https://doi.org/10.2352/issn.2168-3204.2017.1.0.105

Identifying top performing TF*IDF classifiers using the CNN corpus

A. Marie Vans and Steven J. Simske; HP Labs; Fort Collins, CO/U.S.A.

Abstract

*TF*IDF (term frequency times inverse document frequency) is a common metric used to automatically discover keywords in documents for use in classification and other text processing applications. We are interested in determining whether these measures can help in classifying documents. There are multiple ways to define TF*IDF, but there has been no real attempt to determine the value of these different forms. We explore a large family of 112 TF*IDF measures (corresponding to an a priori estimate of 20 degrees of freedom among these measures) applied to 588 CNN articles belonging in 12 classes such as Business, Sport, and World. We postulate that at least some sets of these measures must be effective for classification. The goal is to use a set of TF*IDF measures that best match the a priori classifications by CNN. We also show that by combining the results of a few well-performing TF*IDF measures can increase classification results.*

Introduction

The idea of using somewhat rare terms to identify a document extends back several decades [1]. One well-known method of identifying these fairly rare terms is the term frequency times the inverse of the document frequency, or TF*IDF. The TF*IDF measurement increases for a given document in proportion to the specific term's occurrence, and inversely proportionate to its occurrence in other documents. Ideally, a term that only occurs in one document is a perfect query or indexing term for that document. However, TF*IDF has at least 112 mathematical expressions, and to date no one has determined which is optimal for a given corpus, or how much the performance of these different incarnations of the TF*IDF varies.

Last year, we presented a word-frequency based classification at the Archiving'16 conference in which we demonstrated how a set of 3,000 CNN articles could be classified using this simple frequency metric. The results of those experiments showed how the same approach might be used with a more complex set of measures which we defined as family of 112 TF*IDF measures. We consider their utility for classification of a ground-truthed set of documents organized into 12 classes. This comprehensive study is used to identify which paired combinations of 8 different TF and 14 different IDF formulations (meaning there are 7 degrees of freedom for TF and 13 degrees of freedom for IDF) are likely to be useful for the archiving task of classification. In so doing, we provide insight into the nature of term rareness for the automation of corpus tagging and usability throughout its life cycle.

In Methods and Materials, we cover the TF*IDF measures and how we generate 112 of them. The CNN data set and the pre-processing of the files are discussed. A step-by-step description for each of the algorithms used for classification purposes is presented. The results are then shown for the best performing measures as well as a measure that demonstrates how many attempts to classify each measure undergoes before identifying the correct class. Finally, we determine the maximum accuracy possible for combinations of the top performing TF*IDF measures.

Methods and Materials

*TF*IDF*

TF*IDF (Term Frequency x Inverse Document Frequency) [2, 3, 4, 5] is commonly used in information retrieval and classification tasks [6, 7, 8, 9, 10, 11, 112]. We have defined a total of 112 TF*IDF equations created by using a combination of 14 inverse document frequency equations for each of 8 term document frequency equations. These were computed for a set of CNN articles, which were divided between 12 classes. Within each class, articles were assigned to two equally-sized groups: one for training and one for testing. The total number of files used for each class depends on the number of files in the class with the smallest number of files assigned to it. In our case, one class had only 98 files total assigned to it. The largest class contained 988 files. In order to make sure all classes contributed evenly to the classification task, we used 49 randomly chosen files for each training and each test set from each class. For this paper, we focus on determining the effectiveness of using the set of TF*IDF measures on all words for classifying the CNN file. Table 1 provides the 8 term frequency (TF) measures while Table 2 provides the 14 inverse document frequency (IDF) measures. To build a measure, we multiply one of the TF measures by one of the IDF equations. For example, the Power-Mean measure would be implemented as shown in Equation 1:

$$w_{i,j}{}^{Power} * N - 1/w_{i,n} \quad (1)$$

An experiment consists of preprocessing each document and creating an input stream for each article. We create a stream of tokens composed of individual words using the sharpNLP [13] C# open source project. The stream is then converted into a bag of words consisting of all non-stop words in each file.

Once the TF*IDF measures are generated for each word in the file, we can create a master list of words for all the files in a given class. We create this master list by summing all the TF*IDF values for each word and normalizing this sum by the number of files in which the word is found. This gives us a single TF*IDF measure for each word found in a class which we can then use for classifying articles from the test set of documents.

