Study and Analysis of Text Document Classification Algorithms

Arjun Patel[1], Harshil Prajapati[1], and Bowen Song[1]

[1]Department of Electrical and Computer Engineering, Boston University

1. Problem Definition

This project explores machine learning algorithms for document classification. The specific application of the project is sensitive to datasets. The contribution of the project is to compare and contrast algorithm performances based on results and computation efficiency with respect to different types of datasets.

2. Literature Review

Document Classification is a traditional problem where usually a bag-of-words approach is to extract features and are used for supervised classification using Naive Bayes or Support Vector Machine (SVM) algorithms [1].

In classical papers, the order of words is ignored, but the order can be included by using Term Frequency - Inverse Document Frequency (TF-IDF) scheme where basic identification of sets of words that are discriminative for documents in the collection but reveals little in the way of inter- or intra- document statistical structure [2]. To overcome the shortcomings of TF-IDF, latent semantic indexing was introduced in [3]. It uses a singular value decomposition of the X matrix to identify a linear subspace in the space of TF-IDF features that capture most of the variance in the collection. This was modified by [4] to fitting the model to data using maximum likelihood or Bayesian methods known as Probabilistic latent semantic indexing (pLSI). Latent Dirichlet allocation (LDA) considers exchangeable representations for documents and words, we need to consider mixture models that capture the exchangeability of both words and documents [5]. A recent study shows a use of Convolutional Neural Network (CNN) [6] to get improved results.

Due to the size of documents, different techniques like topological spaces for developing Information Retrieval System (IRS) [7], Ontology-Based Feature Vector Generation [8] have been developed which can effectively reduce the computation while extracting features that aids in classification. Kernel methods (KMs) are an effective alternative to explicit feature extraction. Another way to efficiently extract features is based on finding the conditional probability using N-gram models [9].

Moreover the traditional supervised learning techniques, research has been done to improve predictions in k-NN by using pyramidal decomposition [10], skip-gram and paragraph vectors-distributed bag of words (PV-DBOW) with multiple discriminant analysis [11], expansion method and Powerset-label mechanism for the short hierarchical classification using the Support Vector Machine (SVM) classifier [12], Bayes-optimum decision with the maximum margin principle yields kernels for SVM [13].

3. Proposed Work

This project aims at the implementation of various document classification algorithms discussed in Section 4. The project also includes stop-words filtering based on stop-words vocabulary list and TF-IDF in attempt to improve performances for each model. The project learns from different datasets and the overview of the chosen datasets is discussed in Section 5. In addition to discussing algorithm performances, the project also concerns the importance of preserving word ordering within a document with respect to classification accuracy which is explored by the N-gram model.

4. Model and Algorithms

This section gives an overview of all algorithms implemented in this project.

4.1. Preprocessing using TF-IDF

A way to do thresholding aside from just word frequency is by determining how important a word is in predicting the class of a document. To determine how important a word is, we are using TF-IDF weighting, which takes into consideration of word frequency as well as document frequency. The weight of the word is the frequency of a word multiplied by the inverse document frequency. For example, a word with a high word frequency and a low document frequency will be assigned a high weighting. Using the TF-IDF, we can

set a threshold, which would decrease the feature size and remove irrelevant words [14].

4.2. Bag-of-words model with Naive Bayes assumption

The Naive Bayes classifier with bag-of-words model considers all features as conditionally independent given the class label and evaluates using MPE decision rule [15]. The prediction performance is considered to be the baseline for our prediction accuracy. The decision is based on MPE rule (1).

$$h_{MPE} = P(Y_j|\boldsymbol{x}_j) \propto P(Y_j)\prod_{i=1}^{d_j} P(w_{i,j}|Y), \quad j = 1,\ldots,n \tag{1}$$

4.3. Multi-class logistic regression (Maximum Entropy Classifier)

The logistic regression of multiples class is a supervised learning model by maximizing the conditional likelihood.

$$\hat{\theta}(\mathbb{D}) = arg\ max_\theta \sum_{j=1}^{n} p(y_j|x_j,\theta),\ j = 1,\ldots,n \tag{2}$$

Without a closed form, best θ is learnt via gradient decent with respect to the following equations.

