Automatic Coding for Neonatal Jaundice From Free Text Data Using Ensemble Methods

Scott Werwath
UC Berkeley
sbw@berkeley.edu

***Abstract*—This study explores the creation of a machine learning model to automatically identify whether a Neonatal Intensive Care Unit (NICU) patient was diagnosed with neonatal jaundice during a particular hospitalization based on their associated clinical notes. We develop a number of techniques for text preprocessing and feature selection and compare the effectiveness of different classification models. We show that using ensembled decision tree classification with AdaBoost outperforms support vector machines (SVM), the current state-of-the-art technique for neonatal jaundice coding.**

I. Introduction

A. Medical Coding

In the course of normal clinical operations, healthcare providers are often required to *code* patient records. The process of coding involves taking all data associated with a single patient case and abstracting out a set of standardized alphanumeric codes where each unique code is associated with a unique diagnosis or medical procedure. These codes are used for a variety of applications, including medical billing, as labels in statistical analysis of medical datasets, as features in clinical decision making, and as tags in information retrieval systems.

While many coding schemes exists, the most widely used is the International Classification of Diseases, 9th Revision (ICD-9) [25]. Each ICD-9 codes consists of 3 to 5 alphanumeric characters which represents either a diagnosis or a procedure.

In actual clinical practice, coding is performed manually by professional medical coders. This process is both time-intensive and error-prone due to the fact that patient health records contain a large amount of data in free-text narrative format which is difficult to interpret quickly for both humans and computers [26] [20].

A system for automatic coding would have the potential to substantially decrease the number of man-hours required to process clinical cases, but building such a system requires the ability to extract important information from free-text reports. To this end, the application of natural language processing (NLP) and machine learning (ML) techniques to clinical texts show much promise [19].

B. Neonatal Jaundice

Neonatal jaundice, also called neonatal hyperbilirubinemia, is a yellow discoloration of the skin due to elevated bilirubin in a neonate's (infant's) blood. Transient neonatal jaundice is somewhat common in infants and usually carries little risk. However, untreated jaundice can put premature or otherwise ill infants at great risk of developing permanent neurological impairments [16]. Additionally, it has been shown that the diagnosis of neonatal jaundice is a risk factor for both mortality and rehospitalization [9].

Both the risk factors associated with and the relative prevalence of neonatal jaundice make it an excellent subject for study using machine learning techniques; the risk factors provide the motivation, while the prevalence provide a wealth of data to be analyzed.

II. Related Work

The past few years have seen a meteoric rise in the application of natural language processing (NLP) in the medical field [27]. The cost of analyzing medical documents, both monetary and time, has motivated efforts to automate many manual tasks in the analysis and processing of medical texts.

Given both the importance and time-intensive nature of medical coding, many scientists have started developing techniques for automated coding [19]. For most of these attempts, the task was domain-specific (i.e. attempting to predict a single ICD code or a small set of related codes) and utilized a relatively small dataset.

At least one attempt has been made to use deep learning (namely character-level recurrent neural networks) to build more general coding models, but with limited success [23]. The main limitation with this method is that deep learning models require a large amount of data to train [5], but even a large clinical dataset may only contain a few examples of case files with any given ICD code. Additionally, since the ICD-9 standard contains over 14,000 different codes. Treating coding as a simple multi-class classification problem with 14,000 different classes is infeasible.

More successful automated coding models have utilized non-deep machine learning techniques such as support vector machines (SVMs) [7] and focus on training a model to detect the presence of a single ICD code or class of related codes. Some of these methods also leverage structured data stored in health records in addition to free-text narratives [11].

At least one previous attempt has been made to build an automatic coding model for neonatal jaundice using an SVM on the text of clinical notes [18]. Our research improves on this model by introducing a number of preprocessing stages. We also explore and compare alternative classification techniques and analyze their efficacy.

No matter the exact method for extracting medical codes from free-text, two common NLP challenges must be solved. First, one must represent free-text in an appropriate feature space (usually as a vector). Second, one must train a model to accurately classify patient records based on the selected features.

III. Methodology

A. Dataset and Data Extraction

The Medical Information Mart for Intensive Care III (MIMIC-III) dataset contains complete anonymized records from over 53,000 ICU stays [15]. These records were gathered from hospital records from the Beth Israel Deaconess Medical Center in Boston, Massachusetts between 2001 and 2012. Of these records, 7870 correspond to neonates admitted between 2001 and 2008.