During testing, we first determine the TF*IDF measure values for all the words in a document. We then compare them with the normalized values for that word in each of the classes using the dot product of the TF*IDF value for the word in the test file with that of the normalized TF*IDF value for the word in each training class. A high result value indicates that the word may belong to the class. The class that produces the highest dot product values for all the words in the file is then assigned as the class for that document. This procedure is used for all the test files in each class for each of the 112 TF*IDF measures. The performance of each TF*IDF is based on how well it predicts the correct class for each of the test files. The TF*IDF measures with the highest classification accuracy are chosen as the top classifiers. A more detailed description of the algorithms and processes are presented below.

Table 1: TF Equations Used in Experiments

	TF Name	TF Numerator
1	**Power**	$(w_{i,j})^{Power}$
2	**Mean**	$w_{i,j}$
3	**NormLog**	$1+\log_2(w_{i,j}) \Big/ \log_2(k)$
4	**Log**	$1+\log_2(w_{i,j})$
5	**NormLogs**	$1+\log_2(w_{i,j}) \Big/ \log_{\frac{2}{LogRatio}}(k)$ $If\ LogRatio \geq MinLogRatio$ $1+\log_2(w_{i,j}) \Big/ \log_2(k)$ $If\ LogRatio < MinLogRatio$
6	**NormMean**	$w_{i,j} \Big/ k$
7	**NormPower**	$(w_{i,j})^{Power} \Big/ k^{Power}$
8	**NormPowers**	$(w_{i,j})^{WordPower} \Big/ k^{DocPower}$

Table 2: IDF Equations Used in Experiments.

	IDF Name	IDF Denominator
1	**NormLogsOfSums**	$\dfrac{\log_{\frac{2}{LogRatio}}(\sum_{j=1}^{N-1} k_j)}{1+\log_2(w_{i,n})}$ *if LogRatio ≥ MinLogRatio* $\log_2(\sum_{j=1}^{N-1} k_j) \Big/ 1+\log_2(w_{i,n})$ *if LogRatio < MinLogRatio*
2	**NormSumsOfLogs**	$\dfrac{\log_{\frac{2}{LogRatio}}(\sum_{j=1}^{N-1}(k_j))}{(\sum_{n=1}^{N-1}(1+\log_2(w_{i,n})))}$ $If\ LogRatio \geq MinLogRatio$ $\dfrac{\log_2(\sum_{j=1}^{N-1}(k_j))}{(\sum_{n=1}^{N-1}(1+\log_2(w_{i,n})))}$ $If\ LogRatio < MinLogRatio$
3	**SumOfPowers**	$N-1 \Big/ \sum_{n=1}^{N-1}((w_{i,n})^{Power})$
4	**PowerOfSums**	$N-1 \Big/ (w_{i,n})^{Power}$
5	**Mean**	$N-1 \Big/ w_{i,n}$
6	**NormSumOfLogs**	$\sum_{j=1}^{N-1} k_j \Big/ \sum_{n=1}^{N-1}(1+\log_2(w_{i,n}))$
7	**NormLogOfSums**	$\sum_{j=1}^{N-1} k_j \Big/ 1+\log_2(w_{i,n})$
8	**NormSumOfPowers**	$\sum_{j=1}^{N-1} k_j \Big/ \sum_{n=1}^{N-1}(w_{i,n})^{Power}$
9	**NormSumsOfPowers**	$\dfrac{\sum_{j=1}^{N-1}(k_j)^{DocPower}}{\sum_{n=1}^{N-1}((w_{i,n})^{WordPower})}$
10	**SumOfLogs**	$N-1 \Big/ \sum_{n=1}^{N-1}(1+\log_2(w_{i,n}))$
11	**LogOfSums**	$N-1 \Big/ 1+\log_2(w_{i,n})$

Where:

i =current word
j = current document
k = total words in document *j*
n = total words in other than current document
N = total number of documents in the corpus
wi,j = number of occurrences of word *i* in document *j*.
wi,n = word occurrences of word *i* in other documents.
n_i = number of documents in which *i* occurs.
LogRatio = ratio of log for individual word to log for document length
MinLogRatio = user settable minimum for *LogRatio*
WordPower & *DocPower* = adjustable value, we used ***2*** in our experiments.

