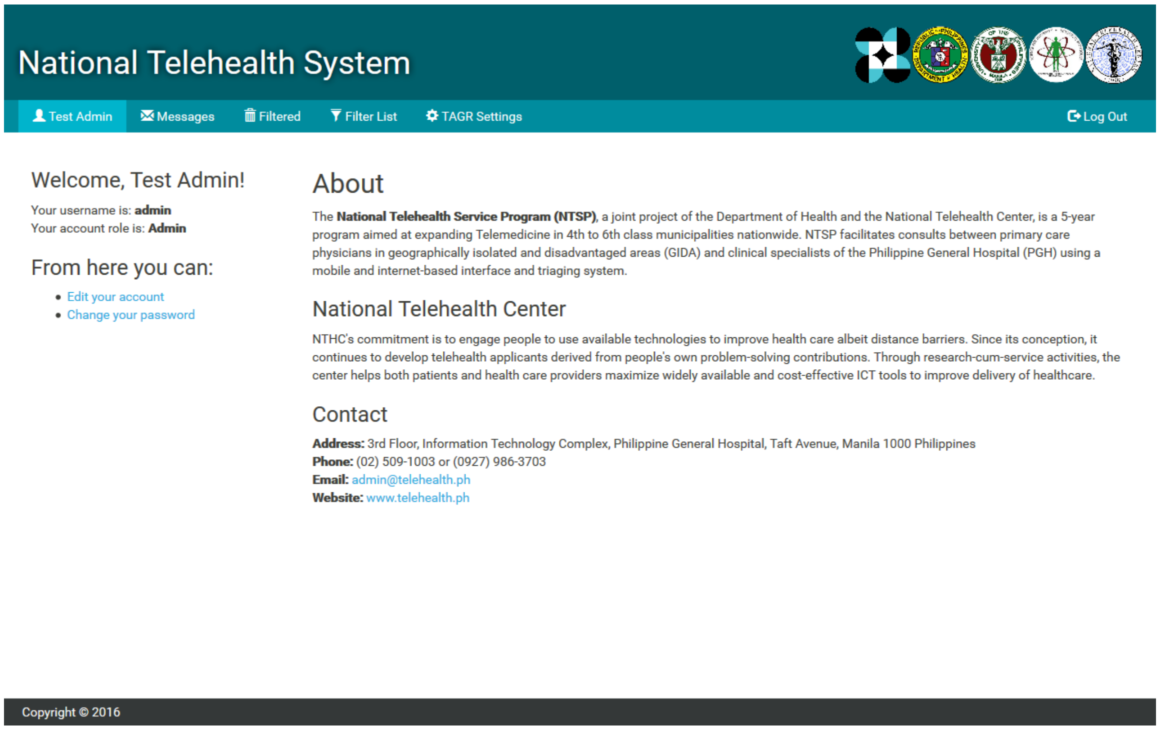
Received	2016-04-29 10:08:29
From	GLOBE
Message	This is a test message.

Received	From	Message	Actions
Apr 29, 2016 10:08 AM	GLOBE	This is a test message.	

Figure 30: Confirm unmarking a message as spam, *TAGR*

C. Administrator View

Welcome Page



Welcome, Test Admin!

Your username is: **admin**
Your account role is: **Admin**

From here you can:

- Edit your account
- Change your password

About

The **National Telehealth Service Program (NTSP)**, a joint project of the Department of Health and the National Telehealth Center, is a 5-year program aimed at expanding Telemedicine in 4th to 6th class municipalities nationwide. NTSP facilitates consults between primary care physicians in geographically isolated and disadvantaged areas (GIDA) and clinical specialists of the Philippine General Hospital (PGH) using a mobile and internet-based interface and triaging system.

National Telehealth Center

NTHC's commitment is to engage people to use available technologies to improve health care albeit distance barriers. Since its conception, it continues to develop telehealth applicants derived from people's own problem-solving contributions. Through research-cum-service activities, the center helps both patients and health care providers maximize widely available and cost-effective ICT tools to improve delivery of healthcare.

Contact

Address: 3rd Floor, Information Technology Complex, Philippine General Hospital, Taft Avenue, Manila 1000 Philippines
Phone: (02) 509-1003 or (0927) 986-3703
Email: admin@telehealth.ph
Website: www.telehealth.ph

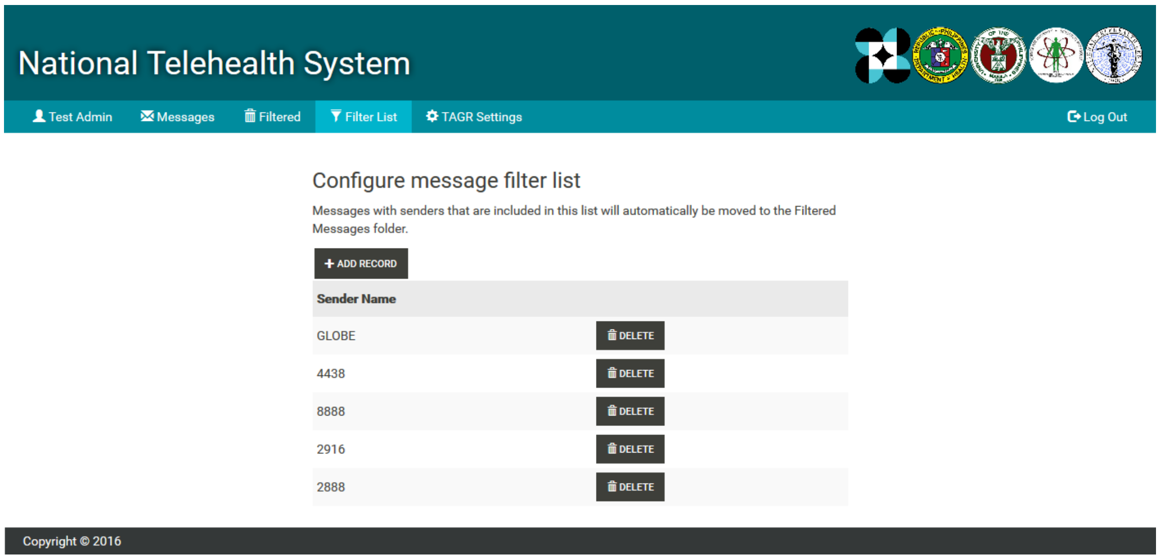
Copyright © 2016

Figure 31: Admin welcome page, *TAGR*

After logging in, the administrator is greeted by a welcome page. As in Figure 31, it is similar to the telehealth nurse's welcome page (Figure 22). The administrator has access to the Filter List and TAGR Settings pages in addition to all the functions that the Telehealth nurse can access. This is reflected in the administrator's version of the navigation menu.

Filter List

One of the administrator's functions is to configure the filter list. The Filter List page shows a list of sender names. If a new message's sender matches an item in this list, it is automatically filtered. The list can be changed by adding new items and deleting existing ones. The filter list page is shown in Figure 32



Configure message filter list

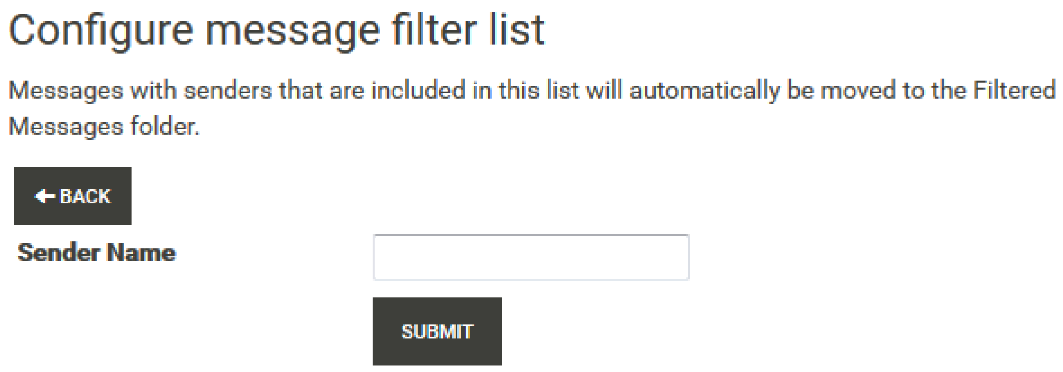
Messages with senders that are included in this list will automatically be moved to the Filtered Messages folder.

Sender Name	
GLOBE	DELETE
4438	DELETE
8888	DELETE
2916	DELETE
2888	DELETE

Copyright © 2016

Figure 32: Configure message filter list, *TAGR*

Add Record brings the page to a form as shown in Figure 33.

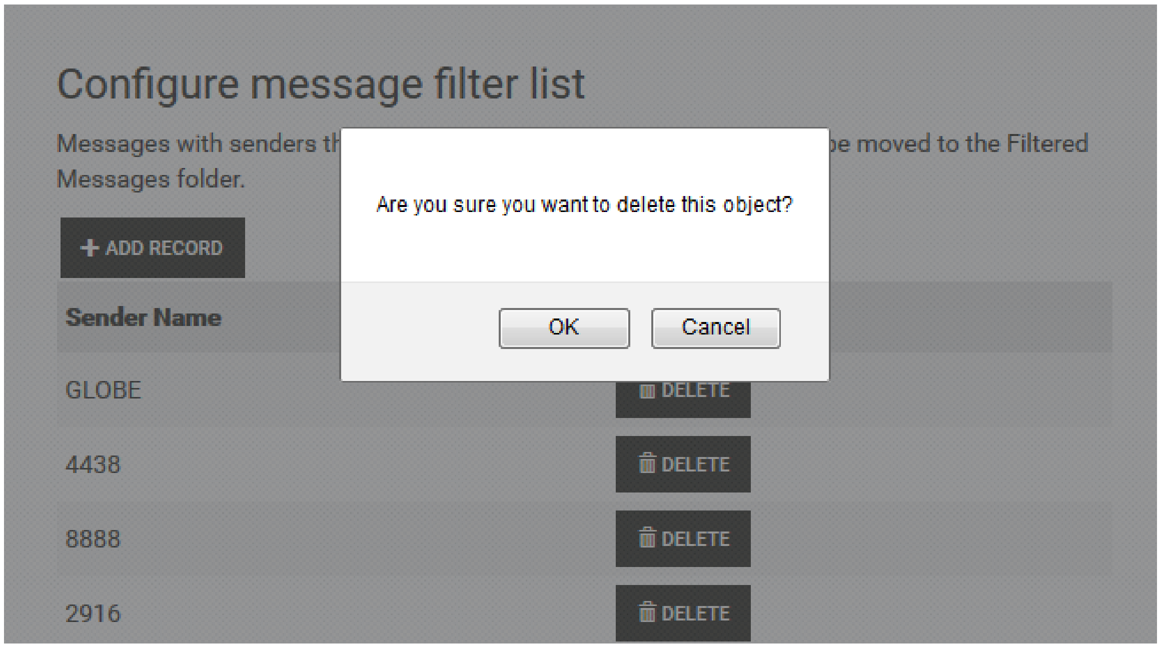


Configure message filter list

Messages with senders that are included in this list will automatically be moved to the Filtered Messages folder.

Figure 33: Adding an item to the message filter list, *TAGR*

Deleting an item on the list will trigger a confirmation prompt as in Figure 34.


Configure message filter list

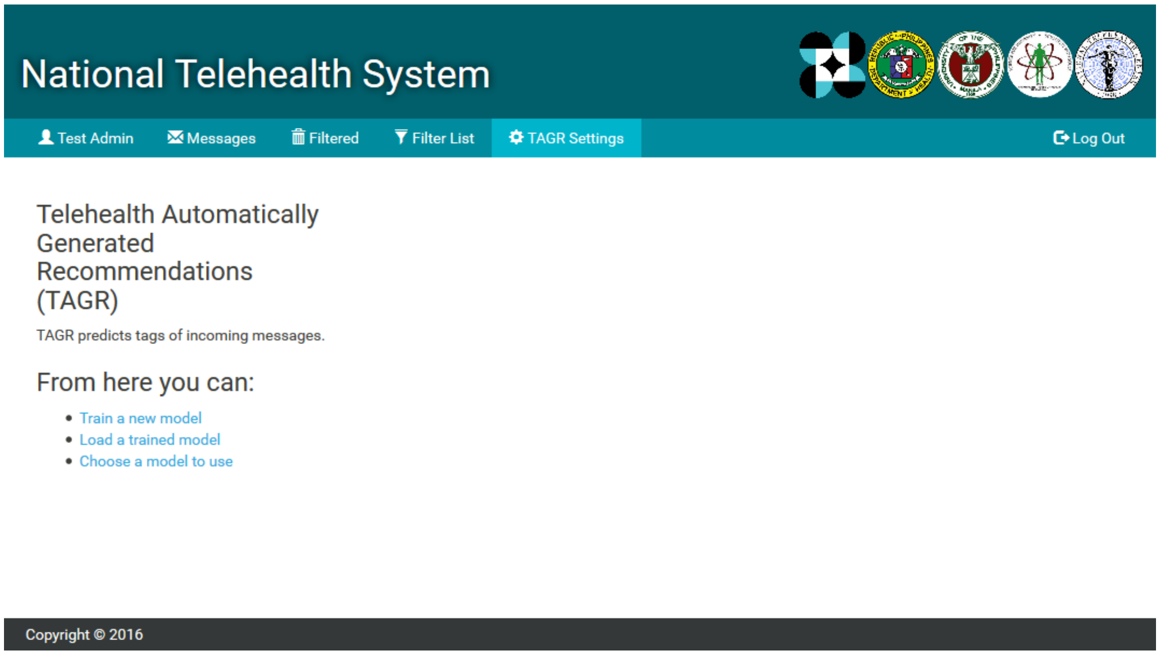
Messages with senders th... be moved to the Filtered Messages folder.

Are you sure you want to delete this object?