$$\nabla_{\boldsymbol{w}_y} NLL(\theta) = \sum_{j=1}^{n}\left(\frac{e^{\boldsymbol{w}_y{}^T \boldsymbol{x}_j^{ext}}}{\sum_{k=1}^{m} e^{\boldsymbol{w}_k{}^T \boldsymbol{x}_j^{ext}}} - 1(y_j = y)\right)\boldsymbol{x}_j^{ext} \tag{3}$$

$$\nabla_{\boldsymbol{w}_y} f(\theta) = \nabla_{\boldsymbol{w}_y} NLL(\theta) + \lambda \boldsymbol{w}_y, y = 1,\ldots,m. \tag{4}$$

As a MPE/MAP Classifier, the decision is based on MPE rule:

$$h_{MPE}(x) = arg\ max_{y\in\{1,\ldots,m\}p;(y|\boldsymbol{x},\theta)} \tag{5}$$

4.4. Sensing aware kernel SVM

This SVM kernel combines Bayes-optimum decision boundary with Maximum margin principle. The optimal classifier, in a binary class scenario, has the following form

$$f(\boldsymbol{x}) = sign\left(\sum_{i=1}^{n} \beta_i^* K(\boldsymbol{x},\boldsymbol{x}^i)\right), \tag{6}$$

where the kernel is defined by

$$K(\boldsymbol{x},\boldsymbol{x}') := < p(\boldsymbol{x}|\boldsymbol{z}), p(\boldsymbol{x}'|\boldsymbol{z}) > . \tag{7}$$

The sensing kernel for Bag of words model targeting two documents x^i and x^j has the following form:

$$\begin{aligned} K(x^i,x^j) &= \prod_{w=1}^{W} \frac{\Gamma(x_w^i + x_w^j + 1)}{\Gamma(x_w^i+1)\Gamma(x_w^j+1)} \frac{\Gamma(N_i+1)\Gamma(N_j+1)}{\Gamma(N_i+N_j+W)} \\ &= \prod_{w=1}^{W} \frac{(x_w^i + x_w^j)!}{(x_w^i)!(x_w^j)!} \frac{N_i!N_j!}{(N_i+N_j+W-1)!} \\ &\text{where } N_{i,j} = \text{Number of words in i,j document} \\ &\text{and } \Gamma(t) = \text{Ordinary Gamma function.} \end{aligned} \tag{8}$$

This method can be extended into a multi-class classification as described in [13].

4.5. N-gram

The N-grams model uses the probability of each word appearing. For unigram, it considers each word as independent of all other words. Therefore, for a sentence to appear, it would be the probability of each word appearing multiplied together. For bigram, it takes the probability of the first word appearing multiplied by the probability of the second word appearing given the first word and so on. The probability of the word depends on the probability of the all the previous words that appear before it in a sentence. [16]

$$\textit{Unigram:} P(w_{1,n}) = P(w_1).P(w_2)\cdots P(w_n) \tag{9}$$

$$\textit{Bigram:} P(w_{1,n}) = P(w_1).P(w_2 \mid w_1)\cdots P(w_n \mid w_{n-1}) \tag{10}$$

In the N-gram models, the features can become very large i.e. D^N for N-gram model which can be reduced by setting a threshold frequency of appearance [9].

4.6. Bigrams

The joint probability of sequence of n words $w_1, w_2, \cdot w_n$ can be written as

$$\begin{aligned} P(w_{1,n}) &= P(w_1).P(w_2 \mid w_1).P(w_2 \mid w_1^2)\cdots P(w_n \mid w_1^{n-1}) \\ &= \prod_{k=1}^{n} P(w_k \mid w_1^{k-1}). \end{aligned}$$

The bigram model, for example, approximates the probability of a word given all the previous words $P(w_n \mid w_1^{n-1})$ by using only the conditional probability of the preceding word $P(w_n \mid w_{n-1})$ i.e. Markov Assumption [17].

$$P(w_n \mid w_1^{n-1}) \approx P(w_n \mid w_{n-1}) \tag{11}$$

The Maximum Likelihood Estimate (MPE) to learn the parameter is by taking the counts for each pair and then normalizing it.

$$P(w_n \mid w_{n-1}) = \frac{count(w_{n-1}, w)}{\sum_w count(w_{n-1}, w)} \quad (12)$$

5. Dataset

We have chosen two datasets in which we want to implement our interested algorithms.