CNN Data Set

The CNN data set is a set of news articles, each of which have been classified by CNN into one of 12 classes. The number of files in each class ranges from 98 to 988. We keep the bias of any single class to a minimum by randomly taking 98 files from each class and then evenly assigning them to a test or training set. The total number of unique words in 588 files is 158,423 and these are divided into 80,710 training-set words and 77,713 test-set words. Table 3 shows the number of unique words found in 49 files for each class for both the test and training sets. Note that this data set was originally procured by Rafael Dueire Lins et. al [14] at the Universidade De Pernambuco in Recife, Brazil.

Table 3: CNN Data Set – Words

Class Name	TTL Number of Train Set Unique Words	TTL Number of Test Set Unique Words	Total Number of Words Processed
Business	6149	5747	11896
Health	6280	6001	12281
Justice	5454	5232	10686
Living	7903	7002	14905
Opinion	7902	7754	15656
Politics	5872	5750	11622
Showbiz	4979	6285	11264
Sport	5657	5400	11057
Tech	6626	6176	12802
Travel	11364	9172	20536
US	6596	6594	13190
World	5928	6600	12528

Preprocessing

Each of the 49 files in the 12 classes are preprocessed in order to identify the words that will be used for classification purposes. One of the first operations we do is remove "stop words". Stop words are common words that add very little information for understanding the content of a document. For example, words like "the", "and", and "or" are stop words. All punctuation, with the exception of hyphenation, is also removed. While many Natural Language Processing approaches use lemmatization, we ran some preliminary experiments and determined that lemmatization did not significantly improve classification accuracy but it did increase processing time. Therefore, we decided not to do lemmatization on the files.

We use the C# Open-Source Natural Language Processing (NLP) program, SharpNLP – (https://sharpnlp.codeplex.com/) for parsing the files and retrieving the data used for our classification algorithm. SharpNLP contains many different NLP functions, but we use only the sentence splitter, the tokenizer, and the part-of-speech tagger. The sentence splitter allows us to collect data on sentences, the tokenizer identifies individual words, and the part-of-speech tagger allows us to identify the function of a word in a sentence, information that we keep as part of the word structure.

Algorithms

Part I:

Using Training Set files in each class: (i.e. do this 12 times)

1. For each file in the set:
   1.1. Create a word object for every unique word in the file
   1.2. Calculate the weight of each word using each TF*IDF measure:
      1.2.1. TF*IDF measure calculated for each word, normalized by the number of occurrences of the word in all the files in the training class.
2. Build a Master Word List for each class separately:
   2.1. For each unique word in the training set for the class, use the average weight of each TF*IDF for the word created by summing the weight of word in each file and normalize by the number of files in which the word is found:

$$w_{class_i} = \sum_{j=0}^{nfiles} TF * IDF_{w_{class_i}} \Big/ nfiles_{w_{class_i}} \quad (2)$$

where w_{class_i} is a unique word in the class and $\boldsymbol{nfiles_{w_{class_i}}}$ denotes the number of files in the class in which the word occurs**.**

3. Save each Master Word list for each class in separate file for later use during classification tasks.

Part II:

Using the Testing Set files for a specific class: (i.e. business)

1. For each file in the set:
   1.1. Create a word object for every unique word in the file
   1.2. Calculate the weight of each word using each TF*IDF measure:
      1.2.1. TF*IDF measure calculated for each word, normalized by the number of occurrences of the word in all the files in the test class.
2. Build a Word List for the current set of test documents that needs to be classified:
   2.1. For each unique word in the test set, use the average weight of each TF*IDF for the word created by summing the weight of word in each file and normalize by the number of files in which the word is found:

$$w_{class_i} = \sum_{j=0}^{nfiles} TF * IDF_{w_{class_i}} \Big/ nfiles_{w_{class_i}} \quad (3)$$

where w_{class_i} is a unique word in the class and $\boldsymbol{nfiles_{w_{class_i}}}$ denotes the number of files in the class in which the word occurs.