Each hospital stay in the MIMIC-III dataset is assigned a unique HADM_ID identifier which can be used to cross-reference associated data back their associated hospitalization.

In order to find all patients who were diagnosed with neonatal jaundice, we found all HADM_IDs that were labeled with an ICD-9 code associated with neonatal jaundice. A list of such codes are shown in Table I. In total, 7092 hospitalizations were of neonates. Of these, 2868 stays were associated with a neonatal jaundice diagnosis and served as positive examples. We also selected the remaining 4224 non-jaundice NICU stays as negative examples.

TABLE I
ICD-9 Codes for Neonatal Jaundice

ICD-9 Code	Description
773.0	Hemolytic disease, RH isoimmunization
773.1	Hemolytic disease, ABO isoimmunization
773.0	Hemolytic disease, unknown
774.1	Perinatal jaundice, excessive hemolysis
774.2	Neonatal jaundice, preterm delivery
774.30	Neonatal jaundice, delayed conjugation
774.31	Neonatal jaundice, delayed conjugation, other
774.39	Other jaundice
774.6	Unspecified fetal and neonatal jaundice

For each relevant hospitalization, we concatenated all ICU notes, including nursing notes, radiology reports, physician progress notes, and discharge summaries into a single string of text called a *noteset*. This string served as the raw data to be preprocessed and then classified.

B. Data Preprocessing

Preprocessing of raw text is an extremely important element of any NLP model, especially in the task of classification [14]. In fact, the preprocessing of raw text is a form of manual feature selection, in which we are able to choose what parts of the document are to be given to the classifier [10].

The first stage of our preprocessing algorithm is *cleaning*, in which punctuation and other extraneous characters are stripped from the raw text. In addition to punctuation, numbers are also removed from the noteset. While numbers are often left in classifiers for medical texts, later cross-validation indicated that they were low-information for this particular task. Finally, we also remove anonymization indicators from the text of the doument. The indicators are special flags left to indicate that confidential patient data had been redacted from the noteset at that point in the text, and generally contained information that was irrelevant for classification such as dates, names, and phone numbers.

After the noteset is cleaned, it is *tokenized*, or split into a stream of individual words. The next series of preprocessing steps act on individual words or multiword phrases, and they work to reduce the *vocabulary*, or number of unique words, of our dataset. Vocabulary reduction often results in improved classification performance [17] because the model does not have to learn to classify low-information words or learn that synonymous words or phrases should be classified similarly. Depending on your representation of documents as vectors (e.g. one-hot encodings), vocabulary reduction corresponds directly to a dimensionality reduction of samples.

The first stage of vocabulary reduction is *semantic mapping*, i.e. the replacement of synonymous words or phrases with a standard token [1]. This allows the model to learn only one representation of the underlying concept rather than having to learn many. For this, we use a list of 727 common medical words and phrases [**?**] and use regular expressions to find and replace each of them in the notesets. Examples of such replacements can be found in Table II.

TABLE II
Example Replacements During Preprocessing

Rule	Raw Text	Processed Text
Semantic Map	”raises concern”	”RISK”
Semantic Map	”cannot rule out”	”RISK”
Semantic Map	”no evidence of”	”NEGEX”
Semantic Map	”rule out”	”NEGEX”
Stemming	”neonatal”	”neonat”
Stemming	”neonate”	”neonat”
Stop Word Removal	”the patient is”	”patient”
Phrase Detection	”heart attack”	”heart_attack”

After semantic mapping, we apply *stemming*, which is the process of reducing words to their root morpheme (see Table II). For this task, we use the Snowball stemming algorithm [22]. Since different morphological variations of the same word (e.g. ”neonatal” and ”neonate”) likely have similar meanings, stemming allows the model to learn a single classification for all morphological forms for a word.

Next, *stop words* were removed from the notesets. Stop words are common words such as ”the” and ”it” which convey no real semantic information. The removal of stop words from raw text has been shown to improve the performance of classification models due to their being low-information and useless for most classification tasks [24].

Then, collocations, or common multiword phrases, are detected and replaced with a single token. In order to automatically detect collocations in our corpus, we calculate the *normalized pointwise mutual information* (NPMI) of every pair

of words in the corpus [2]. The NPMI is given by (1), where x and y are indicators for a two given words appearing in text, $p(x, y)$ is the probability of the two words appearing next to each other, and all probabilities are measured empirically from our corpus.

$$i_n(x, y) = \frac{-1}{\ln p(x, y)} \ln \frac{p(x, y)}{p(x)p(y)} \tag{1}$$

Intuitively, NMPI measures how much the the co-occurance of two words x and y differs from what we would expect if their occurrences were independent of each other. For word pairs in our corpus with high NMPI, we normalized co-occurances in raw text into a single token (see Table II).

