DreamNLP: Novel NLP System for Feature Extraction and Selection based on a Modified Count Sketch Algorithm

Sanghyun Choi
December 2016

Department of Computer Science
Johns Hopkins University
3400 North Charles Street
Baltimore MD 21218

This project fulfills a requirement for the Computer Science MSE degree.

Advisor’s signature: Date: 12/07/2016
Advisor’s Name: Michael A. Jacobs

Co-Advisor’s signature: Date: 12/07/2016
Co-Advisor’s Name: Vladimir Braverman
Abstract

We propose novel methodologies for feature extraction and selection using data from the natural language processing research using the Count Sketch streaming algorithm (CSSA). The CSSA requires very low memory compared to conventional approaches and can be scaled to very large datasets of interest. For feature extraction, we demonstrate how potential breast diagnosis features could be obtained from Breast Imaging Reporting and Data System (BI-RADS) reports. BI-RADS reports are ideal for feature extraction since they contain structured information on each patient’s diagnosis results. To extract features, we first generate a dictionary of frequently occurring terms from the dataset. Next, we evaluate the terms based on how well they are distributed across different texts using Shannon entropy. We then achieve better understanding of the words or phrases within the texts by tracing back the terms to their actual usage in the texts. Based on such analysis, we determine whether a particular term is associated with extractable feature. For feature selection, which reduces the dimensionality of feature vectors to achieve better computational efficiency and less redundancy of features, we compare the frequent terms for two classes of texts and select the statistically significant terms based on p-value tests. Based on our experiments with the OSHUMED corpus, we show that certain categories can be classified with sensitivity and specificity of 93.6% and 95.2%, respectively, with just 5.4% of baseline number of dimensions.
# Table of Contents

1 Introduction.............................................................................................................................................. 1

2 Background ................................................................................................................................................. 4

2.1 Count Sketch Algorithm ...................................................................................................................... 4

2.2 Shannon Entropy .................................................................................................................................... 5

2.3 Feature Selection .................................................................................................................................... 5

2.3.1 Document Frequency ....................................................................................................................... 6

2.3.2 Chi-Square ......................................................................................................................................... 6

2.3.3 Information Gain .............................................................................................................................. 7

3 Methods .................................................................................................................................................... 8

3.1 Feature Extraction .................................................................................................................................. 8

3.2 Feature Selection .................................................................................................................................. 10

4 Experiments ............................................................................................................................................... 12

4.1 Feature Extraction ................................................................................................................................. 12

4.2 Feature Selection .................................................................................................................................. 13

4.2.1 Training/Testing Data ....................................................................................................................... 13

4.2.2 HHD Feature Selection .................................................................................................................. 14

4.2.3 Feature Vectors Construction .......................................................................................................... 14

4.2.4 Machine Learning Classifiers ......................................................................................................... 15

5 Results ....................................................................................................................................................... 17

5.1 Feature Extraction ............................................................................................................................... 17

5.2 Feature Selection .................................................................................................................................. 18

6 Discussion ................................................................................................................................................ 20

7 Conclusion ................................................................................................................................................. 22
1 Introduction

With the explosive growth of text data, automatic text classification using computers has become an increasingly active field of study [1, 2]. One of the most well-known problems in this area is the automatic classification of emails into spam or ham [3]. Major internet companies such as Google and Yahoo provide automatic email classification service which greatly reduces the amount of time we waste on unwanted emails and the costs resulting from possible frauds. Google’s email service can even classify ham emails into more specific predefined categories such as “social” or “promotions” [4]. Hence automatic text classification allows us to quickly sort a large collection of texts into desired categories. This benefit can be further leveraged by performing independent study on each category and discovering valuable information that is inherent to a particular category.

The most common approach to tackling automatic text classification problems is to use machine learning [5]. In the context of text classification, machine learning involves “training” a statistical model with texts that are known to belong to a certain class so that the model can automatically classify unseen texts based on trained texts. Since the input to machine learning models take the form of numerical vectors, one of the main challenges is to find a vector representation of the input texts. The numerical vectors are called feature vectors because each component in the vector corresponds to a quantized feature of the input data. Feature vectors can be constructed through a process called feature extraction, which involves obtaining a set of attributes from texts (e.g. e-mails, news articles, medical reports, Twitter posts, etc.) that can be assigned quantitative values. For example, the bag-of-words model is a widely used feature extraction technique which simply uses a set of words from texts as attributes and computes their frequencies to construct feature vectors [6]. But using bag-of-words model often results in high-dimensional vectors, which introduces problems such as bias, noise, computational
overhead, and the “curse of dimensionality” [7-9]. To mitigate these problems, *feature selection* can be applied to filter out redundant terms that may contribute to noise and leave only the most informative terms for feature vectors construction. As a result, the dimensionality of feature vectors is reduced and the curse of dimensionality, which occurs with high-dimensional vectors, can potentially be avoided.

Feature extraction from texts is a non-trivial task because it is difficult to know which set of attributes could be obtained without a strong prior knowledge of the contents of the texts. Hence our hypotheses is that observation that one of the simplest ways to gain a basic understanding of large corpora of text is to extract and analyze frequently occurring terms in the texts. Our proposed methodology starts from this premise. For example, suppose there are word clouds, consisting of frequent terms. If defined correctly, then they can be effective in visualizing and summarizing particular texts [10]. Other well-known methods such as *TileBars* and term-topic matrix also visualize term frequency to summarize texts [11, 12].

Yet, finding frequent terms from large corpora is a challenging task by itself since every unique term in the text has to be stored in memory if approached in a naïve way. We address the memory issue by applying Count Sketch streaming algorithm [13]. Studies indicate robustness of sketch algorithms in the context of large scale Natural Language Processing (NLP) tasks [14-16]. After finding the frequently occurring terms, we use Shannon entropy to quantitatively evaluate how well the terms in the dictionary are distributed across all the texts [17]. Shannon entropy measures the “amount of information required to describe a random variable” and it can used to evaluate the homogeneity of event distribution [18]. To assess whether a term is associated with extractable feature, we trace back the terms to some of the texts they belong to and observe their context with the aid of automatically highlighted terms.