3. Save the test word lists for later classification tasks.

Part III:

Classify a single document or a set of documents

1. Load all training set words created in Part I, one set for each class.

2. Load the test set words created in Part II.

3. For each of the 112 TF*IDF measure set of results:

   3.1. For the single file or set of files in the test set:

      3.1.1. Find each unique word in the file or set of files in each of the training set class and record its weight.

      3.1.2. Chose the class each word belongs to, based on the maximum class value (see equation 3):

$$e.g.\ Test_{word_i} = word_i \text{ in Business test class}$$

$$Class_{word_i} = Max \left\{ \begin{array}{l} C_{health} = Test_{word_i} \times Health\_Train_{word_i} \\ C_{justice} = Test_{word_i} \times Justice\_Train_{word_i} \\ \vdots \\ C_{world} = Test_{word_i} \times World\_Train_{word_i} \end{array} \right\} \quad (4)$$

      3.1.3 Each C_{Class} becomes a summation of each word calculation in (3) above. Once all words have been processed, the class chosen for the file is the class containing the max value for all C_{class} variables:

$$Class_{file_n} = max \sum_{i=1}^{nwords} C_{Class_i} \quad (5)$$

For example, if the results from one TF*IDF were as shown in table 4, the resulting classification would be "***Business***".

Table 4: Example Result for single TF*IDF Measure

Class Name	TF*IDF Result
Business	0.00069364
Health	0.00030063
Justice	0.00025000
Living	0.00026707
Opinion	0.00033446
Politics	0.00034694
Showbiz	0.00025372
Sport	0.00029984
Tech	0.00033337
Travel	0.00023201
US	0.00031539
World	0.00040208

MAX

Part IV:

Calculate the mean attempts to classify:

1. Determine how many attempts each TF*IDF equation classifies a file until it is classified correctly:

$$(\mathbf{1} \times \boldsymbol{P_1} + \mathbf{2} \times \boldsymbol{P_2} + \mathbf{3} \times \boldsymbol{P_3} \ldots + \mathbf{12} \times \boldsymbol{P_{12}}) \,/\, \boldsymbol{nfiles} \quad (5)$$

Where:

$\boldsymbol{P_1} = \boldsymbol{number\ correctly\ classified\ on\ first\ try}$

$\boldsymbol{P_2} = \boldsymbol{number\ correctly\ classified\ after\ two\ tries}$

$\vdots$

$\boldsymbol{P_{12}} = \boldsymbol{number\ correctly\ classified\ on\ the\ last\ try}$

$\boldsymbol{nfiles} = \boldsymbol{number\ of\ files\ in\ testing\ class}$

Table 5 shows an example taken from a simple frequency metric [15]. This table shows that, based on the value in the results column, this metric would classify the file as "Politics". The next highest value will classify it as a "World" file. Finally, the 3rd highest value is the correct "Opinion" class. In this case

Equation (6) shows an example of how *attempts to classify* is calculated for the results of the frequency metric. For this experiment, there were 297 total files in the test set and the metric misclassified as many as 8 times. Based on the equation in (6), the average number of times it took for the frequency metric to correctly classify an "Opinion" file was 3.09.

Table 5: Example Result for single TF*IDF Measure – 3 attempts to classify a file from the *Opinion* class

Class	Results
Business	0.00033924
Health	0.00025056
Justice	0.00027728
Living	0.00027807
Opinion	0.00041936
Politics	0.00046704
Showbiz	0.00023136
Sport	0.00028422
Tech	0.00025991
Travel	0.00021793
US	0.00032973
World	0.00043251

It takes 3 tries to get it right

3rd Max

Max

2nd Max

Example: Worst Class – Opinion

$$\sum \begin{matrix} P_1 = 3 \\ P_2 = 35 \text{ x } 2 \\ P_3 = 31 \text{ x } 3 \\ P_4 = 14 \text{ x } 4 \\ P_5 = 7 \text{ x} 5 \\ P_6 = 3 \text{ x} 6 \\ P_7 = 2 \text{ x } 7 \\ P_8 = 1 \text{ x} 8 \end{matrix} = 297/96 = 3.09 \quad (6)$$

Part V:

Create the confusion matrices to determine the accuracy of each of the 112 TF*IDF measures:

1. Each TF*IDF measure classifies a file or document set as one of 12 classes (from *business* to *world*). The confusion matrix is constructed by creating a two-dimensional matrix wherein both the columns and rows are representations of the classes, as seen in table 4 below. The rows represent the "true" class, while the columns represent the class chosen by the classification algorithm described above.