Sender Name	
GLOBE	DELETE
4438	DELETE
8888	DELETE
2916	DELETE

Figure 34: Deleting an item from the message filter list, *TAGR*

TAGR Settings



Telehealth Automatically Generated Recommendations (TAGR)

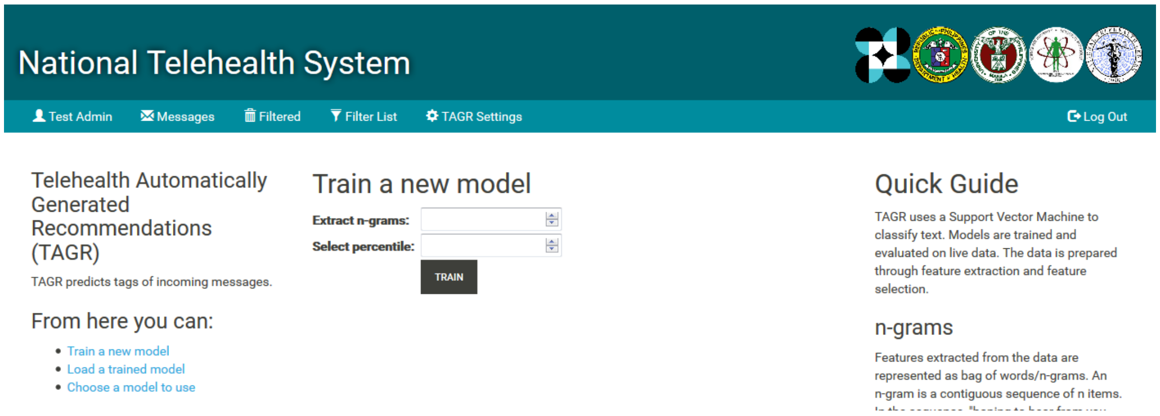
TAGR predicts tags of incoming messages.

From here you can:

- Train a new model
- Load a trained model
- Choose a model to use

Figure 35: TAGR Settings page, *TAGR*

The TAGR Settings page contains links to further functions that the administrator can access. These are: Train a new model, Load a trained model, and Choose a model to use, seen in Figure 35.


National Telehealth System

Test Admin · Messages · Filtered · Filter List · TAGR Settings · Log Out

Telehealth Automatically Generated Recommendations (TAGR)

TAGR predicts tags of incoming messages.

From here you can:

- Train a new model
- Load a trained model
- Choose a model to use

Train a new model

Extract n-grams:

Select percentile:

TRAIN

Quick Guide

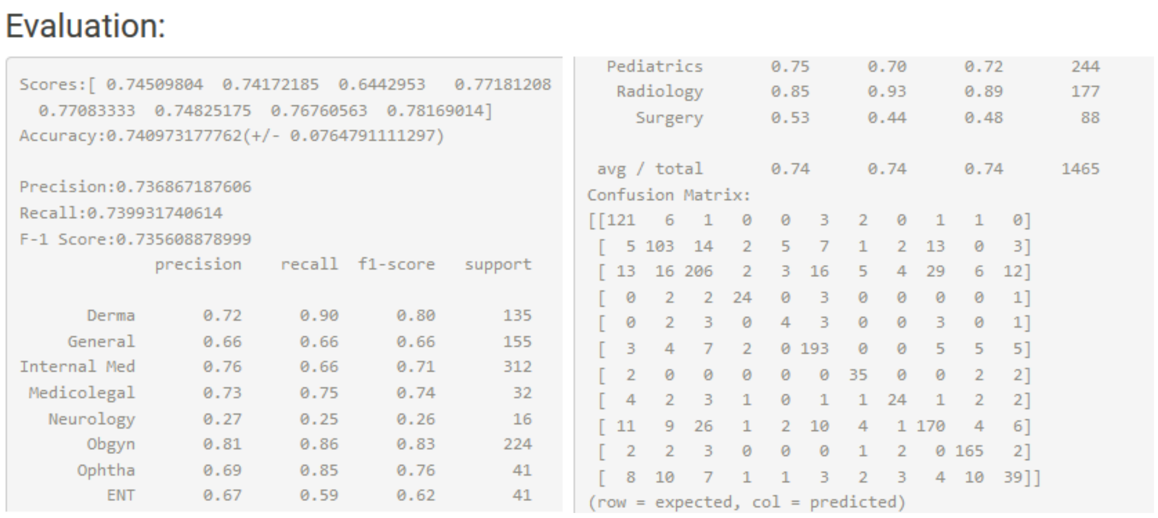
TAGR uses a Support Vector Machine to classify text. Models are trained and evaluated on live data. The data is prepared through feature extraction and feature selection.

n-grams

Features extracted from the data are represented as bag of words/n-grams. An n-gram is a contiguous sequence of n items.

Figure 36: Train a new model, *TAGR*

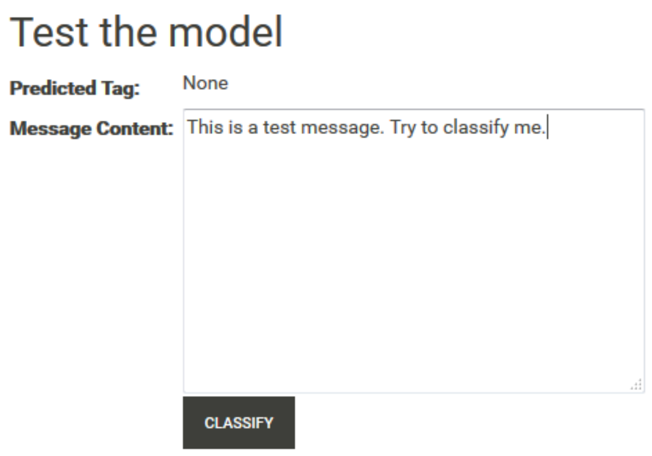
Choosing to train a new model brings the user to the page in Figure 36. The administrator can specify the two parameters, n-grams and percentile before training the model. A brief guide to the values is also displayed on the right.


Evaluation:

```
Scores:[ 0.74509804  0.74172185  0.6442953   0.77181208
  0.77083333  0.74825175  0.76760563  0.78169014]
Accuracy:0.740973177762(+/- 0.0764791111297)

Precision:0.736867187606
Recall:0.739931740614
F-1 Score:0.735608878999
              precision    recall  f1-score   support

       Derma       0.72      0.90      0.80       135
     General       0.66      0.66      0.66       155
Internal Med       0.76      0.66      0.71       312
 Medicolegal       0.73      0.75      0.74        32
   Neurology       0.27      0.25      0.26        16
       Obgyn       0.81      0.86      0.83       224
      Ophtha       0.69      0.85      0.76        41
         ENT       0.67      0.59      0.62        41
  Pediatrics       0.75      0.70      0.72       244
   Radiology       0.85      0.93      0.89       177
     Surgery       0.53      0.44      0.48        88

 avg / total       0.74      0.74      0.74      1465
Confusion Matrix:
[[121   6   1   0   0   3   2   0   1   1   0]
 [  5 103  14   2   5   7   1   2  13   0   3]
 [ 13  16 206   2   3  16   5   4  29   6  12]
 [  0   2   2  24   0   3   0   0   0   0   1]
 [  0   2   3   0   4   3   0   0   3   0   1]
 [  3   4   7   2   0 193   0   0   5   5   5]
 [  2   0   0   0   0   0  35   0   0   2   2]
 [  4   2   3   1   0   1   1  24   1   2   2]
 [ 11   9  26   1   2  10   4   1 170   4   6]
 [  2   2   3   0   0   0   1   2   0 165   2]
 [  8  10   7   1   1   3   2   3   4  10  39]]
(row = expected, col = predicted)
```

Figure 37: Training evaluation, *TAGR*

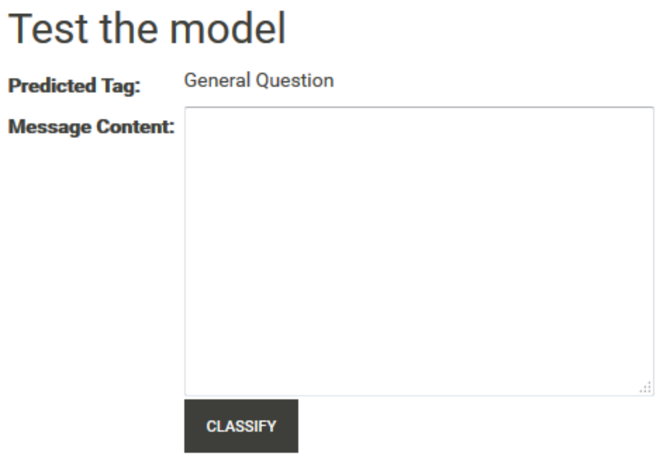
After training a model, it is evaluated via 10-fold cross validation. The evaluation report is displayed in Figure 37. The metrics used are accuracy, precision, recall, and F-score. The evaluation report also includes details per class, as well as a confusion matrix. Here, the trained model is shown to have 74% accuracy.


Test the model

Predicted Tag:	None
Message Content:	This is a test message. Try to classify me.

Figure 38: Manual testing, *TAGR*

Right after the evaluation report is a form where the trained model can manually be tested. The form is shown in Figure 38


Test the model

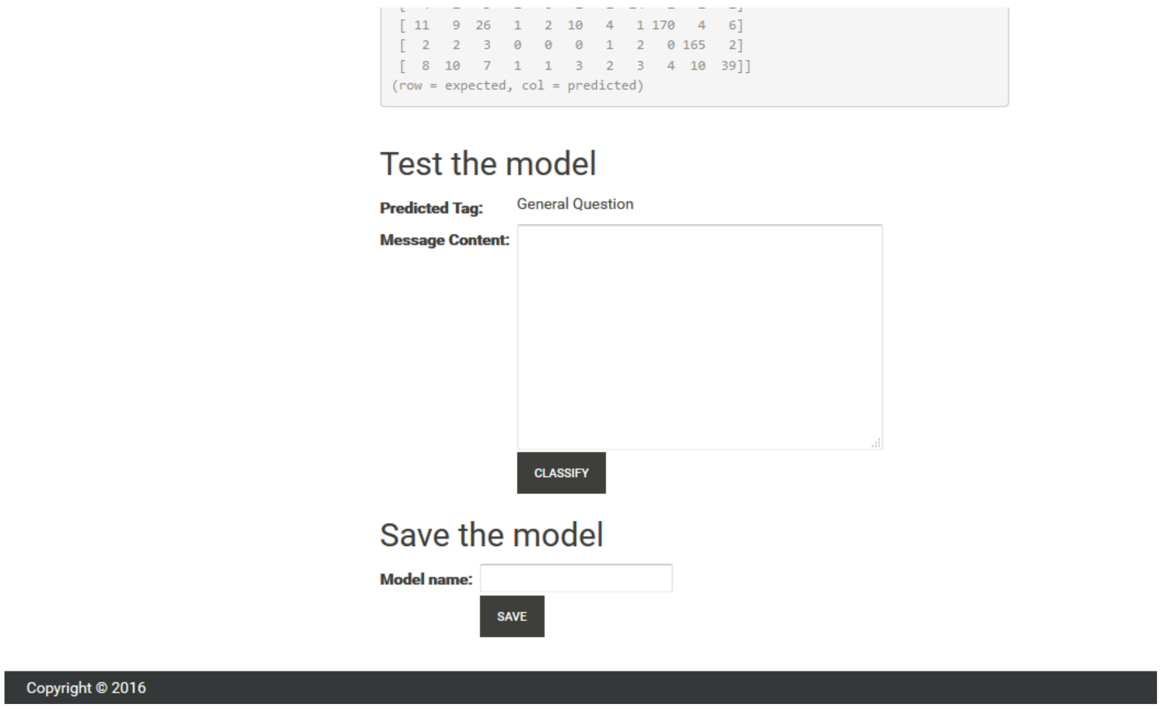
Predicted Tag:	General Question
Message Content:	

Figure 39: Manual testing with prediction, *TAGR*

The result of the manual test is displayed in the same form as in Figure 39.

Once a model has been trained and evaluated, the administrator can choose to save the model by giving it a name. This is shown in Figure 40.

Choosing to load a model will load a list of saved models. The list includes details such as the date when the model was created, the model name, the parameters, and its deployment status. This can be seen in Figure 41. A dropdown list allows the



```
 [ 11   9  26   1   2  10   4   1 170   4   6]
 [  2   2   3   0   0   0   1   2   0 165   2]
 [  8  10   7   1   1   3   2   3   4  10  39]]
(row = expected, col = predicted)
```

Test the model

Predicted Tag: General Question

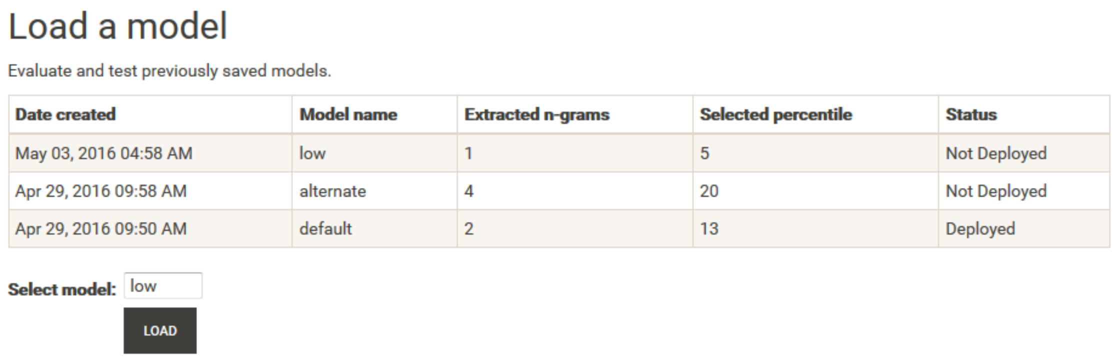
Message Content:

Save the model

Model name:

Copyright © 2016

Figure 40: Save the trained model, *TAGR*



Load a model

Evaluate and test previously saved models.