If the bag-of-words model is used for text representation, feature selection on this model
requires a memory-efficient approach as well since it involves analyzing all possible unique terms in the large set of texts. In this regard, we propose a novel feature selection method for binary text classification based on heavy hitters analysis. Informally, heavy hitters are elements that occur frequently given a particular set of elements [19]. Hence, our method is called the Heavy Hitters Difference (HHD) method. Compared to other feature selection methods, HHD method offers the following combined advantages:

- **Scalability to very large data**: HHD makes use of Count Sketch streaming algorithm to initially generate a dictionary of candidate terms for feature selection. The key characteristic of the algorithm is its low memory usage.

- **Use of robust terms attributes**: For small feature vectors, it has been shown that term frequency outperforms document frequency [20]. HHD captures term frequency as well as collocation of words which may be more informative than simple unigram words.

- **Simple term selection metric**: Only term frequencies and \( p \)-values are required to determine whether a term should be selected as feature.

In summary, we present DreamNLP (Data Streaming Count Sketch NLP), a novel system to analyze text records for feature extraction and feature selection of words and related structures. This paper is thus organized into two main parts, feature extraction and selection, for each subsequent sections. Although the methods and experiments for feature extraction and selection are independent from one another, the implementations of feature extraction and selection are both tied into a single piece of user-interactive software implemented using C++ and QT framework, called the DreamNLP software (see ‘Appendix’ section).
2 Background

2.1 Count Sketch Algorithm

The basic Count Sketch algorithm (CSA) solves the problem of finding top-k most frequent items in data stream [13]. Compared to na"ive approach to this problem which implies keeping track of each item’s frequency, CSA requires relatively low memory. Although it is a randomized approximate algorithm, (i.e. outputs good approximation with high probability), the approximation level and probability of success can be controlled via input parameters.

“Count Sketch” is simply an $h \times b$ matrix $C$ of counters that can be viewed as $h$ hash tables with $b$ buckets for each hash table. For a particular choice of $k$, the parameters $h$ and $b$ play an important role in determining the accuracy of estimated frequencies.

Suppose we have a stream of items $\{o_1, \ldots, o_n\}$, where $n$ is the total number of items in the data stream. Also, let $\{f_1, \ldots, f_h\}$ and $\{g_1, \ldots, g_h\}$ be two sets of $h$ hash functions that map $o_l$ to a particular value in $\{1, \ldots, b\}$ and $\{-1,1\}$, respectively, where $o_l$ is some $l$-th item in the stream. The Count Sketch algorithm involves two operations:

- Updating the counters:
  
  For each $i$-th row in matrix $C$, update counter in $f_i(o_l)$-th column by adding $g_i(o_l)$.

- Estimating the frequency:
  
  For each $i$-th row in matrix $C$, compute $C_{ij}g_i(o_l)$, where $j=f_i(o_l)$ as above, and estimate the frequency of $o_l$ by their median.

Based on these two operations, the Count Sketch algorithm outputs an estimate of the frequencies for top $k$ items. In order to make Count Sketch applicable to our context (i.e. stream of terms from texts), we modified the algorithm to use MurmurHash. MurmurHash is a robust string hashing algorithm that was used to design the hash functions $f$ and $g$ [21].
2.2 Shannon Entropy

Shannon entropy is defined as follows (denoted by $H$) [17]:

$$ H = - \sum_{i=1}^{n} (p_i \log p_i) $$

Here $n$ is the number of possible events and $p_i$ is the probability that $i$-th event occurs. If $H$ attains its maximum value, it implies that all events are equally likely (i.e. $p_i = 1/n$ for all $i$).

In the context of our feature extraction method, $i$-th event corresponds to particular a term appearing in $i$-th text when there are $n$ texts. Therefore, if the Shannon entropy for a particular term $t$ (i.e. $H_t$) is high, it implies that $t$ appears in most of the texts. Where, a low value of entropy implies that $t$ only appears in a small subset of texts. Based on this property, $H_t$ can be used as an assessment of term $t$’s suitability for feature extraction of terms. That is, assuming $t$ is associated with some feature, we want $t$ to be present in majority of texts in equally distributed manner.

$H_t$ can be obtained by computing the values of $p_i$ as “$n_i / n_{\text{total}}$” where $n_i$ is the number of occurrences of $t$ in the $i$-th text and $n_{\text{total}}$ is the total number of occurrences of $t$ in the entire set of texts. Since there are at most $k$ terms under consideration after finding the heavy hitters with Count Sketch algorithm, we can compute exact values of $n_i$ without any memory issue.

2.3 Feature Selection

The HHD method employs a filtering based approach to select a subset of terms from the complete set of distinct terms existing in the pool of texts under analysis and used by many feature selection methods [22]. The filtering is done by quantitatively scoring each term using a particular metric and then choosing only those terms that pass a certain threshold score. For example, after computing the frequency of each term, terms that do not exceed the minimum
frequency can be filtered out. Besides filtering methods, there are two other main approaches to feature selection, they are the wrappers method and embedded method [22, 23]. The wrappers method uses a brute-force approach to determine which subset of features yields the optimal classification performance. The embedded method selects features as part of the machine learning model’s training algorithm. Although these approaches use more sophisticated term selection techniques, they are computationally more expensive than the filter method [22-24]. There are many different feature selection methods that exist, such as Information Gain, Mutual Information, Chi-Square, Probability Proportional Difference, Document Frequency, etc. and we will describe three of these methods which are widely used and presented in [25]. Interested readers may refer to some of the feature selection surveys for a more comprehensive overview [25-27].

2.3.1 Document Frequency

Document frequency (DF) is one of the simplest methods for feature selection [28]. For each distinct term in the collection of documents, the document frequency is computed by counting the number of documents in which the term appears. If the document frequency for a particular term does not exceed a certain threshold value, then the term is discarded from the feature set. Therefore, the DF method is based on the assumption that the terms which occur in a large number of documents are the most informative features.