Table 5: Example Confusion Matrix for 3 classes

Normalized Confusion Matrix		Classifier Output (Computed Classification) Prediction		
		A	B	C
True Class of Samples (Input)	A	0.94	0.03	0.03
	B	0.08	0.85	0.07
	C	0.08	0.04	0.88

1.1. Table 6 shows an actual confusion matrix for a single TF*IDF. The highlighted diagonal in the table shows how well the example TF*IDF measure performs on all 12 classes. The cells contain the number of files in the test set classified into the classes named by the columns.

1.2. In the case of the reported experiment there are a total of 49 files in each class. In the confusion matrices for the 112 TF*IDF measures the rows sum to 49 since the left column represents the actual class from which the document is taken. The columns have a mean of 49 with some variance depending on whether the class in the column is an ***attractor class*** (> 49) or a ***repulsor class*** (< 49).

2. Determine the accuracy of the TF*IDF measure divide the sum of values in the diagonal by the total number of files in all test files for all classes:

$$Accuracy = \sum(a_{1,1} \dots a_{n,n}) \Big/ \sum_{i=1}^{nClasses} nfiles \quad (7)$$

3. Sort the TF*IDF measures by accuracy from highest to lowest accuracy and set a minimum accuracy to identify a subset of the 112 TF*IDF measures that best classify the files.

Part VI:

1. Once the TF*IDF measures are sorted by accuracy and the subset of measures that meet the minimum requirement are identified, this subset can be used by adding the accuracies of all possible combinations of the measures in the subset. This allows us to determine whether we can get better accuracy than any one single equation and to determine if there is a maximum accuracy that can be obtained by combining the results using combination:

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} \quad (8)$$

Table 6 shows, in general, how values are calculated. Each combination number represents a single TF*IDF measure. For example, ***"1+2"*** in the "***Combinations***" column represent the sum of the accuracies for TF*IDF equations 1 and 2, where

Equation 1 = TF_Log*IDF_LogOfSums and

Equation 2 = TF_Log*IDF_Mean.

Table 7: Example Combinations of TF*IDF Measures

TF*IDF Combinations	Combinations
2	**From: 1+2, 1+3, 1+4…** **To: *n-1, n***
3	**From: 1+2+3, 1+3+4…** **To: *n-2 ,n-1, n***
⋮	⋮
k	**From: 1+2+3…+k,** **To: *n-k,…,n-1, n***

Results

*Identifying TF*IDF Measures for Classification*

While we ran the experiment using all 112 TF*IDF measures, table 8 shows the accuracy for twelve measures only. The minimum cut off used for performance was 0.34, or an accuracy of 34% in the confusion matrix, and only twelve measures were above that minimum. Additional minimums were experimented with, but it was found that even a small drop to 30% identified 15 more equations, bringing the number up to 27 equations. As will be seen in the section on combining results, using all possible combinations of 27 equations greatly increases the processing time without adding any accuracy.

Recall that the ground truth for these experiments is the classification provided by tags originally assigned by CNN. One conclusion that might be drawn from table 8 is that none of the TF*IDF measures classify all the classes with high accuracy. However, the confusion matrices tell us that they perform much better on some classes than others. For example, table 6 shows that TF*IDF 1 (TF_Log_IDF_LogOfSums) actually classifies documents into specific classes much better than the maximum 45% shown in table 8. In fact, for the "Sports" category, TF*IDF 1 (TF_Log_IDF_LogOfSums) classifies a real sports document 65% correctly. If most of the 112 measures correctly classify most of the documents in a given class, but incorrectly classify most of the documents in a different class, it may be that the class itself was not classified well to begin with.

Another thing to notice about these results is the actual measures themselves. Table 9 includes the names, from which we are able to identify that only half of the defined TFs show up in the top 12. TF_Log makes 7 out 12 of the TF half of the best performing equations. TF_NormLog, TF_NormLogs, occur only twice, while TF_Mean appears only once. There is greater variability for the IDF portion of the winning metrics. IDF_NormSumsOfPowers shows up 4 times, IDF_NormPowersOfSums occurs 3 times; and IDF_NormLogOfSums, IDF_NormMean, IDF_NormPowerOfSums, IDF_Mean, and IDF_LogOfSum all occur only once.