1. News Group (preprocessed) - 20 class, 11,314 training documents and 7,532 test documents [18]

2. Personalized Medicine: Redefining Cancer Treatment - 9 classes, 3,320 training documents and 5,667 testing document [19]

6. Implementation and Challenges

6.1. TF-IDF

For TF-IDF, the number of time each a word appears in each document is needed. To obtain the TF-IDF, first a matrix with the size of number of documents by length of vocabulary. The matrix contains a value of one for each word that appears in each document. Once the matrix was created, the number of times the word appears across all documents as well as the number of documents the word appears in. With the two sums for each word, the TF-IDF is calculated using

$$TF - IDF = TF * log(\frac{N}{DF}) \quad (13)$$

where TF is term frequency, N is the total number of documents and DF is document frequency

This is then multiplied to the matrix created before to get the term frequency inverse document frequency. With this, each word, that appears in a document, should have the same value for all documents. TF-IDF was also slightly modified to create a matrix of vocabulary by class. In this class N and DF represented the total number of documents in that class and document frequency of that class. The TF-IDF has been completed for the newsgroup dataset. TFIDF for the personalized medicine dataset was created using the MATLAB built in function but could be implemented the same way it was for News Group dataset once the bag of words model was created.

The equations used are from [15], [16] and [17]

6.2. Naive Bayes with Bag-of-Words Model

The Naive Bayes implementation challenges were with improving CCR. A regularization term was added to all word counts to account for words that occured zero times. This regularization term helped increase the overall CCR. For the news group data set, removing more stop words and incorporating TF-IDF to remove non-significant words did not improve the overall CCR, which has to do with over filtering the data. Therefore, using the news group data set as is provides the best CCR.

As for the personalized medicine dataset, a lot of preprocessing is needed. Due to the large size of the dataset and unique vocabulary, removing stop words and using TF-IDF is essential. To remove stop words, we are using three different stop lists, one provided by Professor Ishwar, MATLAB's built-in stop words, and the "Terroir stop word" list. To improve the CCR even further more words were filtered out based on frequency and TFIDF value. If a word appeared less than 15 times across all documents or had a TFIDF below 1, those words were removed from the vocabulary. With the removal of additional words using TFIDF, the overall CCR would increase.

6.3. Maximum Entropy Classifier

Logistic Regression implantation took more computational time due to no close form to determine minimum. Therefore, we had to tune the gradient descent formula to determine the best parameters to obtain convergence. A challenge we will be facing with logistic regression is finding the location at which it is optimized. We could increase the number of iteration to a very large number until it is found with a small Eta, or create a condition which has to be meant to be considered optimized.Once this is done, we will be removing stop words as well as implementing TF-IDF in the logistic regression algorithm. For both datasets, they were initialized to run on different combinations of parameters for a set number of cycles. Once this was completed, the best combination which lead closer to convergence was used. Once these parameters were obtained, we ran an while loop to find when the objective function(ob) converges to a minimum. We determined that if $|ob(t) - ob(t-1)| < 10^{-4}$ then the objective function had converged.

6.4. Sensing Aware Kernel SVM

A computational concern for the sensing kernel (8) is towards the size of the factorials, since W is usually a large constant and $N_{i,j}$ varies among documents. [13]

Implemented Kernels: According to [13], as approximation forms of the sensing kernel that addresses the computational concern, the sensing 2 kernels generates relatively accurate results amongst all kernel approximations mentioned, we

choose to implement sensing 1 and 2 kernels as approximation forms to the original kernel proposed by [13] to avoid numerical and unbalanced number of words per document issues.

Sensing 1 kernel:

$$K_N^1(x^i, x^j) = log\left\{ \prod_{w=1}^{W} \frac{\Gamma(n\bar{x}_w^i + n\bar{x}_w^j + 1)}{\Gamma(n\bar{x}_w^i + 1)\Gamma(n\bar{x}_w^j + 1)} \right\} \tag{14}$$

where:

$n =$ a tuning parameter set as some constant
$\bar{x}^{i,j}$ =word occurrence from re-sampling i,j-th document

The sensing 1 kernel gets rid of varying N and large constant W from the equation and replace as n, a tuning parameter constant through out.

Sensing 2 kernel: $K_N^2(x^i, x^j) = log(K(\hat{x}_N^i, \hat{x}_N^j))$ (15)

where:

$N =$ a tuning parameter defining unified document length
$\hat{x}^{i,j}$ =word occurrence from resampling i-th document

And:

$$K(x^i, x^j) = \prod_{w=1}^{W} \frac{\Gamma(x_w^i + x_w^j + 1)}{\Gamma(x_w^i + 1)\Gamma(x_w^j + 1)} + \alpha \tag{16}$$

where:

$\alpha = \frac{\Gamma(N_i+1)\Gamma(N_j+1)}{\Gamma(N_i+N_j+W)}$
$i, j \in$ (1,...,N)

The sensing 2 kernel addresses numerical issues by using Gamma function which avoids digit precession problem when numbers get very small and changes multiplications into additions to speed up computation process.