Finally, the text was searched for phrases such as "no" and "rule out" that indicate negation. We assume that the three words following such phrases are negated, and so we remove them from the document. Since our bag of words representation fails to take word order into account, this prevent the model from interpreting the word "jaundice" in the phrase "rule out jaundice" as indicating a positive result.

C. Document Representation

In order to classify notesets, we have to represent them as vectors. While deeper methods (e.g. convolutional neural networks) are able to take word order into account, we choose to use a *bag-of-words* (BoW) model in which we ignore the actual ordering of words in a document and instead simply count the number of occurrences of each unique word in the document. This has the benefit of greatly simplifying our document representation, but it throws away potentially valuable information contained in word ordering (e.g. "negative for cancer, positive for flu" and "positive for cancer, negative for flu" will have the same BoW representation). Despite this disadvantage, however, BoW representations work very well in practice for many applications.

A naive approach to BoW representation is a *count vector*. In a count vector, each unique word is represented by a V-dimensional one hot encoding vector, where V is the size of the vocabulary. A document is then represented by the sum (or sometimes average) of the one hot encoding of all the words in the document. The result is that, at each index in a document vector, the entry at index i is the number of times the i^{th} word in the vocabulary occurs in the document.

A commonly used technique to improve upon the count vector representation to weight each entry by the term's *inverse document frequency* (IDF), given in (2). N is the total number of documents in the corpus, and the denominator represents the number of documents in the corpus where the word t appears.

$$idf(t) = \log \frac{N}{|\{d \in D : t \in d\}|} \tag{2}$$

The IDF measures the rarity of a word in a given corpus. From an information theoretic perspective, words with high IDF (i.e. rarer words) provide more information, so we want to give them more weight in our decision making process.

When we weight each count vector entry by its word's IDF and normalize appropriately, we get that each entry t in the vector representation for document d is given by (3).

$$tfidf(t, d) = \frac{|t' \in d : t' = t|}{|d|} idf(t) \tag{3}$$

This value is called the *term frequency-inverse document frequency* (TF-IDF), and a vector of these values is called a TF-IDF vector.

D. Baseline SVM Model

Reference [**?**] demonstrates the use of a support vector machine (SVM) to perform automatic coding for neonatal jaundice. This study used a grid search to determine that a linear kernel with penalty hyperparameter of $C = 100$. This model was reimplemented, and our own grid search over relevant hyperparmeters with 10-fold cross-validation found those same settings to be optimal. This SVM model was used as a baseline to compare with the performance of new ensemble learning methods.

E. Ensemble Methods

Ensemble learning refers to the practice of training and using multiple machine learning models and combining their outputs into a final output via a "voting" system. Ensemble methods often provide better results than using a single model because they can reduce variance. When a set of models are trained with random samples of a dataset or on a random subset of features, we are essentially sampling from the space of possible classifiers. By taking many samples and averaging their results, we reduce our reliance on the particularities of any one classifier and therefore reduce variance [8].

We used the sklearn package [21] to implement two ensembled classifiers: AdaBoost with decision trees [28] and bagged decision trees [3].

The base estimator of both our ensemble methods are *decision trees*. A decision tree uses a tree-like model, where each node represents a decision based on the value of a single feature, and each subsequent branch represents a path chosen based on that decision. Trees are constructed such that each decision boundary maximally decreases the Gini Impurity of the two sets of samples which would fall on either side of the boundary. Put another way, each decision node of the trees attempts to best separate the training data based on the value of a single feature. For our implementation of decision trees, the depth of the tree is allowed to grow indefinitely until all leaves contain pure subsets of the training set.

Decision trees tend to suffer from problems such as instability and overfitting [12]. One technique to counter such problems is *bagging*, in which n different decision trees (in our case, $n = 10$ was found to be optimal) are trained on different samples of the training data. To classify a new sample, each of the n trees performs classification, and then a vote is taken to determine the final label.

AdaBoost is another ensemble method which attempts to improve on the performance of a single base classifier. Rather

than sampling uniformly from the training set to train each classifier, each decision tree is trained on the same training set, but with each training point has different weights for each classifier. The values of the weights for classifier k are determined by how poorly the first $k-1$ classifiers predict the point, so that greater weight is given to misclassified points. To classify a new sample, each individual trees performs classification, and then a weighted vote is taken to determine the final label. Using cross-validation, we found that using 40 boosted decision trees worked best for our problem.