2.3.2 Chi-Square

The Chi-Square method is based on Pearson’s chi-squared test [29]. The test statistic for Pearson’s chi-squared test is given as:

$$\chi^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i}$$
Given $n$ possible states of underlying events (i.e. events to test for independence), $O_i$ and $E_i$ are the observed frequency and the expected frequency of some $i$-th state of events, respectively. Hence $\chi^2$ measures the degree to which observed frequencies deviate from expected frequencies. Since expected frequencies are computed based on the assumption that the underlying events are independent of each other, a high value of $\chi^2$ implies a high probability of dependence.

In the context of Chi-Square feature selection method, the two underlying events are occurrence of term $t$ and occurrence of class $C$. The four possible states of these events are $t$ and $C$ occurring together, both not occurring, and only one of them occurring. Therefore, given a particular class, a high value of $\chi^2$ for some term indicates that the term could potentially serve as a good feature since there is a high chance of the term being dependent on the class.

### 2.3.3 Information Gain

Given a particular term $t$, the Information Gain (IG) feature selection method computes how much information, in terms of number of bits, could be gained for category prediction when $t$ is present and absent. It is based on the Shannon entropy formula. If we let $C_i$ denote the $i$-th class, the multi-class IG formula for term $t$ is given as [28, 30]:

$$
\text{IG}(t) = - \sum_{i=1}^{m} P(C_i) \log P(C_i)
+ P(t) \sum_{i=1}^{m} P(C_i|t) \log P(C_i|t)
+ P(\bar{t}) \sum_{i=1}^{m} P(C_i|\bar{t}) \log P(C_i|\bar{t})
$$
Based on this formula, it is possible to quantitatively assess the degree to which a term contributes to the predictability of text categories. By discarding terms with low information gain, only the most informative terms will be left in the feature set.

3 Methods

3.1 Feature Extraction

DreamNLP pipeline is demonstrated in figure 1 where a set of reports are analyzed such that potential features could be extracted from them. The reports can be viewed as a data stream of terms (in this paper, a “term” could mean a single word or a sequential set of words) to our software platform. With very low memory usage, the DreamNLP finds top-\(k\) most frequent terms (i.e. heavy hitters) using a modified Count Sketch streaming algorithm.

Upon obtaining the heavy hitters, they are evaluated using Shannon entropy. A high value of Shannon entropy for a particular term indicates that the term is present in majority of reports. In such case, the term is considered as a good candidate for feature extraction since our ideal goal is to be able to extract the same features for every report. For visual analysis of term distribution, the DreamNLP software provides a multidimensional histogram to easily determine the relationship between frequencies, terms, and reports and shown in figure 2.

The final steps of the DreamNLP pipeline involves better understanding the context of each term. Such steps are performed to qualitatively assess whether the terms with high Shannon entropy values are associated with extractable features. Using the DreamNLP software, it is possible to view how each term is used in the actual reports through simple user interactions. Moreover, the software is capable of generating many different outputs for further data analysis. For example, the amount of words to the left and right of a term can be configured to extract phrases involving the terms. A complete description of the DreamNLP software can
be found in the ‘Appendix’ section.

Figure 1: An overview of the DreamNLP feature extraction pipeline.

Figure 2: A multidimensional histogram showing the relationship between frequencies, terms, and reports
3.2 Feature Selection

The HHD feature selection method, outlined in algorithm 1, requires $O(hN)$ time and $O(k+hb)$ space, where $h$ is the number of hash tables for Count Sketch data structure, $k$ is the number of heavy hitters, and $N$ is the total number of terms in training data. Each “term” in the algorithm can be composed of any desired number of sequential words (e.g. unigram, bigram, trigram, etc.). The role of the first step is to filter out low frequency terms which may contribute to noise and to obtain terms that represent each class well. For example, we can often conclude that an e-mail is spam if its content contains certain keywords like “Free” and “Viagra” because these words appear very often in spam e-mails. To extract potential candidates, we use Count Sketch algorithm to find top-$k$ most frequently occurring terms for each class of texts with very low memory requirement. The results are then combined into a single set of distinct terms, $T$.

In the second step, another full pass over each class of texts is performed to compute the frequency of each term in $T$. Although this second pass is necessary to compute exact values of frequencies, it can be skipped entirely by tolerating some approximations. The modification is further described in the ‘Discussion’ section but the experimental results described in this paper are based on executing this second step.

The third step selects the final candidates for feature vector construction. The intuition behind this step is that if the probability of term $t$ occurring in the positive class ($C^+$) is significantly different to that of the negative class ($C^-$) then this difference can be exploited to distinguish the two classes. The following hypothesis test is used to determine whether the probabilities are different:

$$H_0: P(t|C^+) = P(t|C^-) \text{ versus } H_1: P(t|C^+) \neq P(t|C^-)$$

If the null hypothesis is rejected for some $t$, we conclude that $P(t|C^+) \neq P(t|C^-)$ and use $t$ as one
of the features. Once we obtain the final set of terms to use as features, we represent each text as a feature vector composed of probabilities (i.e. term frequencies) of each term in the set.

Algorithm 1: Heavy Hitters Difference (HHD) feature selection method

**Inputs:**
- Set of texts labelled as positive ($D^+$) or negative ($D^-$)
- Number of heavy hitters for Count Sketch ($k$)
- Number of hash tables for Count Sketch ($h$)
- Number of buckets for each hash table in Count Sketch ($b$)
- Desired confidence level for $p$-value test ($\alpha$)

**Output:**
- Set of terms (i.e. dictionary) to be used as features ($S$)

// Step 1: Find top-$k$ most frequently occurring terms for each class of texts

$N^+, N^- \leftarrow 0$ // Number of terms in $D^+$ and $D^-$, respectively

Initialize two Count Sketch data structures ($CS^+$ and $CS^-$) with $k$, $h$, and $b$

for each term ($t$) in $D^+$:
  
  if $t \not\in \{ \text{stop words } \}$:
    
    $N^+ \leftarrow N^+ + 1$

    Update $CS^+$ with $t$
  
end

for each term ($t$) in $D^-$:

  if $t \not\in \{ \text{stop words } \}$:
    
    $N^- \leftarrow N^- + 1$

    Update $CS^-$ with $t$

end

$T^+ \leftarrow$ Set of top-$k$ most frequently occurring terms from $CS^+$

$T^- \leftarrow$ Set of top-$k$ most frequently occurring terms from $CS^-$

$T \leftarrow$ Combined set of terms from $T^+$ and $T^-$ with duplicate terms removed