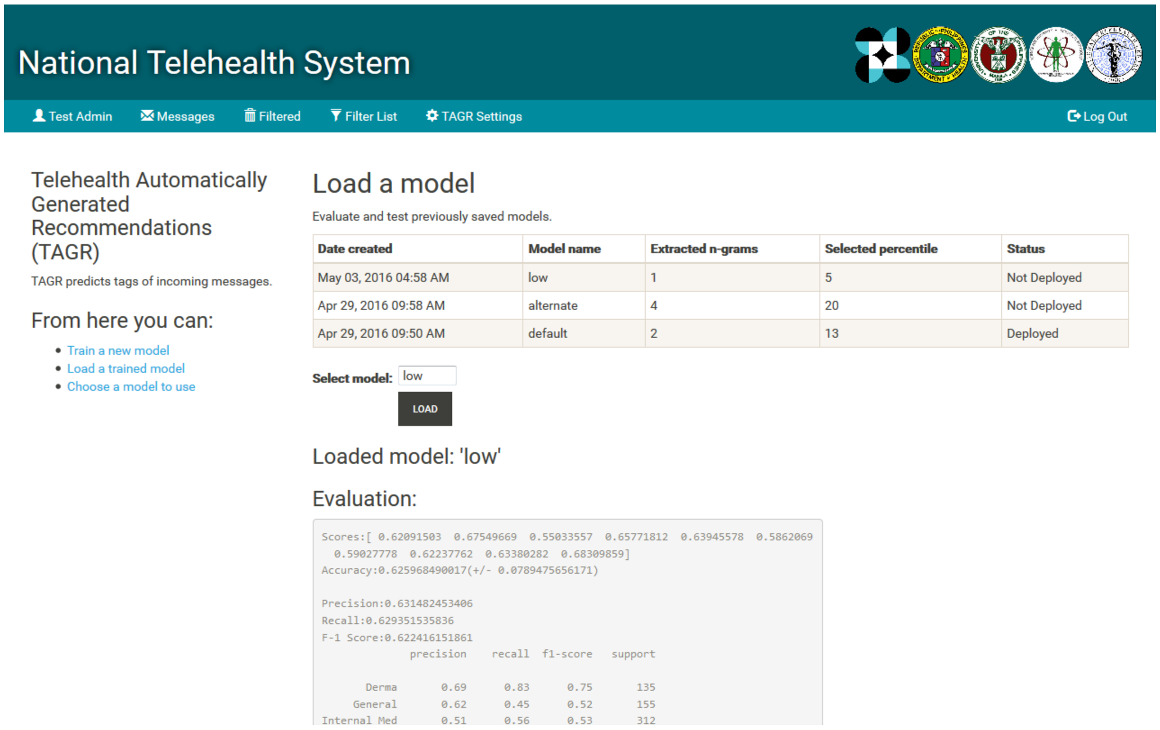
Date created	Model name	Extracted n-grams	Selected percentile	Status
May 03, 2016 04:58 AM	low	1	5	Not Deployed
Apr 29, 2016 09:58 AM	alternate	4	20	Not Deployed
Apr 29, 2016 09:50 AM	default	2	13	Deployed

Select model: low

Figure 41: Load a trained model, *TAGR*

user to select which model to load. When a model is loaded, it is evaluated in the same vein as new models. This is seen in Figure 42.

Choosing a model brings the administrator to a page (Figure 43) similar to the load model page in Figure 41.


National Telehealth System

Test Admin | Messages | Filtered | Filter List | TAGR Settings | Log Out

Telehealth Automatically Generated Recommendations (TAGR)

TAGR predicts tags of incoming messages.

From here you can:

- Train a new model
- Load a trained model
- Choose a model to use

Load a model

Evaluate and test previously saved models.

Date created	Model name	Extracted n-grams	Selected percentile	Status
May 03, 2016 04:58 AM	low	1	5	Not Deployed
Apr 29, 2016 09:58 AM	alternate	4	20	Not Deployed
Apr 29, 2016 09:50 AM	default	2	13	Deployed

Select model: low

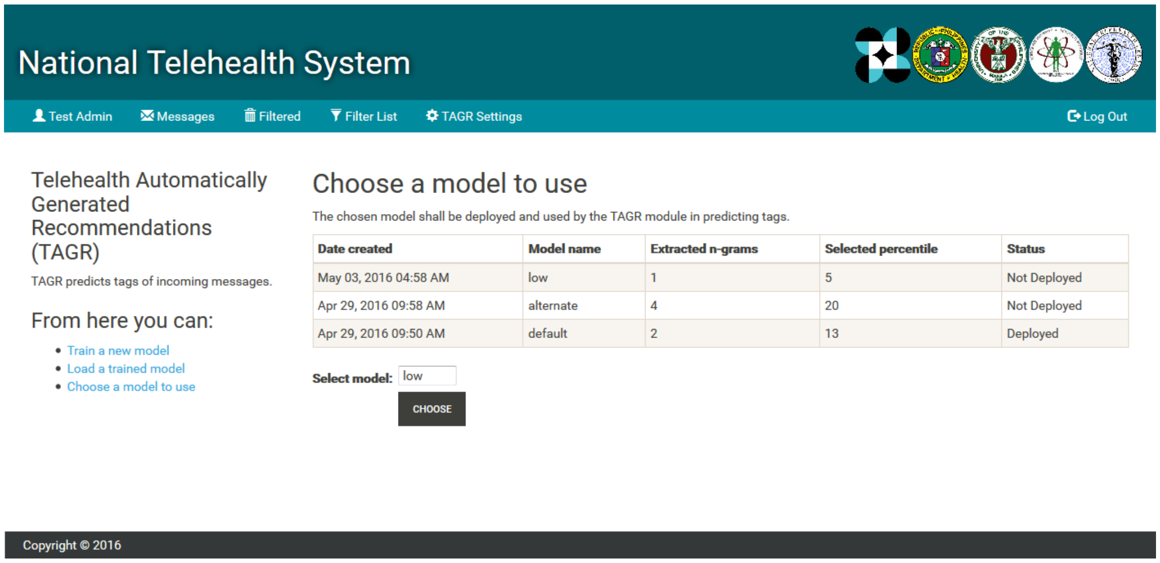
LOAD

Loaded model: 'low'

Evaluation:

```
Scores:[ 0.62091503  0.67549669  0.55033557  0.65771812  0.63945578  0.5862069
  0.59027778  0.62237762  0.63380282  0.68309859]
Accuracy:0.625968490017(+/- 0.0789475656171)

Precision:0.631482453406
Recall:0.629351535836
F-1 Score:0.622416151861
              precision    recall  f1-score   support


      Derma       0.69      0.83      0.75       135
    General       0.62      0.45      0.52       155
Internal Med       0.51      0.56      0.53       312
```

Figure 42: Loaded model evaluation, *TAGR*


National Telehealth System

Test Admin | Messages | Filtered | Filter List | TAGR Settings | Log Out

Telehealth Automatically Generated Recommendations (TAGR)

TAGR predicts tags of incoming messages.

From here you can:

- Train a new model
- Load a trained model
- Choose a model to use

Choose a model to use

The chosen model shall be deployed and used by the TAGR module in predicting tags.

Date created	Model name	Extracted n-grams	Selected percentile	Status
May 03, 2016 04:58 AM	low	1	5	Not Deployed
Apr 29, 2016 09:58 AM	alternate	4	20	Not Deployed
Apr 29, 2016 09:50 AM	default	2	13	Deployed

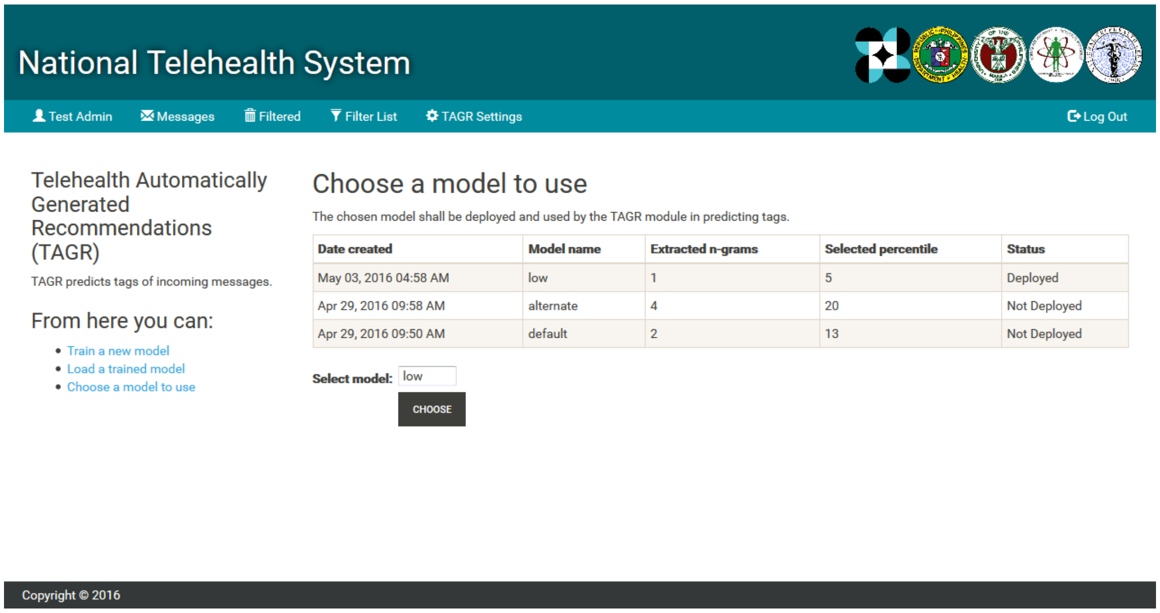
Select model: low

CHOOSE

Copyright © 2016

Figure 43: Choose a model to use, *TAGR*

When a model is chosen, it is given the Deployed status while the rest are given the Not Deployed status as seen in Figure 44


National Telehealth System

Telehealth Automatically Generated Recommendations (TAGR)

TAGR predicts tags of incoming messages.

From here you can:

- Train a new model
- Load a trained model
- Choose a model to use

Choose a model to use

The chosen model shall be deployed and used by the TAGR module in predicting tags.

Date created	Model name	Extracted n-grams	Selected percentile	Status
May 03, 2016 04:58 AM	low	1	5	Deployed
Apr 29, 2016 09:58 AM	alternate	4	20	Not Deployed
Apr 29, 2016 09:50 AM	default	2	13	Not Deployed

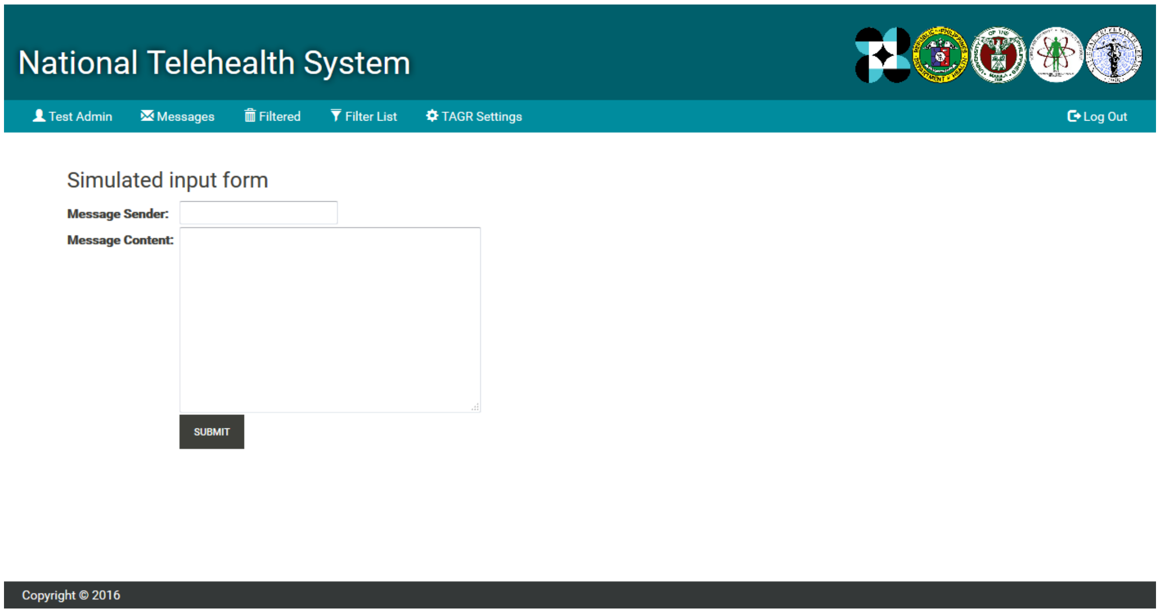
Select model: low

Copyright © 2016

Figure 44: A different model is deployed, *TAGR*

Hidden input page

A hidden input page is provided to allow the system to simulate incoming messages. The page includes a form with message sender and content fields shown in Figure 45. The page cannot be accesed without knowing its address.


National Telehealth System

Simulated input form

Message Sender:

Message Content:

Copyright © 2016

Figure 45: Hidden input page for simulating new messages, *TAGR*

VI. Discussions

Telehealth Automatically Generated Recommendations (TAGR) is a module for automatically classifying SMS and e-mail messages with their appropriate tags as required by the National Telehealth Center for their National Telehealth Service Program (NTSP). For demonstration purposes, a minimal web-based interface was built around the TAGR module which simulates the real-world processes of the NTSP. The web-based component allows telehealth nurses to manage the messages in the system. It also allows the TAGR module to be configured by administrators.

The inclusion of a rudimentary spam blocker frees the telehealth nurses of the burden of having to deal with unsolicited messages. This allows them to focus their time and effort on the verification of messages.

The TAGR module's classifier trained on the base dataset yields an average prediction accuracy of 73% upon validation. The accuracy may be considered decent enough to allow messages to be sent by rural physicians without keywords. The absence of keywords would not pose a problem for the telehealth nurses, as the TAGR module automatically assigns the appropriate tag. As a fallback measure, telehealth nurses are able to correct the assigned tag.

The TAGR module can also train on both new data and old but corrected data. This opens the possibility of improving the accuracy of the classifier even if the module is deployed in a production environment.