IV. Results

The performance of the baseline classifier from [18], our implementation of SVM, and the two ensemble classifiers are summarized in Table III. The F_1 score refers to the harmonic

TABLE III
Performance of Various Classifiers

Classifier	Accuracy	Precision	Recall	F_1 Score
SVM	0.865	0.835	0.839	0.837
Boosted Trees	0.907	**0.923**	0.846	0.883
Bagged Trees	**0.909**	0.904	**0.876**	**0.890**

average of precision and recall, given in (4).

$$F_1 = \frac{2}{precision^{-1} + recall^{-1}} \tag{4}$$

We see that our reimplementation of the SVM model in [18] did not perform as well as the original, though this is likely due to differences in the dataset. Both of our ensemble methods outperform the two SVM implementations in at least one of the performance metrics.

We see the receiver operating characteristics (ROC) of the ensemble methods in Fig. 1. We see that the baseline SVM performance lies under the the ROC curves of both ensemble methods, implying that the ensemble methods are able to outperform SVM.

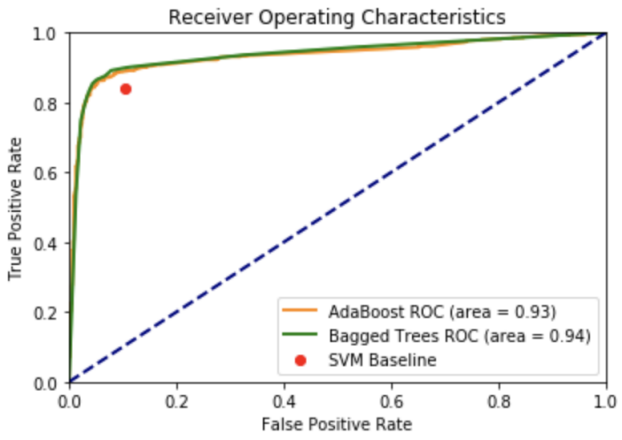

Fig. 1. ROC curves of ensemble methods compared to SVM baseline

Finally, an AdaBoost classifier was trained on only discharge summaries, rather than entire notesets. The accuracy of the classifier decreased from 0.91 to 0.86, implying that non-discharge notes contained information relevant to the classification.

V. Discussion

A. Performance of Ensemble Methods

Both the AdaBoost and bagged trees classifiers outperformed the baseline SVM. While we were unable to find statistics on the accuracy of human medical coders in coding for neonatal jaundice, the average accuracy of humans in coding for a large variety of diagnoses seems to range roughly from 0.90 to 0.95 [20]. Therefore, it is possible that our models are approaching human-level performance at the task of coding for neonatal jaundice.

When training ensemble methods, one important hyperparameter is the number of estimators to train. Fig. 2 shows the performance of AdaBoost as a function of the number of boosted decision tree estimators. We see that adding more

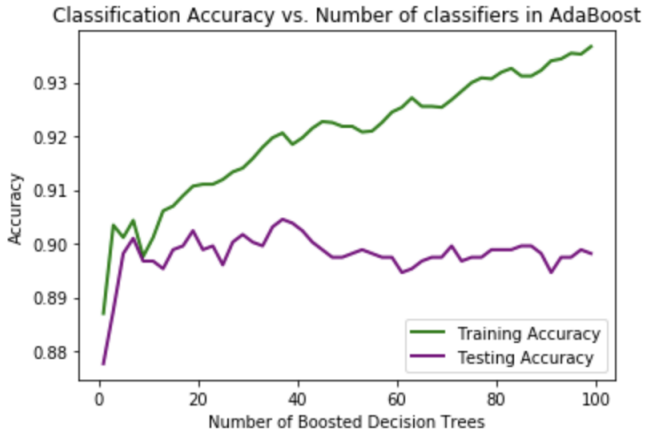

Fig. 2. The effect of adding more classifiers on AdaBoost accuracy. Overfitting occurs when more than 40 classifiers are used.

than 40 classifiers increases training accuracy while having no effect on testing accuracy, demonstrating that overfitting is occurring.

In the task of medical detection, recall is often considered a more important metric than precision, since false positives can be manually checked and corrected by a human while false negatives may go unnoticed. As such, the bagged trees model may be favored for actual use in a clinical setting.