Table 9: Top 12 TF*IDF Measures for Classification

TF*IDF Equation	TF*IDF NUM	Accuracy
TF_Log_IDF_NormLogOfSums	2	0.4473
TF_Log_IDF_NormMean	4	0.4388
TF_NormLogs_IDF_NormSumsOfPowers	52	0.4099
TF_NormLog_IDF_NormSumsOfPowers	38	0.4082
TF_Log_IDF_NormSumsOfPowers	10	0.4048
TF_Log_IDF_NormPowerOfSums	5	0.3639
TF_Log_IDF_NormPowersOfSums	6	0.3622
TF_Mean_IDF_NormSumsOfPowers	24	0.3622
TF_NormLogs_IDF_NormPowersOfSums	48	0.3622
TF_NormLog_IDF_NormPowersOfSums	34	0.3605
TF_Log_IDF_Mean	1	0.3588
TF_Log_IDF_LogOfSum	0	0.3401

Mean Attempts to Classify

Another way to measure how effective a particular TF*IDF measure is in classifying a document is to look at how many incorrect classifications occur before the correct class is chosen. As described above, the "mean attempts to classify" counts how incorrect classes are chosen by the metric before the correct class is chosen. For 112 TF*IDF, we kept track of the classification of each file. Table 10 contains a snippet from the entire table of 112 results showing 21 measures. We used a cut-off mean attempt value of 2.5 attempts. Table 11 shows the names of the measures for comparison to table 9 above.

One thing to notice when comparing tables 9 and 11 is that not every equation that appears in the *Accuracy* (table 9) appears near the top of the *Mean Attempts to Classify* (table 11) table. In fact, only 7 of the twelve in table 9 appear in table 11. The measure with the highest accuracy does not even appear in table 11. The reason for this can be seen by looking at table 10. For example, if we look at TF*IDF measure 2, which has the highest accuracy (TF_Log_IDF_NormLogOfSums), table 10 shows that whereas it finds the correct class on the first attempt 8 out of 12 times, those instances where it did not find the correct class took as long as 5 or even 10 attempts before finding it. If we look at the other measures such as 5 or 6 in table 10, we see that the highest number of attempts to classify is 3 and that occurred for a single class.

Table 11: Mean Attempts to Classify – Cut off value < 2.0

TF*IDF Equation	TF * IDF Num	Mean Attempts to Classify
TF_Log_IDF_NormPowerOfSums	5	**1.25**
TF_Log_IDF_NormPowersOfSums	6	**1.25**
TF_Mean_IDF_NormPowersOfSums	20	**1.25**
TF_NormLogs_IDF_NormPowersOfSums	48	**1.33**
TF_Mean_IDF_NormSumsOfPowers	24	**1.67**
TF_NormLog_IDF_NormPowersOfSums	34	**1.67**
TF_Power_IDF_NormSumsOfPowers	108	**1.67**
TF_Log_IDF_NormMean	4	**1.75**
TF_Power_IDF_NormLogOfSums	100	**1.75**
TF_Log_IDF_NormSumsOfPowers	10	**1.92**
TF_Power_IDF_NormMean	102	**1.92**
TF_Power_IDF_NormPowersOfSums	104	**1.92**

*Combinations of the top 12 Accuracy TF*IDF Measures*

The final aspect of this series of experiments looks at ways to increase accuracy using these measures. While there much more sophisticated ways of combining these metrics using meta-algorithms [15], we focus here on a simple combination using the sum of the accuracies. In the interest of brevity, we focus here only on the twelve reported in table 9.

Table 12 contains information on how many sets of combinations were looked at and the total number of combinations each set contains. As can be seen from the table, up to 924 combinations using 6 TF*IDF metrics can be generated and the total number of unique combinations is 4, 082 just from the twelve metrics shown.

Figure 1 shows the results for accuracy when the combinations are applied for each class. There are a few very interesting things to be seen just from looking at the graph. First, there is a large gap between three of the classes and the remaining 9. In general, the classes "Opinion", "Travel", and "Living" have been consistently shown to be difficult to classify correctly by all of the measures. Even when combining the highest accuracy metrics the best accuracy we can achieve is about 30.6%. For the rest of the classes, combination accuracy is much higher with the maximum value at 88%, with an average for the upper 9 of 77%.

Another interesting result that can be seen from figure 1 is that even a combination of twelve metrics is overkill. With the exception of the "Tech" and "Showbiz" classes, which don't benefit at all beyond a combination of two measures, all the remaining combination accuracies plateau by time 4 accuracy results are combined.