The dataset used was provided by the National Telehealth Center and contains actual messages processed by the NTSP. They were given in two batches totaling 1733 messages. However, only a total of 1464 messages were used.

As seen in Table 8, the dataset is imbalanced. The Internal Medicine class contains 481 messages while the Psychology class only has 8 messages. It is also noted that three classes: Internal Medicine, Pediatrics, and Family Medicine, have very similar occurring features in them, which can easily confuse the classifier.

Tag	Batch 1	Batch 2	Total	Used
Internal Medicine	312	169	481	312
Pediatrics	244	83	327	244
Obgyn	157	67	224	224
General Question	111	44	155	155
Surgery	48	40	88	88
Radiology	49	127	176	176
Dermatology	29 1	106	135	32
Medico Legal	24	8	32	32
Ophthalmology	19	22	41	41
Otorhinolaryngology	18	23	41	41
Neurology	14	2	16	16
Psychology	0	8	8	0
Family Medicine	9	0	9	0
Total	1034	699	1733	1464

Table 8: Data from National Telehealth Center

To reduce the imbalance, messages under Internal Medicine and Pediatrics from the second batch of data were not included in the dataset used by the classifier. Messages under the Psychology and Family Medicine classes were also removed, as they had too few instances. However, the final dataset was still fairly imbalanced.

The nature of the dataset also lead to the scrapping of the planned use of Conditional Random Fields (CRF) as the machine learning method for the classifier in favor of Support Vector Machines. CRF would possibly be better for sequential data such as natural language text because of how it takes into account context and patterns. For example, a word is likely to be of a certain class if it is preceded by words from another class.

The dataset however, did not reveal any distinct patterns for each individual class. Rather, classes relied more on occurrences of words specific to that class than with the order in which they appear in. Here, the classifier may benefit on a dictionary of medical terms and their associated class. The CRF would have been more useful in separating the data into relevant and irrelevant chunks. A different classifier can then make a prediction using only the relevant chunks.

The classifier was built using the scikit-learn library. Specifically, the classifier is an instance of the LinearSVC object. The LinearSVC is an implementation of a support vector machine with a linear kernel. While TAGR lets the user specify n-grams and feature percentiles, the classifier itself has preset parameters.

These preset parameters were chosen after running a grid search on the classifier, and returned the best performance. The parameters of the LinearSVC are:

```
C = 1
loss = 'squared_hinge'
penalty = 'l2'
class_weight = 'balanced'
```

There was difficulty in integrating the classifier component with the web-based component. The classifier component was not designed to run as a live web service. This posed a problem as TAGR configuration required a persistent instance of the classifier that can be accessed across multiple pages. This was solved by caching an instance in the server with no expiry.

VII. Conclusions

Telehealth Automatically Generated Recommendations (TAGR) is a module for automatically classifying SMS and e-mail messages with their appropriate tags as required by the National Telehealth Center for their National Telehealth Service Program (NTSP). To accomplish this, it utilizes natural language processing techniques such as tokenization, n-gram representation, and term frequency-inverse document frequency weighting, coupled with support vector machines, a machine learning method.

The module was developed with the ultimate goal of providing an automated tagging component running in real-time for the existing National Telehealth Service Program system. With its decent base accuracy, it can be considered for deployment into a production environment.

The author considers this a positive study in that it was able to achieve the objectives. It also demonstrates the potential of utilizing natural language processing techniques and machine learning methods in more real-world applications, particularly in telehealth and other related fields.

VIII. Recommendations

The Telehealth Automatically Generated Recommendations (TAGR) module as it stands now, is tightly connected to the web-based interface that was built around it for demonstration purposes. This bars it from being called a true stand-alone module that can easily be plugged into systems. It is recommended to modify TAGR into an executable that accepts parameters so that it can be called anytime through scripts.

Another improvement is to configure TAGR as such that it can be run as a web service. Currently, an instance of TAGR is always created on demand. This can be costly performance-wise in the case that multiple TAGR instances are created at the same time.

While the TAGR classifier's accuracy may be considered decent enough, there is still room for improvement. The following are some recommendations on improving accuracy:

1. Build a balanced dataset to train on.

2. Explore other machine learning methods such as Conditional Random Fields, which takes into account the context of sequential data.

3. Combine machine learning methods with rule-based methods to maximize accuracy.

4. Build a tagged corpora of medical terminologies that can add weight to certain classes.

5. Build a dedicated module for identifying textspeak and equating them with regular terms.

6. Set the module to always run a grid search to find the parameters that will yield the highest accuracy everytime it is trained.

IX. Bibliography

[1] "Doh profile." http://www.doh.gov.ph/profile, Sep 2015.

[2] "Mission and vision." http://www.doh.gov.ph/mission-vision, Sep 2015.

[3] W. H. Organization *et al.*, *The Philippines health system review.* Manila, Philippines: World Health Organization, Western Pacific Region, 2011.

[4] D. o. H. P. World Health Organization, "Philippines health service delivery profile." Web, 2012.

[5] J. Leonardia, H. Prytherch, K. Ronquillo, R. G. Nodora, and A. Ruppel, "Assessment of factors influencing retention in the philippine national rural physician deployment program," *BMC Health Services Research*, vol. 12, no. 1, p. 411, 2012.

[6] C. A. T. Antonio, "Addressing the maldistribution of physicians: Policy options for the up college of medicine," Master's thesis, University of the Philippines College of Medicine, 2008.

[7] "About - national telehealth center." https://telehealth.ph/about/, Sep 2015.

[8] "Program areas - national telehealth center." https://telehealth.ph/about/onehealth/, Sep 2015.

[9] "Infographic: How does telemedicine work in nthc?." https://telehealth.ph/2015/02/18/infographic-how-does-telemedicine-work-in-nthc, Feb 2015.

[10] Z. R. A. Lorraine Carlos Salazar, Shelah Lardizabal-Vallarino, *Digital review of Asia Pacific 2007-2008.* SAGE Publications, 2008.

[11] A. B. B. P. F. A. M. Alex Gavino, Pia Athena Tolentino, "Philippines: A telemedicine program utilizing short message service (sms) for remote village doctors," *Question 14-2/2: Mobile eHealth solutions for Developing Countries*, pp. 94–98, 2010.

[12] R. V. G. Randy Joseph Fernandez, "Design of a ticketing system for health care using web2py." Presentation, 2011.

[13] C. W. Nadkarni PM, Ohno-Machado L, "Natural language processing: an introduction," *Journal of the American Medical Informatics Association*, vol. 18, no. 5, pp. 544–551, 2011.

[14] N. D. Tatyana Shcherbakova, Maria Vergelis, "Spam and phishing in the first quarter of 2015." https://securelist.com/analysis/quarterly-spam-reports/69932/spam-and-phishing-in-the-first-quarter-of-2015/, May 2015.

[15] H. Shirani-Mehr, "Sms spam detection using machine learning approach," tech. rep., Stanford University, 2013.

[16] R. Wootton, *Telehealth in the developing world*. International Development Research Centre Royal Society of Medicine Press, 2009.

[17] J. Dzenowagis, "Bridging the digital divide: linking health and ict policy," *Telehealth in the developing world*, 2009.

[18] F. Ngabo, J. Nguimfack, F. Nwaigwe, C. Mugeni, D. Muhoza, D. R. Wilson, J. Kalach, R. Gakuba, C. Karema, and A. Binagwaho, "Designing and implementing an innovative sms-based alert system (rapidsms-mch) to monitor pregnancy and reduce maternal and child deaths in rwanda," *The Pan African Medical Journal*, vol. 13, 2012.

[19] A. B. Marcelo, "Telemedicine in developing countries: Perspectives from the philippines," *Telehealth in the developing world*, p. 27, 2009.

[20] S. A. Nisperos and Y. Urano, "Sms text messaging: A ubiquitous tool for cheaper medicine in the philippines," in *Advanced Communication Technology (ICACT), 2011 13th International Conference on*, pp. 1415–1419, IEEE, 2011.

[21] C. Tamban, I. T. Isip-Tan, and C. Jimeno, "Use of short message services (SMS) for the management of type 2 diabetes mellitus: A randomized controlled trial," *Journal of the ASEAN Federation of Endocrine Societies*, vol. 28, no. 2, pp. 143–149, 2013.

[22] C. Déglise, L. S. Suggs, and P. Odermatt, "Sms for disease control in developing countries: a systematic review of mobile health applications," *Journal of Telemedicine and Telecare*, vol. 18, no. 5, pp. 273–281, 2012.

[23] P. Melville, V. Chenthamarakshan, R. D. Lawrence, J. Powell, M. Mugisha, S. Sapra, R. Anandan, and S. Assefa, "Amplifying the voice of youth in africa via text analytics," in *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1204–1212, ACM, 2013.

[24] M. S. Sorower, "A literature survey on algorithms for multi-label learning," tech. rep., Oregon State University, 2010.

[25] S. P. Stenner, K. B. Johnson, and J. C. Denny, "Paste: patient-centered sms text tagging in a medication management system," *Journal of the American Medical Informatics Association*, vol. 19, no. 3, pp. 368–374, 2012.

[26] G. Tsoumakas, A. Dimou, E. Spyromitros, V. Mezaris, I. Kompatsiaris, and I. Vlahavas, "Correlation-based pruning of stacked binary relevance models for multi-label learning," in *Proceedings of the 1st International Workshop on Learning from Multi-Label Data*, pp. 101–116, 2009.

[27] W. Bi and J. Kwok, "Efficient multi-label classification with many labels," in *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pp. 405–413, 2013.

[28] A. Alessandro, G. Corani, D. Mauá, and S. Gabaglio, "An ensemble of bayesian networks for multilabel classification," in *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*, pp. 1220–1225, AAAI Press, 2013.

[29] C. Clark, J. Aberdeen, M. Coarr, D. Tresner-Kirsch, B. Wellner, A. Yeh, and L. Hirschman, "Mitre system for clinical assertion status classification," *Journal of the American Medical Informatics Association*, vol. 18, no. 5, pp. 563–567, 2011.

[30] S. N. Kim, D. Martinez, L. Cavedon, and L. Yencken, "Automatic classification of sentences to support evidence based medicine," *BMC bioinformatics*, vol. 12, no. Suppl 2, p. S5, 2011.

[31] S. Gella and D. T. Long, "Automatic sentence classifier using sentence ordering features for event based medicine: Shared task system description," in *In Proceedings of Australasian Language Technology Association Workshop*, pp. 130–133, 2012.

[32] S. Spat, B. Cadonna, I. Rakovac, C. Gutl, H. Leitner, G. Stark, P. Beck, *et al.*, "Multi-label text classification of german language medical documents," 2007.

[33] M. K. Dalal and M. A. Zaveri, "Automatic text classification: a technical review," *International Journal of Computer Applications*, vol. 28, no. 2, pp. 37–40, 2011.

[34] A. K. Uysal and S. Gunal, "The impact of preprocessing on text classification," *Information Processing and Management*, 2014.

[35] Y.-T. Chen and M. C. Chen, "Using chi-square statistics to measure similarities for text categorization," *Expert Systems with Applications*, 2011.

[36] P. Meesad, P. Boonrawd, and V. Nuipian, "A chi-square-test for word importance differentiation in text classification," in *Proceedings of International Conference on Information and Electronics Engineering*, pp. 110–114, 2011.

[37] T. Ahmad, H. Akhtar, A. Chopra, and M. W. Akhtar, "Satire detection from web documents using machine learning methods," in *Soft Computing and Machine Intelligence (ISCMI), 2014 International Conference on*, pp. 102–105, 2014.

[38] K. Lee, D. Palsetia, R. Narayanan, M. M. A. Patwary, A. Agrawal, and A. Choudhary, "Twitter trending topic classification," in *Data Mining Workshops (ICDMW), 2011 IEEE 11th International Conference on*, pp. 251–258, 2011.

[39] J. Huh, M. Yetisgen-Yildiz, and W. Pratt, "Text classification for assisting moderators in online health communities," *Journal of biomedical informatics*, 2013.

[40] A. A. Argaw, A. Hulth, and B. B. Megyesi, "General-purpose text categorization applied to the medical domain," tech. rep., Department of Computer and Systems Sciences, Stockholm University, 2007.

[41] S. Hassan, M. Rafi, and M. S. Shaikh, "Comparing svm and naive bayes classifiers for text categorization with wikitology as knowledge enrichment," in *Multitopic Conference (INMIC), 2011 IEEE 14th International*, pp. 31–34, 2011.

[42] Z. Ju, J. Wang, and F. Zhu, "Named entity recognition from biomedical text using svm," in *Bioinformatics and Biomedical Engineering, (iCBBE) 2011 5th International Conference on*, pp. 1–4, 2011.

[43] Z. Ju, M. Zhou, and F. Zhu, "Identifying biological terms from text by support vector machine," in *2011 6th IEEE Conference on Industrial Electronics and Applications*, pp. 455–458, 2011.

[44] R. L. Figueroa, D. A. Soto, and E. J. Pino, "Identifying and extracting patient smoking status information from clinical narrative texts in spanish," in *2014 36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pp. 2710–2713, 2014.

[45] A. B. Abacha and P. Zweigenbaum, "Identifying biological terms from text by support vector machine," in *Proceedings of BioNLP 2011 Workshop*, pp. 56–64, 2011.

[46] "National telehealth center - history." https://telehealth.ph/history, December 2015.

[47] "National telehealth center - projects." https://telehealth.ph/projects/, December 2015.

[48] A. T. Association, "What is telemedicine." http://www.americantelemed.org/about-telemedicine/what-is-telemedicine, December 2015.

[49] M. Mars, "Telemedicine and advances in urban and rural healthcare delivery in africa," *Progress in cardiovascular diseases*, vol. 56, no. 3, pp. 326–335, 2013.

[50] "What are the available areas for deployment? what are the categories of these areas?." http://www.doh.gov.ph/node/1098, Dec 2015.

[51] "Information: doctors to the barrios under study." http://www.mb.com.ph/information-doctors-to-the-barrios-under-study/, May 2014.