// Step 2: Compute frequency of each term in $T$ with respect to each class of texts

$N^+_t, N^-_t \leftarrow 0$ for all $t \in T$ // Frequency of term $t$ in $D^+$ and $D^-$, respectively

for each term ($t$) in $D^+$:

  if $t \in T$:
    
    $N^+_t \leftarrow N^+_t + 1$

end

for each term ($t$) in $D^-$:

  if $t \in T$:
    
    $N^-_t \leftarrow N^-_t + 1$

end
4 Experiments

4.1 Feature Extraction

Following the pipeline described in figure 1, we tried to extract useful features from clinical reports of patients who underwent breast imaging consisting of mammography, ultrasound, and MRI. They were obtained by an approved IRB study following the HIPPA criteria. For this study, the texts were anonymized with no patient health identifiers within the text.

Breast imaging reports are structured and adhere to the Breast Imaging Reporting and Data System (BI-RADS) format [31]. Briefly, the BI-RADs scoring gives a numerical index to the degree of potential malignancy of breast tumors. The scale ranges from 1-6, where is one to three (1- considered negative, 2 and 3 are considered benign and 4-6 is considered highly suspicious (4-5) or proven malignancy (6). Moreover, the right and left breasts are described by defined features, such as breast density, lesion shape, size, and distribution. Each report was converted to a text file and read into DreamNLP system.
4.2 Feature Selection

As outlined in figure 3, we used DreamNLP software to apply the HHD algorithm on a pool of texts reserved for training in order to obtain a dictionary of statistically significant terms. Based on the dictionary, each text was represented as a feature vector and, finally, machine learning classifiers were used to test the robustness of the features selected by HHD algorithm.

Figure 3: An overview of feature selection experiment pipeline

4.2.1 Training/Testing Data

We used the OSHUMED corpus for experimental data. It is a publicly available data consisting of medical abstracts from MEDLINE (a collection of “journal citations and abstracts for biomedical literature”[32]) for years ranging from 1987 to 1991. In our experiment, we used a collection of 34,389 cardiovascular diseases abstracts from year 1991, which were grouped into 23 different categories.

Among the 23 categories, we chose to classify pairs of categories in the OSHUMED
corpus that had sufficiently large number of instances in each category. The pairs were formed within a group of categories with similar number of instances. We randomly split the instances so that 60% and 40% of the instances in each category were reserved as training data and testing data, respectively. The set-up is summarized in table 1.

<table>
<thead>
<tr>
<th>Group</th>
<th>Category Name</th>
<th>Number of Instances</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Training</td>
</tr>
<tr>
<td>1</td>
<td>C01 Bacterial Infections and Mycoses</td>
<td>1,524</td>
</tr>
<tr>
<td></td>
<td>C08 Respiratory Tract Diseases</td>
<td>1,553</td>
</tr>
<tr>
<td></td>
<td>C12 Urologic and Male Genital Diseases</td>
<td>1,511</td>
</tr>
<tr>
<td>2</td>
<td>C04 Neoplasms</td>
<td>3,796</td>
</tr>
<tr>
<td></td>
<td>C14 Cardiovascular Diseases</td>
<td>3,661</td>
</tr>
<tr>
<td>3</td>
<td>C06 Digestive System Diseases</td>
<td>1,793</td>
</tr>
<tr>
<td></td>
<td>C20 Immunologic Diseases</td>
<td>1,870</td>
</tr>
<tr>
<td></td>
<td>C21 Disorders of Environmental Origin</td>
<td>1,760</td>
</tr>
</tbody>
</table>

### 4.2.2 HHD Feature Selection

For each pair of categories, we ran the HHD algorithm based on values of $k$ ranging from 100 to 1,000 in order to observe how output dictionary size, $|S|$, affected classification performance. For $k=100, 250, 500, 750, 1000$, the average dictionary size turned out to be 109, 254, 485, 725, 961, respectively. In addition to these values of $k$, the Count Sketch data structure was initialized with $h=20$ and $b=50,000$ since they produced reasonable approximations for our purpose.

As part of pre-processing, stop words were removed in step 1 of the algorithm but no stemming was performed. The two-tailed hypothesis test in step 3 was done at 95% confidence level (i.e. $\alpha=0.95$).

### 4.2.3 Feature Vectors Construction

Based on the output of HHD algorithm, $S$, each text in training data and testing data were
represented as feature vectors by computing the term frequency of each term in $S$. That is, given a term $t$ from $S$ and a text, its term frequency was computed as $N_t / N$ where $N_t$ was the number of times $t$ occurred in the text and $N$ was the total number of terms in the text. Since the frequency values tended to be very small, we expressed them as percentage values by multiplying 100.

### 4.2.4 Machine Learning Classifiers

Once the feature vectors were constructed, Support Vector Machine and Artificial Neural Network models were used for classification [33, 34]. Both models were shown to perform well on texts classification problems [35]. We used a linear kernel for the SVM model (Matlab 2016A), which produced much better results than using non-linear kernels (e.g. radial basis functions and polynomial kernels) based on our experiments. The ANN model we used was a simple two-layer feedforward neural network implemented using Deeplearning4j, an open-source deep learning library [36].

Neural networks are often considered as “black box” classifiers. There is no single accepted theory on choosing the right model parameters. Therefore, the architecture of our ANN was based on a mixture of various results from relevant studies. For example, we only introduced a single hidden layer, if it was sufficient to make good approximations to functions based on the universal approximation theorem [37, 38]. There are also numerous studies on choosing the number of hidden nodes [39-41]. We decided to use the formula that captures both the dimensionality of feature vectors and the number of training samples defined below [41]:

$$N_h = (N_d + \sqrt{N_{\text{train}}}) / L$$

$N_h$ is the number of hidden nodes, $N_d$ is the dimensionality of feature vectors, which is $|S|$ in our case, $N_{\text{train}}$ is the number of training sample, and $L$ is the number of hidden layers. The final
architecture of our ANN model is illustrated in figure 4.