Table 12: Combinations of TF*IDF Measures

TF*IDF Combin-ations	Total Number of Combin-ations	Combinations
2	66	From: 1+2 To: 11+12
3	220	From: 1+2+3 To: 10+11+12
4	495	From: 1+2+3+ 4 To: 9+10+11+12
5	792	From: 1+2+3+ 4+5 To: 8+9+10+11+12
6	924	From: 1+2+3+ 4+5+6 To: 7+8+9+10+11+12
7	792	From: 1+2+3+ 4+5+6+7 To: 6+7+8+9+10+11+12
8	495	From: 1+2+3+ 4+5+6+7+8 To: 5+6+7+8+9+10+11+12
9	220	From: 1+2+3+ 4+5+6+7+8+9 To: 4+5+6+7+8+9+10+11+12
10	66	From: 1+2+3+ 4+5+6+7+8+9+10 To: 3+4+5+6+7+8+9+10+11+12
11	12	From: 1+2+3+ 4+5+6+7+8+9+10+11 To: 2+3+4+5+6+7+8+9+10+11+12

Discussion and Conclusions

Our goal is determine whether a single metric or a combined set of metrics defined by a series of TF*IDF equations might work well as classifiers for news articles, such as that found in the CNN corpus. We described the algorithms used and the results of our experiments show promise. With a copy of the CNN corpus, it should be straightforward to reproduce the results described herein.

We have described how we used 112 TF*IDF equations to classify 588 CNN articles into a set of 12 classes. Along the way, we found that while no single TF*IDF could accurately classify all the documents, by combining the individual accuracies of several metrics we could at least attain a 63% accuracy and higher for 9 of the twelve classes. For the other three classes, the low performance of all 112 TF*IDF indicates a problem with the original classification (by CNN) in general. It could be that "Living", "Opinion", and "Travel" contain words that are too generic for pinpointing the appropriate class. As a result, we believe this approach can also be used to identify improper sets of class definitions. We also discovered that in reality, only 4 combinations of the top 12 most accurate TF*IDF classifiers are needed to maximize the accuracy of the classification.

While this research focuses on the twelve classes found in the CNN Corpus, we wish to re-run the experiments using additional corpora, such as the New York Times Annotated Corpus [16] which contains articles written and published by the New York Times between January 1, 1987 and June 19, 2007. It contains metadata provided by the New York Times Newsroom, the New York Times Indexing Service and the online production staff at nytimes.com. Specifically it has over 1.8 million articles, over 650,000 article summaries written by library scientists, over 1,500,000 articles manually tagged by library scientists with tags drawn from a normalized indexing vocabulary of people, organizations, locations and topic descriptors, and over 275,000 algorithmically-tagged articles that have been hand verified by the online production staff at nytimes.com This comprehensive data set should provide more accurately ground-truthed information and is actually quite promising for taking this research forward.

We also plan to apply a set of meta-algorithms [15] using both the raw TF*IDF scores as well as the accuracy results to determine whether we can create a stable set of classifiers for any classification of documents, including the CNN Corpus as well as the New York Times Annotated Corpus.

References

[1] Levinson, S.1983. Pragmatics. Cambridge University Press, New York, NY.

[2] Gerard Salton and Christopher Buckley, Term-Weighting Approaches in Automatic Text Retrieval, Information Processing and Management 24.5 (1988): 513-23.

[3] Stephen Robertson, Understanding inverse document frequency: on theoretical arguments for IDF, Journal of documentation 60.5 (2004): 503-520.

[4] C. Manning and H. Schütze, Foundations of Statistical Natural Language Processing. MIT Press, Cambridge, MA, (1999).

[5] Kishore Papineni, Why inverse document frequency?, Proceedings of the second meeting of the North American Chapter of the Association

for Computational Linguistics on Language technologies, Association for Computational Linguistics, (2001).

[6] K.L Kwok, Experiments with a component theory of probabilistic informational retrieval based on single terms as document components. *ACM Transactions on Information Systems, 8(4)*. (1990), Pp. 363-386.

[7] J. Ramos, Using tf-idf to determine word relevance in document queries, Proceedings of the first instructional conference on machine learning (2003).

[8] S. Karbasi, and M Boughanem, Effective level of term frequency impact on large-scale retrieval performance: by top-term ranking method, Proceedings of the 1st international conference on Scalable information systems, ACM (2006), pp, 37.