[52] T. M. Mitchell, *Machine Learning*. McGraw-Hill, Inc, 1997.

[53] S. Russell and P. Norvig, *Artificial intelligence: a modern approach.* Pearson, 1995.

[54] W. B. Cavnar, J. M. Trenkle, *et al.*, “N-gram-based text categorization,” *Ann Arbor MI*, 1994.

[55] A. Rajaraman, *Mining of Massive Datasets.* Cambridge University Press, 2011.

[56] C. D. Manning, P. Raghavan, and H. Schutze, *Introduction to Information Retrieval.* Cambridge University Press, 2008.

[57] V. Vryniotis, “Using feature selection methods in text classification.” http://blog.datumbox.com/using-feature-selection-methods-in-text-classification/, January 2014.

[58] S. R. Gunn *et al.*, “Support vector machines for classification and regression,” *ISIS technical report*, 1998.

[59] C.-W. Hsu, C.-C. Chang, C.-J. Lin, *et al.*, “A practical guide to support vector classification,” 2003.

[60] M. D. Pierro, “web2py for scientific applications,” *Computing in Science and Engineering*, vol. 13, no. 2, pp. 64–69, 2011.

[61] 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*, vol. 12, pp. 2825–2830, 2011.

X. Appendix

A. Source Code

source–code/models/db.py

```
# -*- coding: utf-8 -*-

#########################################################################
## This scaffolding model makes your app work on Google App Engine too
## File is released under public domain and you can use without limitations
#########################################################################

if request.global_settings.web2py_version < "2.14.1":
    raise HTTP(500, "Requires web2py 2.13.3 or newer")

## if SSL/HTTPS is properly configured and you want all HTTP requests to
## be redirected to HTTPS, uncomment the line below:
# request.requires_https()

## app configuration made easy. Look inside private/appconfig.ini
from gluon.contrib.appconfig import AppConfig
## once in production, remove reload=True to gain full speed
myconf = AppConfig(reload=True)

if not request.env.web2py_runtime_gae:
    ## if NOT running on Google App Engine use SQLite or other DB
    db = DAL(myconf.get('db.uri'),
             pool_size = myconf.get('db.pool_size'),
             migrate_enabled = myconf.get('db.migrate'),
             check_reserved = ['all'],
             db_codec='UTF-8')
else:
    ## connect to Google BigTable (optional 'google:datastore://namespace')
    db = DAL('google:datastore+ndb')
    ## store sessions and tickets there
    session.connect(request, response, db=db)
    ## or store session in Memcache, Redis, etc.
    ## from gluon.contrib.memdb import MEMDB
    ## from google.appengine.api.memcache import Client
    ## session.connect(request, response, db = MEMDB(Client()))

## by default give a view/generic.extension to all actions from localhost
## none otherwise. a pattern can be 'controller/function.extension'
response.generic_patterns = ['*'] if request.is_local else []
## choose a style for forms
response.formstyle = myconf.get('forms.formstyle')  # or 'bootstrap3_stacked' or 'bootstrap2' or
     other
response.form_label_separator = myconf.get('forms.separator') or ''


## (optional) optimize handling of static files
# response.optimize_css = 'concat,minify,inline'
# response.optimize_js = 'concat,minify,inline'
## (optional) static assets folder versioning
# response.static_version = '0.0.0'
#########################################################################
## Here is sample code if you need for
## - email capabilities
## - authentication (registration, login, logout, ... )
## - authorization (role based authorization)
## - services (xml, csv, json, xmlrpc, jsonrpc, amf, rss)
## - old style crud actions
## (more options discussed in gluon/tools.py)
#########################################################################

from gluon.tools import Auth, Service, PluginManager

# host names must be a list of allowed host names (glob syntax allowed)
auth = Auth(db, host_names=myconf.get('host.names'))
service = Service()
plugins = PluginManager()

## create all tables needed by auth if not custom tables
auth.define_tables(username=True, signature=False)
db.auth_user.email.readable = db.auth_user.email.writable = False
db.auth_user.password.requires = [IS_LENGTH(minsize=8, error_message='Password must be at least 8
     characters in length'), CRYPT(auth.settings.hmac_key)]
auth.messages.invalid_login = 'Invalid username/password combination'

auth.settings.change_password_next = URL('default','user',args='change_password')
auth.settings.profile_next = URL('default','user',args='profile')

## configure auth policy
```

```
auth.settings.create_user_groups = None
auth.settings.registration_requires_verification = False
auth.settings.registration_requires_approval = False
auth.settings.reset_password_requires_verification = True
auth.settings.remember_me_form = False

#########################################################################
## Define your tables below (or better in another model file) for example
##
## >>> db.define_table('mytable',Field('myfield','string'))
##
## Fields can be 'string','text','password','integer','double','boolean'
##       'date','time','datetime','blob','upload', 'reference TABLENAME'
## There is an implicit 'id integer autoincrement' field
## Consult manual for more options, validators, etc.
##
## More API examples for controllers:
##
## >>> db.mytable.insert(myfield='value')
## >>> rows=db(db.mytable.myfield=='value').select(db.mytable.ALL)
## >>> for row in rows: print row.id, row.myfield
#########################################################################

## after defining tables, uncomment below to enable auditing
# auth.enable_record_versioning(db)

MESSAGE_TAGS = ('Unassigned',
                'Dermatology',
                                'General Question',
                                'Internal Medicine',
                                'Medico Legal',
                                'Neurology',
                                'Obgyn',
                                'Ophthalmology',
                                'Otorhinolaryngology',
                                'Pediatrics',
                                'Radiology',
                                'Surgery',
                                )

db.define_table('t_message',
                Field('msg_time', 'datetime', represent=lambda x, row: x.strftime("%b %d, %Y %I:%M
                     %p"), label='Received'),
                Field('msg_sender', 'string', notnull=True, label='From'),
                Field('msg_content', 'text', notnull=True, label='Message'),
                Field('msg_tag', 'string', requires=IS_IN_SET(MESSAGE_TAGS, error_message='Select
                    a value'), label='Tag'),
                Field('is_verified', 'boolean'),
                Field('is_spam', 'boolean'))

db.define_table('t_filter', Field('msg_sender','string',required=True,unique=True, label='Sender
    Name'))

db.define_table('t_model',
                Field('mdl_time','datetime', represent=lambda x, row: x.strftime("%b %d, %Y %I:%M
                    %p"), label='Date Created'),
                Field('mdl_name','string',required=True,unique=True, label='Model Name'),
                Field('mdl_percent','integer',requires = IS_INT_IN_RANGE(1, 100, error_message='
                    Out of range! 1-100 only'), label='Select Percentile'),
                Field('mdl_ngrams','integer',requires = IS_INT_IN_RANGE(1, 10, error_message='Out
                    of range! 1-10 only'), label='Select n-grams'),
                Field('is_deployed','boolean',default=False))

from tagr import TAGR
tagr = cache.ram('c_tagr', lambda: TAGR(), time_expire=None)
```

source–code/models/menu.py

```
# -*- coding: utf-8 -*-
# this file is released under public domain and you can use without limitations

#########################################################################
## Customize your APP title, subtitle and menus here
#########################################################################

response.logo = A(B('web',SPAN(2),'py'),XML('&trade;&nbsp;'),
                  _class="navbar-brand",_href="http://www.web2py.com/",
                  _id="web2py-logo")
response.title = request.application.replace('_',' ').title()
response.subtitle = ''

## read more at http://dev.w3.org/html5/markup/meta.name.html
response.meta.author = myconf.get('app.author')
response.meta.description = myconf.get('app.description')
response.meta.keywords = myconf.get('app.keywords')
response.meta.generator = myconf.get('app.generator')

#########################################################################
## this is the main application menu add/remove items as required
#########################################################################
```

```
if auth.is_logged_in():
    response.menu += [
                        (CAT(I(_class='glyphicon glyphicon-user'), ' '+auth.user.first_name+' '+
                            auth.user.last_name), False, URL('default', 'index', args=['login']),
                            []),
                        (CAT(I(_class='glyphicon glyphicon-envelope'), ' Messages'), False, URL('
                            messages', 'index'), []),
                        (CAT(I(_class='glyphicon glyphicon-trash'), ' Filtered'), False, URL('
                            filtered', 'index'), [])
                    ]
    if auth.has_membership('Administrator'):
        response.menu += [
                            (CAT(I(_class='glyphicon glyphicon-filter'), ' Filter List'), False,
                                URL('filter_settings', 'index'), []),
                            (CAT(I(_class='glyphicon glyphicon-cog'), ' TAGR Settings'), False,
                                URL('tagr_settings', 'index'), []),
                        ]
    response.menu += [(CAT(I(_class='glyphicon glyphicon-log-out'), ' Log Out'), False, URL('
        default', 'index', args=['logout']), []),]

if "auth" in locals(): auth.wikimenu()
```

source–code/controllers/default.py

```
# -*- coding: utf-8 -*-
# this file is released under public domain and you can use without limitations

#########################################################################
## This is a sample controller
## - index is the default action of any application
## - user is required for authentication and authorization
## - download is for downloading files uploaded in the db (does streaming)
#########################################################################

response.title='National Telehealth System'

def index():
    if auth.has_membership('Telehealth Nurse'):
        response.currentrole = 'Nurse'
    elif auth.has_membership('Administrator'):
        response.currentrole = 'Admin'
    else:
        response.currentrole = 'User'
    pass
    return dict(form=auth())

def user():
    """
    exposes:
    http://..../[app]/default/user/login
    http://..../[app]/default/user/logout
    http://..../[app]/default/user/register
    http://..../[app]/default/user/profile
    http://..../[app]/default/user/retrieve_password
    http://..../[app]/default/user/change_password
    http://..../[app]/default/user/bulk_register
    use @auth.requires_login()
        @auth.requires_membership('group name')
        @auth.requires_permission('read','table name',record_id)
    to decorate functions that need access control
    also notice there is http://..../[app]/appadmin/manage/auth to allow administrator to manage
        users
    """
    """
    example action using the internationalization operator T and flash
    rendered by views/default/index.html or views/generic.html

    if you need a simple wiki simply replace the two lines below with:
    return auth.wiki()
    """
    if request.args(0) == 'not_authorized':
        redirect(URL('default', 'not_authorized.html'))
    elif request.args(0) == 'login':
        redirect(URL('default', 'index', args='login'))
    else:
        return dict(form=auth())

def not_authorized():
    return dict()

@cache.action()
def download():
    """
    allows downloading of uploaded files
    http://..../[app]/default/download/[filename]
    """
    return response.download(request, db)

def call():
    """
    exposes services. for example:
    http://..../[app]/default/call/jsonrpc
    decorate with @services.jsonrpc the functions to expose
    supports xml, json, xmlrpc, jsonrpc, amfrpc, rss, csv
    """
    return service()
```

source–code/controllers/messages.py

```
response.title='National Telehealth System'

@auth.requires_login()
def index():
    response.title += ' | Messages'
    posts_per_page = 20
    display_links = 5

    total_posts = db(db.t_message.is_spam == False).count()
    total_pages = total_posts / posts_per_page
    if (total_posts % posts_per_page != 0): total_pages += 1

    if not request.vars.page or int(request.vars.page) < 1:
            redirect(URL(vars={'page':1}))
    elif int(request.vars.page) > total_pages:
        redirect(URL(vars={'page':total_pages}))
    else:
        current_page = int(request.vars.page)

    start_row = (current_page-1)*posts_per_page
    end_row = current_page*posts_per_page
    messages = db(db.t_message.is_spam == False).select(orderby=~db.t_message.msg_time, limitby=(
        start_row,end_row))

    if total_pages == 1:
        pagination = None
    else:
        if current_page - display_links > 0:
            start_link = current_page - display_links
        else:
            start_link = 1

        if current_page + display_links < total_pages:
            end_link = current_page + display_links
        else:
            end_link = total_pages

        pagination = []

        if start_link > 1:
            pagination.append(LI(A("1", _href=URL(index, vars=dict(page='1')))))
            pagination.append(LI(SPAN("..."), _class="disabled"))

        for i in range(start_link, end_link + 1):
            if current_page == i:
                pagination.append(LI(A(str(i), _href=URL(index, vars=dict(page=str(i)))), _class="
                    active"))
            else:
                pagination.append(LI(A(str(i), _href=URL(index, vars=dict(page=str(i))))))

        if end_link < total_pages:
            pagination.append(LI(SPAN("..."), _class="disabled"))
            pagination.append(LI(A(str(total_pages), _href=URL(index, vars=dict(page=str(
                total_pages))))))

    return dict(messages=messages, pagination=pagination)

@auth.requires_signature()
@auth.requires_login()
def mark():
    messages = db(db.t_message.id == request.vars.mark_id).select()
    form = FORM(INPUT(_type='submit', _value='Yes'),
                INPUT(_type='button', _value='No', _onclick='window.location.href="' + URL('index.
                    html',vars={'page':request.vars.page}) + '";return false;')
               )
    if form.process().accepted:
        db(db.t_message.id == request.vars.mark_id).update(is_spam=True)
        response.flash = None
        redirect(URL('index.html',vars={'page':request.vars.page}), client_side=True)
    return dict(form=form, messages=messages)

@auth.requires_signature()
@auth.requires_login()
def remove():
    messages = db(db.t_message.id == request.vars.remove_id).select()
    form = FORM(INPUT(_type='submit', _value='Yes'),
                INPUT(_type='button', _value='No', _onclick='window.location.href="' + URL('index.
                    html',vars={'page':request.vars.page}) + '";return false;')
               )
    if form.process().accepted:
        db(db.t_message.id == request.vars.remove_id).update(is_verified=False)
        response.flash = None
        redirect(URL('index.html',vars={'page':request.vars.page}), client_side=True)
    return dict(form=form, messages=messages)

def verify_processing(form):
    if form.vars.is_verified=='on' and form.vars.msg_tag=='Unassigned':
        form.errors.msg_tag = 'Cannot verify a message tagged as Unassigned'

@auth.requires_signature()
@auth.requires_login()
def verify():
    verify_id = request.vars.verify_id
    db.t_message.msg_time.writable = False
    db.t_message.msg_sender.writable = False
    db.t_message.msg_content.writable = False
    db.t_message.is_spam.readable = db.t_message.is_spam.writable = False
    form = SQLFORM(table = db.t_message,
                   record = verify_id,
                   formstyle = "table3cols",
                   showid = False,
                    labels = {'is_verified':'Check to verify'})
    if form.process(onvalidation=verify_processing).accepted:
        response.flash = None
        redirect(URL('index.html',vars={'page':request.vars.page}), client_side=True)
    return dict(form=form)
```