We also used Multinomial Naïve Bayes (MNB) model based on Weka to produce baseline results [42]. The MNB model is commonly used for text classification problems and is ideal as baseline since it uses minimal feature selection (i.e. employs almost entire set of unique terms in the texts) [43]. For pre-processing, Snowball stemmer and Rainbow stop words removal were applied. All terms were normalized to lower-case and their frequency values were used for feature vectors.

![Figure 4: Architecture of feedforward neural network used in the experiment](image)

Figure 4: Architecture of feedforward neural network used in the experiment
5 Results

5.1 Feature Extraction

DreamNLP was initially applied to a mixture of different types of clinical reports from breast cancer patients with breast imaging for diagnostic purposes. Using DreamNLP, we were able to easily filter out non-BI-RADS reports (which consisted a small portion relative to BI-RADS reports) as they contained terms with low Shannon entropy such as “FDG uptake” and were missing high entropy terms such as “BIRADS” and “assessment”. We were eventually left with 46 instances of BI-RADS reports for experimentation.

Performing Count Sketch on 24,487 terms to find top 100 terms took about 20 ms. Parameters $t$ and $b$ were chosen such that the results closely matched true values (i.e. $t = 20$ and $b = 10,000$). Careful analysis of the 100 terms revealed useful information on how we could potentially extract the BI-RADS reports for specific features. Table 2 summarizes the analysis of the terms that were considered significant based on entropy evaluation and context analysis.

The high entropy values associated with each term indicate that these terms are present in the majority of reports and thus could be used for information extraction. For example, we were able to easily extract “BI-RADS Scores” from each report using the terms “BI-RADS” and “assessment” so that we could classify the reports between malignant benign patients.
Table 2: Feature extraction results for BI-RADS reports

<table>
<thead>
<tr>
<th>Dictionary Generation</th>
<th>Evaluation</th>
<th>Context Analysis</th>
<th>Insight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rank 1</td>
<td>Term(s)</td>
<td>Shannon Entropy</td>
<td>Sample Neighboring Words</td>
</tr>
<tr>
<td>5, 7</td>
<td>right breast, left breast</td>
<td>94.38, 94.42</td>
<td><strong>RIGHT BREAST</strong>: <strong>LEFT BREAST</strong>:</td>
</tr>
<tr>
<td>6</td>
<td>cm</td>
<td>89.33</td>
<td>mass measuring 1.1 x 0.8 x 0.9 cm</td>
</tr>
<tr>
<td>8</td>
<td>mass</td>
<td>83.65</td>
<td>lobulated, spiculated mass</td>
</tr>
<tr>
<td>13, 20</td>
<td>bi-rads, assessment</td>
<td>84.69, 96.19</td>
<td><strong>BI-RADS 5</strong>: Highly suspicious … Assessment: 6 Known malignancy</td>
</tr>
<tr>
<td>41, 95</td>
<td>o’clock, quadrant</td>
<td>72.39, 70.58</td>
<td>at the two o’clock position right upper outer quadrant</td>
</tr>
<tr>
<td>34</td>
<td>history</td>
<td>92.81</td>
<td>strong family history of breast cancer</td>
</tr>
<tr>
<td>54, 74</td>
<td>density, dense</td>
<td>96.11, 88.34</td>
<td>Breast density: Moderately dense.</td>
</tr>
</tbody>
</table>

1: The rank of a term based on its number of occurrences in the entire set of texts (estimated by Count Sketch algorithm)
2: Percentage with respect to maximum possible value

5.2 Feature Selection

Within the scope of our experiments, optimal performance for SVM was achieved when we requested 250 heavy hitters to the Count Sketch algorithm, as shown in table 3. Beyond k=250, the performance generally decreased. As evident from table 4, the ANN classifier generally performed better than SVM and produced optimal results at higher dimensions. Classification of C04 and C14 using ANN yielded the highest values of sensitivity and specificity, which were 93.6% and 95.2%, respectively. Some of the terms included in the dictionary were “breast”, “cancer”, “cell”, “tumor”, “coronary artery disease”, “heart”, “myocardial infarction”, “hypertension”, “blood pressure”, and “left ventricular”.

Table 5 shows the comparison between optimal results of ANN classifier and the baseline results. Except for classifying C06 and C21, ANN classifier outperforms Multinomial Naïve
Bayes classifier with much smaller dictionary size. On average, only 6% of the terms were used for ANN model compared to baseline number of terms.

Table 3: SVM classification results (geometric mean of sensitivity and specificity in %)

<table>
<thead>
<tr>
<th>Group</th>
<th>Classified Categories</th>
<th>Number of Heavy Hitters (k)</th>
<th>100</th>
<th>250</th>
<th>500</th>
<th>750</th>
<th>1000</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>C01, C08</td>
<td></td>
<td>82.0</td>
<td>83.1</td>
<td>82.4</td>
<td>82.9</td>
<td>80.1</td>
</tr>
<tr>
<td></td>
<td>C01, C12</td>
<td></td>
<td>86.7</td>
<td>88.9</td>
<td>88.7</td>
<td>86.8</td>
<td>87.1</td>
</tr>
<tr>
<td></td>
<td>C08, C12</td>
<td></td>
<td>88.7</td>
<td>90.3</td>
<td>89.2</td>
<td>89.9</td>
<td>89.7</td>
</tr>
<tr>
<td>2</td>
<td>C04, C14</td>
<td></td>
<td>91.6</td>
<td>92.8</td>
<td>92.5</td>
<td>92.0</td>
<td>91.0</td>
</tr>
<tr>
<td>3</td>
<td>C06, C20</td>
<td></td>
<td>87.6</td>
<td>89.9</td>
<td>89.9</td>
<td>89.3</td>
<td>89.6</td>
</tr>
<tr>
<td></td>
<td>C06, C21</td>
<td></td>
<td>83.9</td>
<td>87.6</td>
<td>87.1</td>
<td>86.8</td>
<td>85.1</td>
</tr>
<tr>
<td></td>
<td>C20, C21</td>
<td></td>
<td>87.2</td>
<td>90.1</td>
<td>88.6</td>
<td>87.0</td>
<td>86.7</td>
</tr>
</tbody>
</table>

*Higher value that that of k=250 when compared with more than two decimal places.