Implementation: The sensing 1 and 2 kernels from [13] is implemented on MATLAB 2017b environment. The implementation uses *libsvm* functions for custom kernel and involves 5-fold cross validation to choose the best tuning parameter N. All of the chosen datasets classifies data points into multiple classes, which binary classification can not be applied. One versus all (OVA) and one versus one (OVO) classification methods are both implemented to handle multi-class classification.

Included as base line comparisons, Linear and Radial Basis Function (RBF) kernels are trained and tested on the same datasets with exaustive 5-fold cross validation to choose the best boxcostraint and sigma tuning parameters.

Challenges: The sensing 2 kernel implementation pairing with the libsvm [20] function 'svmtrain' is not delivering the same result as [13] on the same dataset [18]. This might be the result of false implementation of the sensing 2 kernel function.

The sensing 1 kernal implemented with libsvm [20] while providing similar results on the same dataset [18] as stated in [13] is taking a very long time for the training and testing process. The same computational challenge also applies to the implementation of sensing 1 kernal. Accroding to kernel functions (14) and (16), dot product between x and x' matrix is not appear to be avalible. From our implementation, the kernel function has to iterate through all samples and thus creates heavy computation tasks.

6.5. Naive Bayes using Bigram Model

Implementation of bigrams needs a lot of preprocessing and the bag of words data set that's provided for Matlab. The preprocessing is done in python using sklearn library. The process of preprocessing is shown in Algorithm 1.

Suppose there are total n unique words in all the documents, with m classes.

Algorithm 1 Preprocessing for Bigram Model

procedure BIGRAM DOCUMENT PREPROCESSOR
Document Loop:i = 1:number of all documents
  document$(i) \leftarrow$ remove non-letters/numbers chars
  document$(i) \leftarrow$ convert all letters to lower case
Word Loop:j = 1:number of word in document(i)
  if $Word(j) : characterlength > 2 \,\&\, \in Dictionary \,\&\, ! \in stoplist$ **then**
    Get word root using **Porter Stemmer**
  else
    Discard word;
  goto *Word Loop*.
  goto *Document Loop*.

For Training, we take documents of each class and try to find the occurrences of pair of words (in order) and make a n $\times$ n $\times$ d matrix with the probability of each pair (row,column) using the equation (12). For Testing we again take each pair form the test document and use Naive Bayes for each class using the probability of pairs calculated for training and make decision that maximize the probability.

The pre-processed data is saved as a .json file and used to perform Naive Bayes. The counts for each word and each pair of words is ready now calculating probability. As not all the words will not be available for all classes there will be a lots of NaNs in the probability matrix. This is solved by regularizing data matrix by adding $\frac{1}{w}$ where w equals to number of words in the vocabulary.

7. Experimental Results

7.1. Results from News Group 20 dataset

	Linear Kernel(OVA)	Linear Kernel(OVO)	RBF Kernel(OVO)
CCR (%)	75.83	72.13	72.17
Avg. Recall (%)	75.92	72.62	72.53
Avg. Precision(%)	75.07	71.39	71.43

Table 1. Results for Linear and RBF kernels with 5-fold cross validation

The performance results from Table 1 are generated from fine tuned models with 5-fold method through an exhaustive search. Every class pair for OVO or class for OVA is using its unique best trained parameter in the training model.

	Sensing 1 Kernel	Sensing 2 Kernel
CCR (%)	73.27	56.58
Avg. Recall (%)	N/A	47.37
Avg. Precision(%)	N/A	8.49

Table 2. Results for approximation sensing-aware kernels for 2 classes

Table 2 shows the results of our implementation of sensing kernels on New Group dataset [18]. While Sensing 1 kernel shows a comparable results as [13], Sensing 2 kernel performance results disagrees with the stated. This might be resulted from our failed implementation of the kernel. Thus sensing 2 kernel is not included for testing on the Personalized Medicine dataset.