B. Feature Importance

While SVM was outperformed by ensemble methods, it can nonetheless provide interesting insight into what words are most likely to indicate neonatal jaundice.

Recall that, in SVM with a linear kernel, the learned classifier is the ideal separating hyperplane. This hyperplane, like all hyperplanes, can be uniquely represented by a single vector which is orthogonal to the hyperplane; we will call this the coefficient vector. Since features that are orthogonal (or close to orthogonal) to the hyperplane are most relevant to

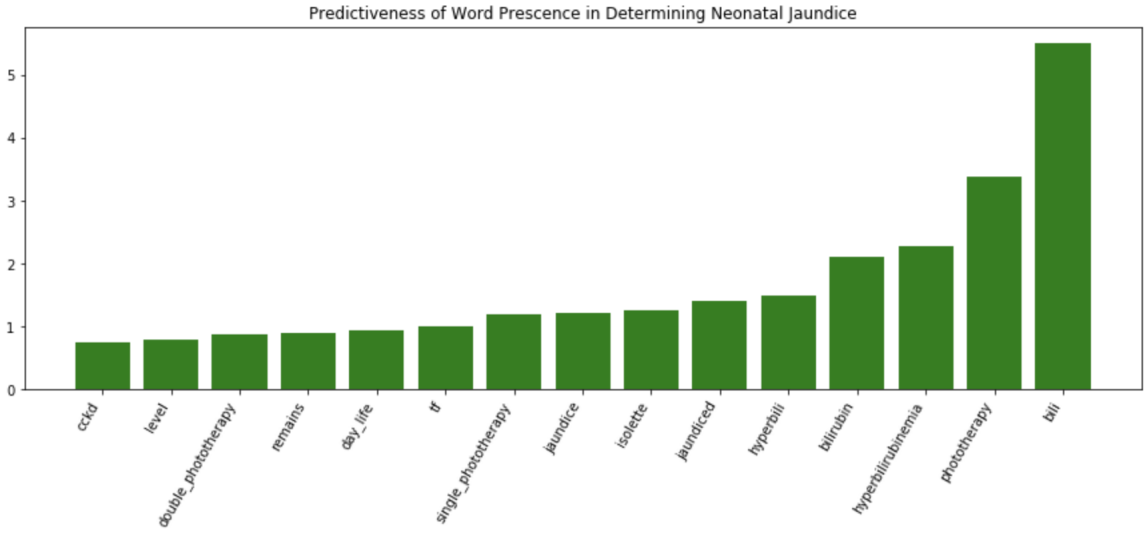

Fig. 3. Relevance of words in determining a positive SVM classification

classification, we can interpret the entries of the coefficient vector as a metric of feature importance, where large positive entries indicate that a feature contributes greatly to a positive classification [13].

In TF-IDF vectors, each dimension (i.e. feature) corresponds to a single word's occurrence in that noteset. Therefore, that feature's corresponding coefficient measures the predictiveness of that word's presence to a diagnosis of neonatal jaundice. We plot these most-relevant terms in Fig. 3. We see that the most relevant features include terms such as "bili", an abbreviation for bilirubin (the chemical which causes jaundice), and "phototherapy", the standard treatment for patients with jaundice. Based on a manual inspection of the top features, it seems that SVM is learning to classify notesets based on words that are relevant to neonatal jaundice.

VI. Conclusion

We have shown that both bagged decision trees and boosted decision trees are able to outperform SVM at the task of automatically detecting a neonatal jaundice diagnosis from clinical notes and reports, and these ensemble methods approach human-level accuracy. The ability of these models to achieve high performance based entirely on free text data implies that the clinical notes contain sufficient information to carry out the task of assigning ICD codes. As more labeled clinical text becomes available, training highly accurate automatic coding models will become feasible for a wider set of ICD codes.

References

[1] Banerjee, I., Madhavan, S., Goldman, R. E., Rubin, D. L. (2017). Intelligent Word Embeddings of Free-Text Radiology Reports. AMIA Annual Symposium.

[2] Bouma, G. (2009). Normalized (Pointwise) Mutual Information in Collocation Extraction. Proceedings of German Society for Computational Linguistics (GSCL 2009), 3140.

[3] Breiman, L. (1996). Bagging predictors. Machine Learning, 24(2), 123140. Centers for Medicare and Medicaid Services. (2016). ICD-10-CM/PCS The Next Generation of Coding.