source–code/controllers/filtered.py

```
response.title='National Telehealth System'

@auth.requires_login()
def index():
    response.title += ' | Filtered Messages'
    posts_per_page = 20
    display_links = 5

    total_posts = db(db.t_message.is_spam == True).count()
    total_pages = total_posts / posts_per_page
    if (total_posts % posts_per_page != 0): total_pages += 1

    if not request.vars.page or int(request.vars.page) < 1:
        redirect(URL(vars={'page':1}))
    elif int(request.vars.page) > total_pages:
        redirect(URL(vars={'page':total_pages}))
    else:
        current_page = int(request.vars.page)

    start_row = (current_page-1)*posts_per_page
    end_row = current_page*posts_per_page
    messages = db(db.t_message.is_spam == True).select(orderby=~db.t_message.msg_time, limitby=(
        start_row,end_row))

    if total_pages == 1:
        pagination = None
    else:
        if current_page - display_links > 0:
            start_link = current_page - display_links
        else:
            start_link = 1

        if current_page + display_links < total_pages:
            end_link = current_page + display_links
        else:
            end_link = total_pages

        pagination = []

        if start_link > 1:
            pagination.append(LI(A("1", _href=URL(index, vars=dict(page='1')))))
            pagination.append(LI(SPAN("..."), _class="disabled"))

        for i in range(start_link, end_link + 1):
            if current_page == i:
                pagination.append(LI(A(str(i), _href=URL(index, vars=dict(page=str(i)))), _class="
                    active"))
            else:
                pagination.append(LI(A(str(i), _href=URL(index, vars=dict(page=str(i))))))

        if end_link < total_pages:
            pagination.append(LI(SPAN("..."), _class="disabled"))
            pagination.append(LI(A(str(total_pages), _href=URL(index, vars=dict(page=str(
                total_pages))))))

    return dict(messages=messages, pagination=pagination)

@auth.requires_signature()
@auth.requires_login()
def unmark():
    messages = db(db.t_message.id == request.vars.unmark_id).select()
    form = FORM(INPUT(_type='submit', _value='Yes'),
                INPUT(_type='button', _value='No', _onclick='location.href="' + URL('index.html',
                     vars={'page':request.vars.page}) + '";return false;')
                )
    if form.process().accepted:
        db(db.t_message.id == request.vars.unmark_id).update(is_spam=False)
        response.flash = None
        redirect(URL('index.html',vars={'page':request.vars.page}), client_side=True)
    return dict(form=form, messages=messages)
```

source–code/controllers/filter_settings.py

```
response.title='National Telehealth System'

@auth.requires_membership('Administrator')
def index():
    response.title += ' | Filter Settings'
    return dict(message="hello from filter_settings.py")

@auth.requires_membership('Administrator')
def blacklist():
    db.t_filter.id.readable = False
    grid = SQLFORM.grid(db.t_filter,
                        fields=db.t_filter,
                        sortable=False,
                        details=False,
                        searchable=False,
                        deletable=True,
                        editable=False,
                        csv=False,
                        formstyle = 'table3cols',
                        maxtextlength=256)
    grid.element('.web2py_counter', replace=None)
    return dict(grid=grid)
```

source–code/controllers/tagr_settings.py

```
import os
response.title='National Telehealth System'

@auth.requires_membership('Administrator')
def index():
    response.title += ' | TAGR'
    return dict()

@auth.requires_membership('Administrator')
def train():
    response.title += ' | Train a new model'
    form = FORM(TABLE(TR(TD(LABEL('Extract n-grams:',_for='ngrams'),_class='w2p_fl'),
                         TD(INPUT(_type='number',_name='ngrams',_min='1',_max='10',requires=
                              IS_NOT_EMPTY()),_class='w2p_fw')),
                      TR(TD(LABEL('Select percentile:',_for='percentile'),_class='w2p_fl'),
                         TD(INPUT(_type='number',_name='percentile',_min='1',_max='100',requires=
                              IS_NOT_EMPTY()),_class='w2p_fw')),
                      TR(TD(_class='w2p_fl'),TD(INPUT(_type='submit', _value='Train'),_class='
                           w2p_fw')))).process()
    return dict(form=form)

def train_results():
    data=__get_data()
    data_messages = data['messages']
    data_labels = data['labels']

    tagr = cache.ram.storage['c_tagr'][1]
    tagr.initialize(int(request.vars.ngrams),int(request.vars.percentile))
    data_prep = tagr.prepare_data(data_messages, data_labels)
    report = tagr.evaluate(data_prep, data_labels)
    data_prep = None
    data = None
    return dict(report=report)

def test():
    form = FORM(TABLE(
                    TR(TD(LABEL('Predicted Tag:'),_class='w2p_fl'),TD(request.vars.result,_class='
                         w2p_fw')),
                    TR(TD(LABEL('Message Content:',_for='message_content'),_class='w2p_fl'),TD(
                         TEXTAREA(_name='message_content',requires=IS_NOT_EMPTY()),_class='w2p_fw'))
                         ,
                    TR(TD(_class='w2p_fl'),TD(INPUT(_type='submit', _value='Classify'),_class='
                         w2p_fw')))
                )

    data=__get_data()
    data_messages = data['messages']
    data_labels = data['labels']

    tagr = cache.ram.storage['c_tagr'][1]
    data_prep = tagr.prepare_data(data_messages, data_labels)
    tagr.train_on_whole(data_prep, data_labels)
    data_prep = None
    data = None

    if form.process().accepted:
        result = tagr.classify(request.vars.message_content)
        redirect(URL('tagr_settings','test_load',vars={'result':result}))
    return dict(form=form)

def save():
    form = FORM(TABLE(
                            TR(TD(LABEL('Model name:',_for='model_name'),_class='w2p_fl'),TD(INPUT(
                                _type='text',_name='model_name',
                            requires=[IS_NOT_EMPTY(error_message='Cannot be empty'),
                          IS_NOT_IN_DB(db, 't_model.mdl_name', error_message='Name already exists')
                                ,
                                                       IS_ALPHANUMERIC(error_message='Name can only
                                                            contain alphanumeric characters')]),_class='
                                                            w2p_fw')),
                            TR(TD(_class='w2p_fl'),TD(INPUT(_type='submit', _value='Save'),_class='
                                w2p_fw')))
                    )
    if form.process(message_onsuccess='Model successfully saved!').accepted:
        tagr = cache.ram.storage['c_tagr'][1]
        tagr.save(os.path.join(request.folder,'private',form.vars.model_name))
        db.t_model.insert(mdl_time=request.now,mdl_name=form.vars.model_name,mdl_ngrams=request.
             vars.ngrams,mdl_percent=request.vars.percentile)
    return dict(form=form)

@auth.requires_membership('Administrator')
def load():
    response.title += ' | Load a trained model'
    mdl = db(db.t_model).select(orderby=~db.t_model.mdl_time)
    form = FORM(TABLE(TR(TD(LABEL('Select model:',_for='model'),_class='w2p_fl'),
                          TD(SELECT(_name='model_id',*[OPTION(mdl[i].mdl_name, _value=str(mdl[i].id
                                )) for i in range(len(mdl))]),_class='w2p_fw')),
                      TR(TD(_class='w2p_fl'),TD(INPUT(_type='submit', _value='Load'),_class='w2p_fw
                          ')))).process()
    return dict(form=form, mdl=mdl)

def load_results():
    data=__get_data()
    data_messages = data['messages']
    data_labels = data['labels']

    tagr = cache.ram.storage['c_tagr'][1]
    tagr.load(os.path.join(request.folder,'private',request.vars.mdl_name))
    data_prep = tagr.prepare_data(data_messages, data_labels)
    report = tagr.evaluate(data_prep, data_labels)
    data_prep = None
    data = None
    return dict(report=report)

@auth.requires_membership('Administrator')
def choose():
    response.title += ' | Choose a model to use'
    mdl = db(db.t_model).select(orderby=~db.t_model.mdl_time)
    form = FORM(TABLE(TR(TD(LABEL('Select model:',_for='model'),_class='w2p_fl'),
                          TD(SELECT(_name='model_id',*[OPTION(mdl[i].mdl_name, _value=str(mdl[i].id
                                )) for i in range(len(mdl))]),_class='w2p_fw')),
                      TR(TD(_class='w2p_fl'),TD(INPUT(_type='submit', _value='Choose'),_class='
                          w2p_fw'))))
    if form.process().accepted:
        for row in mdl:
            if row.id==int(request.vars.model_id):
                db(db.t_model.id == row.id).update(is_deployed=True)
            else:
                db(db.t_model.id == row.id).update(is_deployed=False)
        redirect(URL())
    return dict(form=form, mdl=mdl)

def __get_data():
    rows = db(db.t_message.is_verified==True).select(db.t_message.msg_tag, db.t_message.
         msg_content)
    data_messages = []
    data_labels = []
    for row in rows:
        data_messages.append(row.msg_content)
        data_labels.append(row.msg_tag)
    return dict(messages=data_messages, labels=data_labels)
```

source–code/controllers/input.py

```
from tagr import TAGR
import os

response.title='National Telehealth System'

def index():
    form = FORM(TABLE(
                    TR(TD(LABEL('Message Sender:',_for='message_sender'),_class='w2p_fl'),TD(INPUT(
                        _type='text',_name='message_sender',requires=IS_NOT_EMPTY()),_class='w2p_fw
                        ')),
                    TR(TD(LABEL('Message Content:',_for='message_content'),_class='w2p_fl'),TD(
                        TEXTAREA(_name='message_content',requires=IS_NOT_EMPTY()),_class='w2p_fw'))
                        ,
                    TR(TD(_class='w2p_fl'),TD(INPUT(_type='submit', _value='Submit'),_class='w2p_fw
                        ')))
                )
    if form.process().accepted:
        if is_spam(request.vars.message_sender):
            db.t_message.insert(msg_time=request.now,
                                msg_sender=request.vars.message_sender,
                                msg_content=request.vars.message_content,
                                msg_tag='Unassigned',
                                is_verified=False,
                                is_spam=True)
            response.flash = 'Sent to Filtered Messages folder'
        elif clf_is_online():
            db.t_message.insert(msg_time=request.now,
                                msg_sender=request.vars.message_sender,
                                msg_content=request.vars.message_content,
                                msg_tag=classify(request.vars.message_content),
                                is_verified=False,
                                is_spam=False)
            response.flash = 'Tagged and sent to Messages folder'
        else:
            db.t_message.insert(msg_time=request.now,
                                msg_sender=request.vars.message_sender,
                                msg_content=request.vars.message_content,
                                msg_tag='Unassigned',
                                is_verified=False,
                                is_spam=False)
            response.flash = 'Unassigned; sent to Messages folder'
    return dict(form=form)

def is_spam(sender):
    rows = db(db.t_filter).select(db.t_filter.msg_sender)
    filter = []
    for row in rows:
        filter.append(row.msg_sender)
    return sender in filter

def clf_is_online():
    rows = db(db.t_model).select(db.t_model.is_deployed)
    status = []
    for row in rows:
        status.append(row.is_deployed)
    return (True in status)

def classify(message):
    for row in db(db.t_model.is_deployed == True).select():
        model = row.mdl_name
    clf = TAGR()
    clf.load(os.path.join(request.folder,'private',model))
    return clf.classify(message)
```

source–code/views/layout.html

```
<!DOCTYPE html>
<html class="no-js" lang="{{=T.accepted_language or 'en'}}">
    <head>
        <title>{{=response.title or request.application}}</title>
        <link rel="stylesheet" href="{{=URL('static','css/bootstrap.min.css')}}"/>
        <link rel="stylesheet" href="{{=URL('static','css/web2py-bootstrap3.css')}}"/>
        <link rel="stylesheet" href="{{=URL('static','css/tagr-style.css')}}"/>
        <link rel="shortcut icon" href="{{=URL('static','images/favicon.ico')}}" type="image/x-
            icon">
        <link rel="apple-touch-icon" href="{{=URL('static','images/favicon.png')}}">
        <!-- All JavaScript at the bottom, except for Modernizr which enables
        HTML5 elements & feature detects -->
        <script src="{{=URL('static','js/modernizr-2.8.3.min.js')}}"></script>
        <!--[if lt IE 9]>
        <script src="{{=URL('static','js/respond-1.4.2.min.js')}}"></script>
        <![endif]-->
        {{include 'web2py_ajax.html'}}
        {{block head}}{{end}}
        {{
        # using sidebars need to know what sidebar you want to use
        mc0 = 'col-md-12'
        mc1 = 'col-md-9'
        mc2 = 'col-md-6'
        left_sidebar_enabled = globals().get('left_sidebar_enabled', False)
        right_sidebar_enabled = globals().get('right_sidebar_enabled', False)
        middle_column = {0: mc0, 1: mc1, 2: mc2}[
        (left_sidebar_enabled and 1 or 0)+(right_sidebar_enabled and 1 or 0)]
        }}
    </head>
    <body>
        <div class="w2p_flash alert alert-dismissable">{{=response.flash or ''}}</div>
        <header class="container-fluid background">
            <h1>National Telehealth System</h1>
            <img src="{{=URL('static','images/nthc.png')}}">
            <img src="{{=URL('static','images/nih.png')}}">
            <img src="{{=URL('static','images/upm.png')}}">
            <img src="{{=URL('static','images/doh.png')}}">
            <img src="{{=URL('static','images/dost.png')}}">
        </header>
        <nav class="hlist">
            {{if response.menu:}}{{=MENU(response.menu, _class=None,li_class=None,li_first=None,
                li_last='logout',li_active='active',active_url=request.url)}}{{pass}}
        </nav>
        <div class="container-fluid main-container">
            {{if left_sidebar_enabled:}}
            <div class="col-md-3 left-sidebar">
                {{block left_sidebar}}

                {{end}}
            </div>
            {{pass}}
            <div class="{{=middle_column}}">
            {{block center}}
            {{include}}
            {{end}}
            </div>
            {{if right_sidebar_enabled:}}
            <div class="col-md-3 right-sidebar">
                {{block right_sidebar}}

                {{end}}
            </div>
            {{pass}}
        </div>
        {{block footer}}
        <footer class="footer">
            <div class="container-fluid">
                <div class="copyright pull-left">{{=T('Copyright')}} &#169; {{=request.now.year
                    }}</div>
            </div>
        </footer>
        <script src="{{=URL('static','js/bootstrap.min.js')}}"></script>
        <script src="{{=URL('static','js/web2py-bootstrap3.js')}}"></script>
        {{block page_js}}{{end page_js}}
    </body>
</html>
```