	N-gram	Logistic Regression	Naive Bayes
CCR (%)	63.06	71.05	78.52
Avg. Recall (%)	64.79	70.15	78.31
Avg. Precision(%)	62.88	71.09	77.99

Table 3. Results for N-gram, logistic regression, and Naive Bayes

Originally the vocabulary for Newsgroup was around 65,000 words but for bigrams when we calculate probability for each class we get a $65000 \times 65000 \times 20$ matrix which is not sparse as we have initialized it by 1/65,000 as regularization. To reduce the computation, using some standard preprocessing techniques (using scikit learn library) the vocabulary size was reduced to around 17,000 words which is substantially less than what we have used for Logistic Regression and Naive Bayes. Due to the reduced vocabulary, some important words seem to be lost due to which the CCR for bigram is reduced.

7.2. Results from Personalized Medicine dataset

When it says unfiltered that refers to the dataset when only stop words are removed. When the dataset refers to filtered that is when additional filtering was applied and least frequent words were removed along with the TF-IDF threshold of 1. All results are using filtered dataset unless noted.

	Linear Kernel	Sensing 1 Kernel
CCR (%)	59.51	67.66
Avg. Recall (%)	58.10	66.87
Avg. Precision(%)	44.86	67.16

Table 4. Results for kerneled SVM

The results shown in Table 4 clearly shows a superior performance for the Sensing 1 kernel which agrees with the performance results from the News Group dataset.

We were unable to make the code run for bigrams for

	Logistic Regression	Naive Bayes
CCR (%)	61.14	Unfiltered: 59.78 Filtered: 61.14
Avg. Recall (%)	51.48	53.10
Avg. Precision(%)	48.03	59.65

Table 5. Results for Logistic Regression and Naive Bayes

Personalized Medicine dataset as after removing some stop words and words that we got based on TF-iDF the size of vocabulary was still around 240,000. This means that the probability matrix will be a $240{,}000 \times 240{,}000 \times 9$ non-sparse matrix and it is out of bound for Matlab.

The shown results are within our expectations in relative to the algorithm performance from News Group datasets due to the extensive unique medical terminology which occurs in the Kaggle Personalized Medicine dataset [19].

7.3. Confusion Matrix for Personalized Medicine dataset

		Y Predict								
		1	2	3	4	5	6	7	8	9
Y Label	1	57	4	1	15	5	3	1	0	0
	2	3	19	0	2	0	2	12	1	0
	3	1	1	1	0	0	1	3	0	0
	4	18	2	1	42	6	2	1	0	1
	5	7	2	0	1	13	0	3	0	1
	6	3	3	0	1	0	12	2	0	0
	7	5	15	4	4	1	2	79	1	0
	8	0	0	0	0	0	0	0	0	0
	9	0	0	0	0	0	0	0	0	4

Table 6. Confusion matrix for personalized medicine dataset

The personalized medicine dataset class labels:

1. Likely Loss-of-function
2. Likely Gain-of-function
3. Neutral
4. Loss-of-function
5. Likely Neutral
6. Inconclusive

7. Gain-of-function

8. Likely Switch-of-function

9. Switch-of-function

The confusion matrix from the personalized medicine shows that there are two pairs of classes that are confused the most. That is due to the fact that the labels for the classes are very similar. For example, class 1 and 4 refer to likely loss of function and loss of function respectively. These classes are very similar and could have minor difference. The same goes for class 2 and 7. Therefore the number of classes was reduced to 5 combining the similar classes to improve CCR. Using a Naive Bayes model, classes 1 and 4, 2 and 7, 3 and 5 and 8 and 9 were combined. This improved CCR to 71.72 which is about 10 higher than the CCR for when there were 9 classes (Appendix A).

8. Conclusion

In this project, we explore different document classification algorithms including Naive Bayes, Logistic Regression, kerneled SVM (Aware-Sensing, Linear, RBF), and n-grams models. Each of these algorithms is tested on 2 datasets: newsgroup dataset and Personalized Medicine: Redefining Cancer Treatment dataset (Kaggle project). Each of the algorithms is taken into account of TF-IDF to give each word value of importance. From the results we obtained, for the personalized medicine dataset Sensing Aware 1 Kernel is the best classifier. Unfortunately, we do not know how our results compare to the Kaggle project due to the results being private.

To further improve our project, we want to take a deeper look into the preprocessing of the personalized medicine dataset to hopefully increase the accuracy. If the preprocessing is done right then we can probably make the bigram code work in MATLAB. Another method we could use for classification is exploring the latent state in Word2Vec, as this transforms words to a vector space. As well as, use optimization techniques to find convergence for logistic regression. We also want to optimize our code Implementation to improve computation efficiency.