Table 4: ANN classification results (geometric mean of sensitivity and specificity in %)

<table>
<thead>
<tr>
<th>Group</th>
<th>Classified Categories</th>
<th>Number of Heavy Hitters (k)</th>
<th>100</th>
<th>250</th>
<th>500</th>
<th>750</th>
<th>1000</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>C01, C08</td>
<td></td>
<td>82.8</td>
<td>84.4</td>
<td>85.3</td>
<td>86.7</td>
<td>86.6</td>
</tr>
<tr>
<td></td>
<td>C01, C12</td>
<td></td>
<td>87.5</td>
<td>90.3</td>
<td>91.1</td>
<td>92.1</td>
<td>92.7</td>
</tr>
<tr>
<td></td>
<td>C08, C12</td>
<td></td>
<td>88.4</td>
<td>90.9</td>
<td>91.9</td>
<td>93.1</td>
<td>93.8</td>
</tr>
<tr>
<td>2</td>
<td>C04, C14</td>
<td></td>
<td>91.8</td>
<td>93.0</td>
<td>93.6</td>
<td>94.3</td>
<td>94.4</td>
</tr>
<tr>
<td>3</td>
<td>C06, C20</td>
<td></td>
<td>86.8</td>
<td>90.1</td>
<td>91.5</td>
<td>91.6</td>
<td>92.0</td>
</tr>
<tr>
<td></td>
<td>C06, C21</td>
<td></td>
<td>84.3</td>
<td>86.3</td>
<td>89.0</td>
<td>89.5</td>
<td>88.3</td>
</tr>
<tr>
<td></td>
<td>C20, C21</td>
<td></td>
<td>86.8</td>
<td>89.6</td>
<td>90.7</td>
<td>91.4</td>
<td>91.6</td>
</tr>
</tbody>
</table>

Table 5: Comparison of ANN results to Multinomial Naïve Bayes classification results

<table>
<thead>
<tr>
<th>Group</th>
<th>Classified Categories</th>
<th>Multinomial Naïve Bayes(^1)</th>
<th>Artificial Neural Network</th>
<th>Dictionary Fraction (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Performance(^2)</td>
<td>Size of Dictionary</td>
<td>Performance(^2)</td>
</tr>
<tr>
<td>1</td>
<td>C01, C08</td>
<td>84.7</td>
<td>13,068</td>
<td>86.7</td>
</tr>
<tr>
<td></td>
<td>C01, C12</td>
<td>91.0</td>
<td>13,075</td>
<td>\textbf{92.7}</td>
</tr>
<tr>
<td></td>
<td>C08, C12</td>
<td>92.8</td>
<td>13,429</td>
<td>\textbf{93.8}</td>
</tr>
<tr>
<td>2</td>
<td>C04, C14</td>
<td>94.2</td>
<td>22,331</td>
<td>\textbf{94.4}</td>
</tr>
<tr>
<td>3</td>
<td>C06, C20</td>
<td>90.3</td>
<td>15,223</td>
<td>\textbf{92.0}</td>
</tr>
<tr>
<td></td>
<td>C06, C21</td>
<td>\textbf{90.4}</td>
<td>14,850</td>
<td>89.5</td>
</tr>
<tr>
<td></td>
<td>C20, C21</td>
<td>90.1</td>
<td>15,777</td>
<td>\textbf{91.6}</td>
</tr>
</tbody>
</table>

1: 10-fold cross validation results, 2: Geometric mean of sensitivity and specificity (%)
6 Discussion

Our feature extraction method is highly intuitive since it is based on simple frequent terms analysis. Besides the possibility of obtaining potential features, finding frequent terms can also reveal other useful insights from texts. For example, considering that most of the BI-RADS reports we analyzed were those of malignant cases, the fact that the term “upper outer” appeared among the top 100 terms provides one evidence that the upper outer quadrant of the breast is frequently vulnerable to carcinoma [44].

One obvious drawback of frequent terms analysis is that it might miss out terms which appear consistently throughout many documents but with relatively less frequency than other terms, such as “age” in the case of BI-RADS. However, this could be controlled to a certain extent by increasing the number of top-k terms to find and filtering out more stop words.

Our software-based methodology is open to other possible extensions as well for richer seamless analysis. For example, by applying NegEx algorithm to some of the terms, we are interested in how negated expressions could be used as features for machine learning [45]. Moreover, our methodology can be generalized to analyze other types of documents besides clinical reports.

There were two main reasons why our feature selection method was successful in classifying the categories chosen in our experiment. First, there were many non-trivial terms or phrases that occurred predominantly and commonly in one category of texts compared to another. Second, there were sufficient number of training samples for each category in a pair. Since HHD method is based on the assumption that the terms it has selected will also appear in unseen texts with reasonable probability, having only a few samples would not enable the HHD method to select such terms.
If the two conditions are met, the scalability of HHD method could be leveraged to find robust terms among large collection of texts. The scalability could be improved further by making certain approximations to skip step 2 described in algorithm 1. As shown in algorithm 2, the values of $N_i^+$ and $N_i^-$ can be approximated by frequency values from the Count Sketch data structures $CS^+$ and $CS^-$, respectively. Since Count Sketch holds the top-$k$ most frequently occurring terms, any missing terms are guaranteed to have frequencies less than or equal to the minimum frequency value in Count Sketch. Hence we can make conservative approximations for the missing terms. Such approximations would improve with increasing values of $k$ since the minimum frequency would eventually be very close to 0 for $k$-th ranking term according to Zipf’s law, guaranteeing tighter bound [46]. By skipping step 2, the HHD algorithm can turn into an online feature learning algorithm as long as Count Sketch data structures are kept persistent.