source–code/views/default/index.html

```
{{left_sidebar_enabled = True}}
{{right_sidebar_enabled = False}}

{{extend 'layout.html'}}

{{block left_sidebar}}
{{if auth.is_logged_in():}}
    {{include 'default/welcome.html'}}
  {{else:}}
    {{include 'default/userform.html'}}
  {{pass}}
{{end}}

{{block center}}

<h2>About</h2>
The <strong>National Telehealth Service Program (NTSP)</strong>, a joint project of the Department
      of Health and the National Telehealth Center, is a 5-year program aimed at expanding
     Telemedicine in 4th to 6th class municipalities nationwide. NTSP facilitates consults between
     primary care physicians in geographically isolated and disadvantaged areas (GIDA) and clinical
      specialists of the Philippine General Hospital (PGH) using a mobile and internet-based
     interface and triaging system.

<h3>National Telehealth Center</h3>
NTHC's commitment is to engage people to use available technologies to improve health care albeit
     distance barriers. Since its conception, it continues to develop telehealth applicants derived
      from people's own problem-solving contributions. Through research-cum-service activities, the
      center helps both patients and health care providers maximize widely available and cost-
     effective ICT tools to improve delivery of healthcare.

<h3>Contact</h3>
<strong>Address:</strong> 3rd Floor, Information Technology Complex, Philippine General Hospital,
     Taft Avenue, Manila 1000 Philippines<br>
<strong>Phone:</strong> (02) 509-1003 or (0927) 986-3703<br>
<strong>Email:</strong> <a href='mailto:admin@telehealth.ph'>admin@telehealth.ph</a><br>
<strong>Website:</strong> <a target = '_blank' href='http://www.telehealth.ph'>www.telehealth.ph</
     a><br>

{{end}}

{{block right_sidebar}}
```

```
{{end}}

{{block page_js}}
<script>
    jQuery("#web2py_user_form input:visible:enabled:first").focus();
{{if request.args(0)=='register':}}
    web2py_validate_entropy(jQuery('#auth_user_password'),100);
{{elif request.args(0)=='change_password':}}
    web2py_validate_entropy(jQuery('#no_table_new_password'),100);
{{pass}}
</script>
{{end page_js}}
```

source–code/views/default/not_authorized.html

```
{{extend 'layout.html'}}
<h2>Forbidden</h2>
You are not authorized to access that page.
```

source–code/views/default/welcome.html

```
<h3>Welcome, {{=auth.user.first_name+' '+auth.user.last_name}}!</h3>
        Your username is: <strong>{{=auth.user.username}}</strong><br>
        Your account role is: <strong>{{=response.currentrole}}</strong><br>
    <div>
        <h3>From here you can:</h3>
        <ul>
            <li><a href={{=URL('default', 'user', args=['profile'])}}>Edit your account</a></li>
            <li><a href={{=URL('default', 'user', args=['change_password'])}}>Change your password
                </a></li>
        </ul>
    </div>
```

source–code/views/default/userform.html

```
<h2>{{='Log In' if request.args(0) == 'login' else T(request.args(0).replace('_',' ').title())}}</
    h2>
<div class="container">
    <div class="row">
        {{if request.args(0) == 'login':}}
        <div id="web2py_user_form" class="col-md-3">
            {{=form}}
        </div>
        {{else:}}
        <div id="web2py_user_form" class="col-lg-4">
            {{=form}}
        </div>
        {{pass}}
    </div>
</div>
```

source–code/views/default/user.html

```
{{extend 'layout.html'}}
{{include 'default/userform.html'}}
```

source–code/views/messages/index.html

```
{{extend 'layout.html'}}

    {{if request.vars.mark_id:}}
        {{=LOAD('messages','mark.load',vars={'mark_id':int(request.vars.mark_id),'page':int(
             request.vars.page)},ajax=True,user_signature=True)}}
    {{pass}}
    {{if request.vars.remove_id:}}
        {{=LOAD('messages','remove.load',vars={'remove_id':int(request.vars.remove_id),'page':int(
             request.vars.page)},ajax=True,user_signature=True)}}
    {{pass}}
    {{if request.vars.verify_id:}}
        {{=LOAD('messages','verify.load',vars={'verify_id':int(request.vars.verify_id),'page':int(
             request.vars.page)},ajax=True,user_signature=True)}}
    {{pass}}
    {{if pagination:}}
    <h2>Messages</h2>
    <div class="text-center">
        <ul class="pagination">
            <li><a href="{{=URL(vars={'page':int(request.vars.page)-1})}}"><<</a></li>
            {{for link in pagination:}}
            {{=link}}
            {{pass}}
            <li><a href="{{=URL(vars={'page':int(request.vars.page)+1})}}">>></a></li>
        </ul>
    </div>
    {{pass}}
    <table class="table table-bordered table-striped table-condensed">
        <thead>
            <tr>
                <th>Received</th>
                <th>From</th>
                <th>Message</th>
                <th>Tag</th>
                <th>Actions</th>
            </tr>
        </thead>
        <tbody>
        {{for row in messages:}}
            <tr>
                <td>{{=row.msg_time.strftime("%b %d, %Y %I:%M %p")}}</td>
                <td>{{=row.msg_sender}}</td>
                <td>{{=row.msg_content}}</td>

            {{if row.is_verified:}}
                <td><i class="glyphicon glyphicon-ok"></i> {{=row.msg_tag}}</a></td>
                <td><a href="{{=URL(vars={'page':int(request.vars.page),'remove_id':row.id})}}"
                     title="Remove verification"><i class="glyphicon glyphicon-remove"></i></a></td
                     >
            {{else:}}
                <td><i class="glyphicon glyphicon-exclamation-sign"></i> <a href="{{=URL(vars={'
                    page':int(request.vars.page),'verify_id':row.id})}}" title="Edit and verify
                    ">{{=row.msg_tag}}</a></td>
                <td><a href="{{=URL(vars={'page':int(request.vars.page),'mark_id':row.id})}}"
                     title="Mark as spam"><i class="glyphicon glyphicon-ban-circle"></i></a></td>

            {{pass}}
            </tr>
        {{pass}}
        </tbody>
    </table>
    {{if pagination:}}
    <div class="text-center">
        <ul class="pagination">
            <li><a href="{{=URL(vars={'page':int(request.vars.page)-1})}}"><<</a></li>
            {{for link in pagination:}}
            {{=link}}
            {{pass}}
            <li><a href="{{=URL(vars={'page':int(request.vars.page)+1})}}">>></a></li>
        </ul>
    </div>
    {{pass}}
```

source–code/views/messages/mark.load

```
<div class="container">
    <h3>Are you sure you want to mark the following as spam?</h3>
    <h4>It will be moved to the Filtered Messages folder.</h4>
    <div class="row">
        <div id="web2py_user_form" class="col-sm-9">
            <table>
                {{for row in messages:}}
                <tr>
                    <td class="w2p_fl">
                        <label class="readonly">Received</label>
                    </td>
                    <td class="w2p_fw">{{=row.msg_time}}</td>
                    <td class="w2p_fc"></td>
                </tr>
                <tr>
                    <td class="w2p_fl">
                        <label class="readonly">From</label>
                    </td>
                    <td class="w2p_fw">{{=row.msg_sender}}</td>
                    <td class="w2p_fc"></td>
                </tr>
                <tr>
                    <td class="w2p_fl">
                        <label class="readonly">Message</label>
                    </td>
                    <td class="w2p_fw">{{=row.msg_content}}</td>
                    <td class="w2p_fc"></td>
                </tr>
                <tr>
                    <td class="w2p_fl">
                        <label class="readonly">Tag</label>
                    </td>
                    <td class="w2p_fw">{{=row.msg_tag}}</td>
                    <td class="w2p_fc"></td>
                </tr>
                <tr>
                    <td class="w2p_fl"></td>
                    <td class="w2p_fw">
                        {{=form}}
                    </td>
                    <td class="w2p_fc"></td>
                </tr>
                {{pass}}
            </table>
        </div>
    </div>
</div>
```

source–code/views/messages/remove.load

```
<div class="container">
    <h3>Are you sure you want to remove the following message's verified status?</h3>
    <div class="row">
        <div id="web2py_user_form" class="col-sm-9">
            <table>
                {{for row in messages:}}
                <tr>
                    <td class="w2p_fl">
                        <label class="readonly">Received</label>
                    </td>
                    <td class="w2p_fw">{{=row.msg_time}}</td>
                    <td class="w2p_fc"></td>
                </tr>
                <tr>
                    <td class="w2p_fl">
                        <label class="readonly">From</label>
                    </td>
                    <td class="w2p_fw">{{=row.msg_sender}}</td>
                    <td class="w2p_fc"></td>
                </tr>
                <tr>
                    <td class="w2p_fl">
                        <label class="readonly">Message</label>
                    </td>
                    <td class="w2p_fw">{{=row.msg_content}}</td>
                    <td class="w2p_fc"></td>
                </tr>
                <tr>
                    <td class="w2p_fl">
                        <label class="readonly">Tag</label>
                    </td>
                    <td class="w2p_fw">{{=row.msg_tag}}</td>
                    <td class="w2p_fc"></td>
                </tr>
                <tr>
                    <td class="w2p_fl"></td>
                    <td class="w2p_fw">
                        {{=form}}
                    </td>
                    <td class="w2p_fc"></td>
                </tr>
                {{pass}}
            </table>
        </div>
    </div>
</div>
```

source–code/views/messages/verify.load

```
<div class="container">
    <h3>Verify message</h3>
    <div class="row">
        <div id="web2py_user_form" class="col-sm-9">
            {{=form}}
        </div>
    </div>
</div>
```

source–code/views/filtered/index.html

```
{{extend 'layout.html'}}
    {{if request.vars.unmark_id:}}
        {{=LOAD('filtered','unmark.load',vars={'unmark_id':int(request.vars.unmark_id),'page':int(
            request.vars.page)},ajax=True,user_signature=True)}}
    {{pass}}
    {{if pagination:}}
    <h2>Filtered Messages</h2>
    <div class="text-center">
        <ul class="pagination">
            <li><a href="{{=URL(vars={'page':int(request.vars.page)-1})}}"><<</a></li>
            {{for link in pagination:}}
            {{=link}}
            {{pass}}
            <li><a href="{{=URL(vars={'page':int(request.vars.page)+1})}}">>></a></li>
        </ul>
    </div>
    {{pass}}
    <table class="table table-bordered table-striped table-condensed">
```

```
        <thead>
            <tr>
                <th>Received</th>
                <th>From</th>
                <th>Message</th>
                <th>Actions</th>
            </tr>
        </thead>
        <tbody>
        {{for row in messages:}}
            <tr>
                <td>{{=row.msg_time.strftime("%b %d, %Y %I:%M %p")}}</td>
                <td>{{=row.msg_sender}}</td>
                <td>{{=row.msg_content}}</td>
                <td>
                    <a href="{{=URL(vars={'page':request.vars.page,'unmark_id':row.id})}}" title="
                         Remove from Filtered Messages"><i class="glyphicon glyphicon glyphicon-
                         repeat"></i></a>
                </td>
            </tr>
        {{pass}}
        </tbody>
    </table>
    {{if pagination:}}
    <div class="text-center">
        <ul class="pagination">
            <li><a href="{{=URL(vars={'page':int(request.vars.page)-1})}}"><<</a></li>
            {{for link in pagination:}}
            {{=link}}
            {{pass}}
            <li><a href="{{=URL(vars={'page':int(request.vars.page)+1})}}">>></a></li>
        </ul>
    </div>
    {{pass}}
```

source–code/views/filtered/unmark.load

```
<div class="container">
    <h3>Are you sure you want to remove the following message from the Filtered Messages folder?</
         h3>
        <h4>It will be moved back to the Messages folder.</h4>
    <div class="row">
        <div id="web2py_user_form" class="col-sm-9">
            <table>
                {{for row in messages:}}
                <tr>
                    <td class="w2p_fl">
                        <label class="readonly">Received</label>
                    </td>
                    <td class="w2p_fw">{{=row.msg_time}}</td>
                    <td class="w2p_fc"></td>
                </tr>
                <tr>
                    <td class="w2p_fl">
                        <label class="readonly">From</label>
                    </td>
                    <td class="w2p_fw">{{=row.msg_sender}}</td>
                    <td class="w2p_fc"></td>
                </tr>
                <tr>
                    <td class="w2p_fl">
                        <label class="readonly">Message</label>
                    </td>
                    <td class="w2p_fw">{{=row.msg_content}}</td>
                    <td class="w2p_fc"></td>
                </tr>
                <tr>
                    <td class="w2p_fl"></td>
                    <td class="w2p_fw">
                        {{=form}}
                    </td>
                    <td class="w2p_fc"></td>
                </tr>
                {{pass}}
            </table>
        </div>
    </div>
</div>
```