[4] Chapman, B. E., Lee, S., Kang, H. P., Chapman, W. W. (2011). Document-level classification of CT pulmonary angiography reports based on an extension of the ConText algorithm. Journal of Biomedical Informatics, 44(5), 728-737.

[5] Chen, X.-W., Lin, X. (2014). Big Data Deep Learning: Challenges and Perspectives. IEEE Access, 2, 514-525.

[6] Conneau, A. (2016). Very Deep Convolutional Networks for Text Classification. Neurocomputing, 3(1), 130-142.

[7] Cortes, C., Vapnik, V. (1995). Support Vector Networks. Machine Learning, 20(3), 273-297.

[8] Dietterich, T. G. (2000). Ensemble Methods in Machine Learning. Multiple Classifier Systems, 1857, 1-15.

[9] Escobar, G. J., Joffe, S., Gardner, M. N., Armstrong, M. A., Folck, B. F., Carpenter, D. M. (1999). Rehospitalization in the First Two Weeks After Discharge From the Neonatal Intensive Care Unit. Pediatrics, 104.

[10] Fatih, M., Bayir, S. (2017). Examining the Impact of Feature Selection Methods on Text Classification. IJACSA) International Journal of Advanced Computer Science and Applications, 8(12).

[11] Ferrao, J. C., Janela, F., Ferrao, J. C., Martins, H. M. G. (2013). Using structured EHR data and SVM to support ICD-9-CM coding. Proceedings - 2013 IEEE International Conference on Healthcare Informatics, ICHI 2013, 511-516.

[12] Gareth, J., Witten, D., Hastie, T., Tibshirani, R. (2015). An Introduction to Statistical Learning. New York: Springer.

[13] Guyon, I., Weston, J., Barnhill, S., Vapnik, V. (2002). Gene Selection for Cancer Classification using Support Vector Machines, Machine Learning. Machine Learning, 46(1-3), 389-422.

[14] Haddi, E., Liu, X., Shi, Y. (2013). The role of text pre-processing in sentiment analysis. In Procedia Computer Science (Vol. 17, pp. 26-32).

[15] Johnson, A. E., Pollard, T. J., Shen, L., Lehman, L. W. H., Feng, M., Ghassemi, M., Mark, R. G. (2016). MIMIC-III, a freely accessible critical care database. Scientific Data, 3.

[16] Lantzy, A. (2015). Neonatal Hyperbilirubinemia. In Merck Manual.

[17] Madsen, R. E., Sigurdsson, S., Hansen, L. K., Larsen, J. (2004). Pruning The Vocabulary For Better Context Recognition. Pattern Recognition.

[18] Marafino, B. J., Davies, J. M., Bardach, N. S., Dean, M. L., Dudley, R. A. (2014). N-gram support vector machines for scalable procedure and diagnosis classification, with applications to clinical free text data from the intensive care unit. Journal of the American Medical Informatics Association : JAMIA, 1-6.

[19] Meystre, S. M., Savova, G. K., Kipper-Schuler, K. C., Hurdle, J. F. (2008). Extracting information from textual documents in the electronic health record: a review of recent research. Yearbook of Medical Informatics, (November 2007), 128-44.

[20] OMalley, K. J., Cook, K. F., Price, M. D., Wildes, K. R., Hurdle, J. F.,

Ashton, C. M. (2005). Measuring diagnoses: ICD code accuracy. Health Services Research.
[21] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Duchesnay, . (2012). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, 2825-2830.
[22] Porter, M. (1980). An Algorithm for Suffix Stripping. Program, 14(3), 130-137.
[23] Shi, H., Xie, P., Hu, Z., Zhang, M., Xing, E. P. (2017). Towards Automated ICD Coding Using Deep Learning.
[24] Silva, C., Ribeiro, B. (2003). The importance of stop word removal on recall values in text categorization. Proceedings of the International Joint Conference on Neural Networks, 3, 1661-1666.
[25] Slee, V. (1978). The International Classification of Diseases: Ninth Revision (ICD-9). Ann Intern Med., 424-426.
[26] Tange, H. J., Schouten, H. C., Kester, A. D. M., Hasman, A. (1998). The granularity of medical narratives and its effect on the speed and completeness of information retrieval. Journal of the American Medical Informatics Association, 5(6), 571-582.
[27] Wolniewicz, R. (2015). Computer-assisted coding and natural language processing Executive summary.
[28] Zhu, J., Arbor, A., Hastie, T. (2006). Multi-class AdaBoost, 0-20.