We leant from this project that text classification for medical/clinical literature is intriguing due to minor differences in categories which can make a big different in the medical world if it is wrongly classified.

Division of Labor

- Arjun Patel
  - Logistic Regression
  - TF-IDF
- Harshil Prajapati
  - Pre-Processing of Personalized Medicine: Redefining Cancer Treatment Dataset
  - N-gram
- Bowen Song
  - Naive Bayes
  - Sensing Aware Kernel SVM

References

[1] D. S. Sachan, M. Zaheer, and R. Salakhutdinov, "Investigating the working of text classifiers," *arXiv preprint arXiv:1801.06261*, 2018.

[2] P. Maes, "Agents that reduce work and information overload," in *Readings in Human–Computer Interaction*. Elsevier, 1995, pp. 811–821.

[3] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, "Indexing by latent semantic analysis," *Journal of the American society for information science*, vol. 41, no. 6, p. 391, 1990.

[4] T. Hofmann, "Probabilistic latent semantic analysis," in *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., 1999, pp. 289–296.

[5] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of machine Learning research*, vol. 3, no. Jan, pp. 993–1022, 2003.

[6] Y. Kim, "Convolutional neural networks for sentence classification," *arXiv preprint arXiv:1408.5882*, 2014.

[7] B. Parlak and A. K. Uysal, "On feature weighting and selection for medical document classification," in *Developments and Advances in Intelligent Systems and Applications*. Springer, 2018, pp. 269–282.

[8] M. K. Elhadad, K. M. Badran, and G. I. Salama, "A novel approach for ontology-based feature vector generation for web text document classification," *International Journal of Software Innovation (IJSI)*, vol. 6, no. 1, pp. 1–10, 2018.

[9] J. Fürnkranz, "A study using n-gram features for text categorization," *Austrian Research Institute for Artifical Intelligence*, vol. 3, no. 1998, pp. 1–10, 1998.

[10] P. Heroux, S. Diana, A. Ribert, and E. Trupin, "Classification method study for automatic form class identification," in *Pattern Recognition, 1998. Proceedings. Fourteenth International Conference on*, vol. 1. IEEE, 1998, pp. 926–928.

[11] P. Lauren, G. Qu, F. Zhang, and A. Lendasse, "Discriminant document embeddings with an extreme learning machine for classifying clinical narratives," *Neurocomputing*, vol. 277, pp. 129–138, 2018.

[12] Z. F. Salih and S. Tiun, "Term expansion and powerlabel set for multi-label hierarchical on short document classification," *International Journal of Applied Engineering Research*, vol. 13, no. 1, pp. 539–544, 2018.

[13] W. Ding, P. Ishwar, V. Saligrama, and W. C. Karl, "Sensing-aware kernel svm," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, 2014, pp. 2947–2951.

[14] H. Schütze, C. D. Manning, and P. Raghavan, *Introduction to information retrieval*. Cambridge University Press, 2008, vol. 39.

[15] P. Ishwar, "Ec 503 - learning from data," January - May 2018.

[16] F. Song and W. B. Croft, "A general language model for information retrieval," in *Proceedings of the eighth international conference on Information and knowledge management*. ACM, 1999, pp. 316–321.

[17] D. Jurafsky and J. H. Martin, *Speech and language processing*. Pearson London:, 2014, vol. 3.

[18] K. Lang, "Newsweeder: Learning to filter netnews," in *Proceedings of the Twelfth International Conference on Machine Learning*, 1995, pp. 331–339.

[19] "Kaggle data: Personalized medicine: Redefining cancer treatment," https://www.kaggle.com/c/msk-redefining-cancer-treatment.

[20] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011, software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

Appendices

A. Additional Confusion Matrix Results

		Y Predicted				
		1	2	3	4	5
Y Label	1	121	17	15	6	0
	2	12	124	11	0	0
	3	6	1	25	0	0
	4	5	5	2	10	0
	5	0	2	0	0	6

Table 7. Confusion Matrix for reduced personalized medicine dataset

The reduced personalized medicine dataset class labels:

1. Likely Loss-of-function AND Loss-of-function
2. Likely Gain-of-function AND Gain-of-function
3. Neutral AND Likely Neutral
4. Inconclusive
5. Likely Switch-of-function AND Switch-of-function