source–code/views/filter_settings/index.html

```
{{left_sidebar_enabled = True}}
{{right_sidebar_enabled = True}}

{{extend 'layout.html'}}
<h3>Configure message filter list</h3>
<p>Messages with senders that are included in this list will automatically be moved to the
     Filtered Messages folder.</p>
{{=LOAD('filter_settings','blacklist.load',ajax=True)}}
```

source–code/views/filter_settings/blacklist.load

```
{{=grid}}
```

source–code/views/tagr_settings/index.html

```
{{left_sidebar_enabled = True}}
{{right_sidebar_enabled = False}}

{{extend 'layout.html'}}

{{block left_sidebar}}
{{include 'tagr_settings/tagr_nav.html'}}
{{end}}
```

source–code/views/tagr_settings/tagr_nav.html

```
<h3>Telehealth Automatically Generated Recommendations (TAGR)</h3>
TAGR predicts tags of incoming messages.
<div>
    <h3>From here you can:</h3>
    <ul>
        <li><a href={{=URL('tagr_settings', 'train')}}>Train a new model</a></li>
        <li><a href={{=URL('tagr_settings', 'load')}}>Load a trained model</a></li>
        <li><a href={{=URL('tagr_settings', 'choose')}}>Choose a model to use</a></li>
    </ul>
</div>
```

source–code/views/tagr_settings/train.html

```
{{left_sidebar_enabled = True}}
{{right_sidebar_enabled = True}}

{{extend 'layout.html'}}

{{block left_sidebar}}
{{include 'tagr_settings/tagr_nav.html'}}
{{end}}
<h2>Train a new model</h2>
{{=form}}
{{if request.vars.ngrams and request.vars.percentile:}}
    <h3>n-grams: {{=request.vars.ngrams}} percentile:{{=request.vars.percentile}}</h3>
    {{=LOAD('tagr_settings','train_results.load',vars={'ngrams':int(request.vars.ngrams),'
        percentile':int(request.vars.percentile)},ajax=True,user_signature=True)}}
    {{=LOAD('tagr_settings','save.load',vars={'ngrams':int(request.vars.ngrams),'percentile':int(
        request.vars.percentile)},ajax=True,user_signature=True)}}
{{pass}}

{{block right_sidebar}}
{{include 'tagr_settings/tagr_guide.html'}}
{{end}}
```

source–code/views/tagr_settings/tagr_guide.html

```
<h2>Quick Guide</h2>
TAGR uses a Support Vector Machine to classify text. Models are trained and evaluated on live data
    . The data is prepared through feature extraction and feature selection.

<h3>n-grams</h3>
Features extracted from the data are represented as bag of words/n-grams. An n-gram is a
    contiguous sequence of n items.
In the sequence, "hoping to hear from you soon", the extracted n-grams are:
<ul>
    <li>n=1 (unigram): "hoping", "to", "hear", "from", "you" "soon"</li>
    <li>n=2 (bigram): "hoping to", "to hear", "hear from", "from you", "you soon"</li>
        <li>and so on...
</ul>
The user can set n from 1 to 10.

<h3>percentile</h3>
Not all extracted features can effectively be utilized in training a model. They can be reduced by
     performing feature selection.
The user can specify a percentile from 1 to 100 and selects that amount of best features to use
    based on statistical tests.
```

source–code/views/tagr_settings/train_results.load

```
<h3>Evaluation:</h3>
<pre style="display:inline-block;">{{=report}}</pre>
{{=LOAD('tagr_settings','test.load',ajax=True,user_signature=True)}}
```

source–code/views/tagr_settings/test.load

```
<h2>Test the model</h2>
{{=form}}
```

source–code/views/tagr_settings/save.load

```
<h2>Save the model</h2>
{{=form}}
```

source–code/views/tagr_settings/load.html

```
{{left_sidebar_enabled = True}}
{{right_sidebar_enabled = False}}

{{extend 'layout.html'}}

{{block left_sidebar}}
{{include 'tagr_settings/tagr_nav.html'}}
{{end}}
<h2>Load a model</h2>
<p>Evaluate and test previously saved models.</p>
<table class="table table-bordered table-striped table-condensed">
        <thead>
            <tr>
                <th>Date created</th>
                <th>Model name</th>
                <th>Extracted n-grams</th>
                <th>Selected percentile</th>
                <th>Status</th>
            </tr>
        </thead>
        <tbody>
        {{for row in mdl:}}
            <tr>
                <td>{{=row.mdl_time.strftime("%b %d, %Y %I:%M %p")}}</td>
                <td>{{=row.mdl_name}}</td>
                <td>{{=row.mdl_ngrams}}</td>
                <td>{{=row.mdl_percent}}</td>
                {{if row.is_deployed==True:}}
                <td>Deployed</td>
                {{else:}}
                <td>Not Deployed</td>
                {{pass}}
            </tr>
        {{pass}}
        </tbody>
    </table>
{{=form}}
{{if request.vars.model_id:}}
    {{for row in mdl:}}
        {{if row.id==int(request.vars.model_id):}}
        <h3>Loaded model: '{{=row.mdl_name}}'</h3>
        {{=LOAD('tagr_settings','load_results.load',vars={'mdl_name':row.mdl_name},ajax=True,
            user_signature=True)}}
        {{pass}}
    {{pass}}
{{pass}}
```

source–code/views/tagr_settings/load_results.load

```
<h3>Evaluation:</h3>
<pre style="display:inline-block;">{{=report}}</pre>
{{=LOAD('tagr_settings','test.load',ajax=True,user_signature=True)}}
```

source–code/views/tagr_settings/choose.html

```
{{left_sidebar_enabled = True}}
{{right_sidebar_enabled = False}}

{{extend 'layout.html'}}

{{block left_sidebar}}
{{include 'tagr_settings/tagr_nav.html'}}
{{end}}
<h2>Choose a model to use</h2>
<p>The chosen model shall be deployed and used by the TAGR module in predicting tags.</p>
<table class="table table-bordered table-striped table-condensed">
        <thead>
            <tr>
                <th>Date created</th>
                <th>Model name</th>
                <th>Extracted n-grams</th>
                <th>Selected percentile</th>
                <th>Status</th>
            </tr>
        </thead>
        <tbody>
        {{for row in mdl:}}
            <tr>
                <td>{{=row.mdl_time.strftime("%b %d, %Y %I:%M %p")}}</td>
                <td>{{=row.mdl_name}}</td>
                <td>{{=row.mdl_ngrams}}</td>
                <td>{{=row.mdl_percent}}</td>
                {{if row.is_deployed==True:}}
                <td>Deployed</td>
                {{else:}}
                <td>Not Deployed</td>
                {{pass}}
            </tr>
        {{pass}}
        </tbody>
    </table>
{{=form}}
```

source–code/views/input/index.html

```
{{extend 'layout.html'}}
<div class="container">
    <h3>Simulated input form</h3>
    <div class="row">
        <div id="web2py_user_form" class="col-sm-9">
            {{=form}}
        </div>
    </div>
</div>
```

source–code/modules/tagr.py

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_selection import SelectPercentile
from sklearn.feature_selection import chi2
from sklearn.svm import LinearSVC
from sklearn.externals import joblib
from sklearn.metrics import classification_report, f1_score, accuracy_score, precision_score,
     recall_score, confusion_matrix
from sklearn.cross_validation import cross_val_score, cross_val_predict

class TAGR(object):
    vectorizer = None
    selector = None
    classifier = LinearSVC(class_weight='balanced')
    full_classifier = None

    def initialize(self, n_grams, percentile):
        self.vectorizer = TfidfVectorizer(sublinear_tf=True, stop_words='english', ngram_range=[1,
            n_grams])
        self.selector = SelectPercentile(chi2, percentile=percentile)

    def prepare_data(self, samples, labels):
        data_vectorized = self.vectorizer.fit_transform(samples)
        data_reduced = self.selector.fit_transform(data_vectorized, labels)
        return data_reduced

    def train_on_whole(self, samples, labels):
        self.full_classifier = self.classifier
        self.full_classifier = self.full_classifier.fit(samples, labels)

    def evaluate(self, samples, labels):
        scores = cross_val_score(estimator=self.classifier, X=samples, y=labels, cv=10)
        predictions = cross_val_predict(estimator=self.classifier, X=samples, y=labels, cv=10)

        eval_report = "Scores:" + str(scores) + "\n"
        eval_report += "Accuracy:" + str(scores.mean()) + "(+/- " + str(scores.std() * 2) + ")\n\n
            "
        # eval_report += 'Accuracy:' + str(accuracy_score(labels, predictions)) + "\n"
        eval_report += 'Precision:' + str(precision_score(labels, predictions, average='weighted')
            ) + "\n"
        eval_report += 'Recall:' + str(recall_score(labels, predictions, average='weighted')) + "\
            n"
        eval_report += 'F-1 Score:' + str(f1_score(labels, predictions, average='weighted')) + "\n
            "
        eval_report += classification_report(labels, predictions,
                                    labels=['Dermatology','General Question',
                                            'Internal Medicine','Medico Legal',
                                            'Neurology','Obgyn','Ophthalmology',
                                            'Otorhinolaryngology',
                                            'Pediatrics','Radiology',
                                            'Surgery'],
                                    target_names=['Derma','General',
                                                 'Internal Med','Medicolegal',
                                                 'Neurology','Obgyn','Ophtha',
                                                 'ENT','Pediatrics','Radiology',
                                                 'Surgery'])

        eval_report += "Confusion Matrix:\n" + str(confusion_matrix(labels, predictions))
        eval_report += "\n(row = expected, col = predicted)"
        return eval_report

    def classify(self, sample):
        sample_vectorized = self.vectorizer.transform([sample])
        sample_reduced = self.selector.transform(sample_vectorized)
        return self.full_classifier.predict(sample_reduced)[0]

    def save(self, name):
        joblib.dump(self.vectorizer, name + '-vectorizer.pkl')
        joblib.dump(self.selector, name + '-selector.pkl')
        joblib.dump(self.full_classifier, name + '-classifier.pkl')

    def load(self, name):
        self.vectorizer = joblib.load(name + '-vectorizer.pkl')
        self.selector = joblib.load(name + '-selector.pkl')
        self.full_classifier = joblib.load(name + '-classifier.pkl')
```

source–code/static/css/tagr–style.css

```
/*
www.colourlovers.com/palette/482774/dream_magnet
#343838
#005F6B
#008C9E
#00B4CC
#00DFFC
*/

.hlist ul {
        list-style-type: none;
    margin: 0;
    padding-left: 16px;
    overflow: hidden;
    background-color: #008C9E;
}

.hlist li {
        float: left;
}

.hlist li a {
        display: block;
    color: #fff;
    text-align: center;
    padding: 8px 16px;
    text-decoration: none;
}

.hlist li a:hover {
    background-color: #00DFFC;
}

.hlist .active {
    background-color: #00B4CC;
}

.hlist .logout {
    float: right;
    padding-right: 16px;
}

.footer {
    background: #343838;
    color: #fff;
}

header h1 {
    float: left;
}

header img {
    float: right;
    display: block;
    max-height: 64px;
    width: auto;
    height: auto;
}
```

XI. Acknowledgement

First of all, I would like to thank my adviser, Sir Marvin Ignacio, for all his help in completing my SP. He could have spent the time and effort working on his own thesis, and yet he still gave everything he could to all his advisees.

I would like to thank Sir Randy Fernandez for introducing me to the topic and for referring me to the National Telehealth Center. I would like to acknowledge Dr. Portia Marcelo, Dr. Alvin Marcelo, and Sir Luis Macabasag for all their support.

I would also like to thank Sir Richard Bryann Chua for the time he spent as my former adviser, and for shooting down all my proposed research topics. Likewise, I would like to express my gratitude to Ma'am Avegail Carpio and Sir Bernie Terrado for their guidance.

I would like to thank Lean Rada, for his insights on the technical aspects of my SP, and the rest of my blockmates for their help in the writing process. However, this is not just about the SP. This is also about the end of my five-year journey, and how I want to acknowledge everyone who has been a part of it.

To my professors: thank you for all your efforts. Doc Magboo, Ma'am Magboo, Sir Solano, Sir Baes, Ma'am Perl, Sir Jireh, and many more; I may not be able to name all of you here, but know that I am grateful for all the things I learned from you, and that I shall put them to good use. To Maam Eden: thank you for being patient with all of us, and for always helping us out.

To my friends from UP MOrg, especially my bandmates: thank you for giving me a life outside the block. To our FBCs, UP SoComSci, and the CAS SC: thank you for instilling in me a sense of service to both my fellow students and the people.

To the friends I made during a particularly strange time in my life: thank you.

To my block, UPM 2011-12 Pusoy Varsity: screw you, guys. I'm done. HAHAHA but seriously, thanks for everything. I can't imagine a better set of people to go through college with. Thanks for giving me the opportunity to serve as your blockhead

for more than four years (when it should only have been four, sigh.) You guys are literally too many to mention here, so I might go at it by group.

To Joyce et al.: thanks for serving as an inspiration for everyone else in the block, and for showing that we can all make it by helping each other.

To Lizette et al: thanks for reminding everyone that there's a life outside of acads.

To Marbert: thanks for always being such a good sport, and for always giving back to the block no matter what.

To Christine, Kriselle, Lei, and Sheila: I forgot why I used to hang out with you guys, but thanks anyway.

To Carl: thanks for the overwhelming support that you gave to me and all the others working on their SP, and for taking up some of my responsibilities that I left behind.

To Gio, Renz, and Lean: I am honored to have worked side by side with you guys. Thank you for the experience, and may we have more opportunities to do so in the future. srsly best team gg

To Katdat: thank you for being that friend at a time when I needed it the most.

To Viktor: I am literally just putting you here because you asked me to. Plus, thanks for being there for me since high school.

To Katherine: thanks for sticking with me ever since high school, through the good and the bad. It wasnt perfect, but we made a great team.

To Cara: why are you even here? Hahaha. Thanks for everything, but more importantly, for reminding me about the things that actually mattered. I will forever be grateful to you.

To my family: thank you for your never-ending support, and for understanding my situation. I may have had a lot of words above for my friends, but at the end, it is you who deserve the most credit. We all worked hard for this.

It is done.