**Algorithm 2: Frequency estimation using Count Sketch results**

```plaintext
for each term (t) in T:
    // Positive class
    if t ∈ $T^+$:
        $N_i^+$ ← Estimated frequency for t from $CS^+$
    else
        $N_i^+$ ← Minimum frequency value from $CS^+$
    end
    // Negative class
    if t ∈ $T^-$:
        $N_i^-$ ← Estimated frequency for t from $CS^-$
    else
        $N_i^-$ ← Minimum frequency value from $CS^-$
    end
end
```

Another key feature of HHD method is its flexibility to incorporate other well-known feature selection methods. We have deliberately chosen a general name for our method for such reason. For example, the term selection method we proposed in steps 2 and 3 of algorithm 1
could be replaced by Chi-Square or Information Gain selection method. However, in such cases, the benefit of obtaining estimated frequency values from Count Sketch data structure cannot be realized.

7 Conclusion

We have demonstrated that DreamNLP can extract breast diagnosis features from radiological clinical reports using key BI-RADS features for improved diagnostic performance. DreamNLP uses very low computational memory and minimal prior knowledge of the structure of texts with high accuracy of correct extraction of terms.

We have also shown that our DreamNLP’s Heavy Hitters Difference feature selection method was capable of learning robust low dimensional feature vectors from the OSHUMED corpus. By training an Artificial Neural Network model based on these feature vectors, we were able to classify 6 of the 7 chosen pairs of categories in the OSHUMED corpus better than the Multinomial Naïve Bayes model with just 6% the size of feature dimension compared to the latter model. Support Vector Machines model also produced comparable results with the baseline with an even fewer number of dimensions.

As future work, we will be applying HHD method on other medical texts such as BI-RADS reports to see how well they can be automatically classified to certain categories (e.g. different BI-RADS scores). We are also interested in experimenting with various ways the HHD algorithm could be modified as described in the previous section.
References


32. NIH. MEDLINE/PubMed Resources Guide. 2016;


38. Sonoda S, Murata N. Neural network with unbounded activation functions is universal approximator. Applied and Computational Harmonic Analysis.


46. Powers DMW. Applications and explanations of Zipf's law. Proceedings of the Joint Conferences on New Methods in Language Processing and Computational Natural Language Learning; 1998; Sydney, Australia.
Appendix – DreamNLP Software Guide

1. Main Features

- **Items distribution analysis**
  This feature is used to answer the following questions:
  - How are the heavy hitters distributed across different files?
  - What are some of the neighboring words of these heavy hitters?

  Refer to section 3 for a complete description.

- **Items comparison analysis**
  This feature is used to answer the following questions:
  - Given two categories of files, how are their heavy hitters different?
  - How can some of the terms be used as features for machine learning?

  Refer to section 4 for a complete description.

2. Common Features

![Figure 1: Main tab of Heavy Hitters tool](image)

28
As shown in figure 1, the main tab (i.e. “Heavy Hitters” tab) is divided into two sections – a section for input and another for output. Using the input section, users should enter or select the following parameters:

- **Number of heavy hitters**: This is the value of $k$ when top-$k$ most frequently occurring items are to be found.
- **Type of data stream**: If the data stream consists of numbers only, select ‘Numbers’. Otherwise, select ‘Words’.
- **Folder(s) containing the data set**: “Primary folder” must be specified for all types of analysis. “Secondary folder” is required only if you wish to compare heavy hitters for two categories of texts (see section 4).

### 3. Heavy Hitters Distribution Analysis

**Heavy Hitters**

Using the ‘Input’ section of the main tab, enter the desired number of heavy hitters and specify the location of data in ‘Primary folder’ field. Once these are done, pressing the ‘Run Analysis’ button located on the far right will begin the execution of Count Sketch algorithm on the provided data. Eventually, the output will be displayed as shown in figure 2. The ‘Output’ section presents the results in both tabular form and bar chart form. By default, both forms display the result in descending order with respect to frequency values. Bar charts can be zoomed in/out or dragged using the mouse for checking different ranges.

![Figure 2: Sample output](image)

29
Distribution Analysis

*Click here* to analyze the distribution (across individual files) of selected items in the table below.

<table>
<thead>
<tr>
<th>Item</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>breast</td>
<td>112</td>
</tr>
<tr>
<td>right</td>
<td>79</td>
</tr>
<tr>
<td>mass</td>
<td>46</td>
</tr>
<tr>
<td>left</td>
<td>44</td>
</tr>
<tr>
<td>right breast</td>
<td>44</td>
</tr>
<tr>
<td>patient</td>
<td>36</td>
</tr>
<tr>
<td>cm</td>
<td>32</td>
</tr>
<tr>
<td>left breast</td>
<td>31</td>
</tr>
<tr>
<td>enhancing</td>
<td>25</td>
</tr>
<tr>
<td>bi-rads</td>
<td>24</td>
</tr>
<tr>
<td>breast bi-rads</td>
<td>22</td>
</tr>
<tr>
<td>suspicious</td>
<td>22</td>
</tr>
<tr>
<td>outer</td>
<td>22</td>
</tr>
<tr>
<td>lymph</td>
<td>21</td>
</tr>
<tr>
<td>axillary</td>
<td>20</td>
</tr>
<tr>
<td>history</td>
<td>19</td>
</tr>
<tr>
<td>findings</td>
<td>19</td>
</tr>
<tr>
<td>lesions</td>
<td>18</td>
</tr>
<tr>
<td>upper</td>
<td>18</td>
</tr>
</tbody>
</table>

Figure 3: Selection of items for further analysis

Using the table of results displayed in the ‘Output’ section, select the items of interest and choose to run further analysis on them via the ‘Click here’ link located directly above the table, as shown in figure 3. Note that all items can be selected at once by using the ‘Check All’ option in the right-click menu.

The analysis involves computing the exact frequency of each selected item in a particular file. Based on such computation, the Shannon entropy for each term is computed as well. Entropy provides a quantitative measure of how evenly distributed a term is with respect to different files.

Upon running the analysis, the tool will switch to a new tab (i.e. ‘Items Distribution’ tab), as shown in figure 4.
The upper part displays the list of files containing the selected items. For each file, the exact frequency of each item in the file is given. There is also a table that displays the Shannon entropy values sorted in descending order. The higher the value, the more evenly distributed the items are across different files.

The lower part shows a three-dimensional representation of the tabular result (i.e. x-axis: File, y-axis: Item, z-axis: Frequency). It can be dragged or zoomed via mouse interaction. Each bar can be clicked to view their label as shown in figure 5. Moreover, users can double-click on any of the bars or a particular column in the table in order to view where the particular item occur in the file as shown in figure 6. If the user double-clicks on any of the cells in ‘File’ column in the table, occurrences of *all* items will be marked.
Figure 5: Selection of a particular bar to view its labels

Figure 6: Occurrences of selected item in a particular file
Neighboring Words

By right-clicking on any of the result tables shown in figure 3 ("Heavy Hitters" Tab) and in figure 4 ("Items Distribution" Tab), select “Find Neighboring Words…” option from the menu. This option enables users to generate an Excel report which enumerates neighboring words for a given choice of words. Upon selecting the option, a configuration dialog will pop-up, as shown in figure 7.

![Figure 7: Neighbor Configuration](image-url)
Using the dialog, the desired number of neighboring words to the left and right of each word can be specified. If the user enters a non-numeric text (e.g. period mark, question mark, sequence of letters, etc.), then all words, until that particular text is encountered, will be considered as neighbors.

There is also a ‘Find Negations’ column that can be checked for each term. If checked, any phrase that involves a negation of the term will be reported along with the neighboring words. It is based on the NegEx algorithm.

Once the configuration is done, click the ‘Ok’ button and specify where to save the Excel report.

4. Heavy Hitters Comparison Analysis

Heavy Hitters

Using the ‘Input’ section of the main tab, enter the desired number of heavy hitters and specify TWO locations of data – one category of texts in ‘Primary folder’ field and another category of texts in ‘Secondary folder’ field. For example, if there are BI-RADS reports grouped into “Malignant” reports and “Benign” reports, the former could be specified as ‘Primary folder’ and the latter as ‘Secondary folder’. Once these are done, pressing the ‘Run Analysis’ button located on the far right will begin the execution of Count Sketch algorithm on each set of data individually (i.e. two executions of Count Sketch algorithm).

Eventually, the output will be displayed as shown in figure 8. The ‘Output’ section presents the results in both tabular form and bar chart form. By default, both forms display the result in descending order with respect to frequency values. Bar charts can be zoomed in/out or dragged using the mouse for checking different ranges.

The upper half of the ‘Output’ section displays the results for primary data, while the lower half displays those of secondary data. The tabular results are self-explanatory – each represents the Count Sketch results for one particular category of data. Similarly, each bar chart visualizes the results for one particular category of data in blue color. However, each bar chart also contains the results for the other category in green. These are shown in overlapped format for easier visual comparison. In order to account for the difference in size of texts in each category, the frequency values are normalized by the total number of items in a particular category of texts (the results in the table though are NOT normalized).
Figure 8: Sample output displaying comparison of two categories of texts – C04 and C14

Comparison Analysis

Figure 9: Link for performing further comparison analysis
Run further comparison analysis via the ‘Click here’ link located directly above the table, as shown in figure 9. Doing so will automatically switch the tab to ‘Items Comparison’ tab and display the results as shown in figure 10.

---

**Figure 10: Results of further comparison analysis**

The ‘Items Comparison’ tab is largely divided into the following two sections:

- **Feature Selection**

Based on the collection of heavy hitters found in previous step for two categories of texts, this section displays which terms occur in both categories of texts (‘Overlapping Items’ section) and which terms occur only in one particular category of texts (‘Unique Items’ section). Its ultimate purpose is to provide an interface for selecting statistically significant terms (i.e. terms that will help discriminate two categories of texts if used as features for machine learning). To aid this process, it provides the following metrics:

<table>
<thead>
<tr>
<th>Metric</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>P(Item</td>
<td>Primary)*</td>
</tr>
<tr>
<td>P(Item</td>
<td>Secondary)*</td>
</tr>
<tr>
<td>Prob. Difference</td>
<td>The absolute difference of two probabilities described above.</td>
</tr>
<tr>
<td>p-value</td>
<td>The p-value for testing significant “Prob. Difference”</td>
</tr>
<tr>
<td>DF (Primary)</td>
<td>Number of files in primary data that contains the given item</td>
</tr>
</tbody>
</table>
The most important metric is the “p-value” since this provides the basis for choosing statistically significant terms. Hence every term that has p-value lower than 0.05 (i.e. 95% confidence for rejecting the null hypothesis that two probabilities are equal) are displayed in **bold**. These terms can be selected all at once by choosing the “Check Bold Items” option from the right-click menu.

### Feature Generation

Once the statistically significant terms have been chosen in the ‘Feature Selection’ section, it is possible to represent each text as a feature vector consisting of probability values for each term. The feature vectors can be generated automatically using this section as follows:

**Step 1:** Choose ‘Output Type’
- **CSV:** Feature vectors are generated in the format outlined below.

```
{ P(Item#1|Primary), P(Item#2|Primary), …, P(Item#n|Primary) },0
{ P(Item#1|Primary), P(Item#2|Primary), …, P(Item#n|Primary) },0
  …
{ P(Item#1|Secondary), P(Item#2|Secondary), …, P(Item#n|Secondary) },1
{ P(Item#1|Secondary), P(Item#2|Secondary), …, P(Item#n|Secondary) },1
```

Each row represents a feature vector for a particular file. For each row, the first \( n \) values are comma separated values of probabilities and the last column specifies the label. Label of 0 represents primary category and label of 1 represents secondary category. **Choose this output type for general purpose machine learning applications!**
- **ARFF (Weka):** This output type is only used for Weka machine learning tool.

**Step 2:** Choose ‘Target Data’
- **Current (Train):** Generates feature vectors for the primary data and secondary data that has already been provided via the main tab.
- **Test:** Generates feature vectors for data specified in ‘Test Data Location’.

**Step 3:** Click ‘Generate’ button
BI-RADS Data Extraction

By selecting ‘Tools’ → ‘Extract Data’ → ‘BI-RADS’ menu located at the top of the application, certain categories of data can be automatically extracted from supplied BI-RADS reports. A sample output (i.e. Excel report) is shown in figure 11.

File names highlighted in red indicate those reports with BI-RADS scores 4 to 6. The “Left” and “Right” column headers distinguish left breast and right breast, respectively.

Figure 11: Excel report of data extracted from BI-RADS reports