[9] Ari A. Hakin, Alva Erwin, Kho Eng, Maulahikmah Galinium, and Wuliady Muliady, Automated Document Classification for News Article in Bahasa Indonesia based on Term Frequency Inverse Document Frequency (TF-IDF) Approach, Proceedings of the 6th International Conference on Information Technology and Electrical Engineering (ICITEE), Yogyakarta, Indonesia, IEEE, (2014), pp 1-4.

[10] Giacomo Domeniconi, Gianluca Moro, Roberto Pasolini, and Claudio Sartori, A Comparison of Term Weighting Schemes for Text Classificaiton and Sentiment Analysis with a Supervised Variant of tf.idf, International Conference on Data Management Technologies and Applications, Springer International Publishing, (2015), pp. 39-58.

[11] Youngjoong Ko, A Study of Term Weighting Schemes Using Class Information for Text Classification, Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval, ACM, (2012), pp. 1029-1030.

[12] Bruno Trstenjak, Sasa Mikac, and Dzenana Donko, KNN with TF-IDF Based Framework for Text Categorization, Procedia Engineering 69 (2014): 1356-1364.

[13] CodePlex. 2013. *SharpNLP – open source natural language processing tools*. Retrieved from https://sharpnlp.codeplex.com/#.

[14] Lins, R.D., Simske, S.J., Cabral, L., Silva, G., Lima, R., Mello, R.F. and Favaro, L. (2012, July). A multi-tool scheme for summarizing textual documents. In *Proceedings of 11st IADIS International Conference WWW/INTERNET 2012.* pp. 1–8.

[15] Marie Vans and Steven Simske. Summarization and Classification of CNN. com Articles using the TF* IDF Family of Metrics, Archiving Conference, vol. 2016, no. 1, Society for Imaging Science and Technology, (2016), pp. 21-23.

[16] Steven J. Simske, Meta-algorithmics: patterns for robust, low cost, high quality systems. John Wiley & Sons, 2013.

[17] New York Times Annotated Corpus, Linguistic Data Consortium (LDC), https://catalog.ldc.upenn.edu/LDC2008T19, Accessed September 13, 2016.

Author Biography

Marie Vans is currently a Research Scientist with Hewlett-Packard Labs in Fort Collins, Colorado. Her main interests are security printing and imaging for document workflows, statistical language processing, and other approaches to document understanding. She holds a Ph.D. in Computer Science from Colorado State University. She also recently completed a second master's degree in Library and Information Science at San José State University.

Table 8: Confusion Matrix for 1 TF*IDF Measure: TF_Log_IDF_LogOfSums. Numbers represent number of files identified in each class. Highlighted diagonal represents the number of correctly classified files for each class by this measure.

	Business	Health	Justice	Living	Opinion	Politics	Showbiz	Sport	Tech	Travel	US	World	TTL
Business	**18**	2	2	0	1	3	9	3	4	0	3	4	**49**
Health	1	**19**	4	0	0	4	8	1	1	0	5	6	**49**
Justice	1	1	**26**	0	0	5	6	4	1	0	4	1	**49**
Living	2	3	8	**1**	2	4	17	3	2	0	4	3	**49**
Opinion	7	4	4	0	**7**	9	4	4	0	0	3	7	**49**
Politics	2	3	9	0	1	**19**	4	2	2	0	4	3	**49**
Showbiz	2	2	3	1	0	6	**27**	5	0	0	2	1	**49**
Sport	0	1	5	0	0	1	9	**32**	0	0	0	1	**49**
Tech	5	5	4	2	3	1	6	4	**16**	0	1	2	**49**
Travel	5	5	3	1	2	4	8	6	1	**3**	5	6	**49**
U.S.	0	1	9	0	0	9	10	2	4	0	**11**	3	**49**
World	3	1	5	0	1	5	3	5	2	0	3	**21**	**49**
TTL	**46**	**47**	**82**	**5**	**17**	**70**	**111**	**71**	**33**	**3**	**45**	**58**	**588**

Table 10: Mean Attempts to Classify – Cut-Off Value = 2.5 attempts

TF/IDF Num	Business	Health	Justice	Living	Opinion	Politics	Showbiz	Sport	Tech	Travel	U.S.	World	TTL	Mean Attempts to Classify
5	1	1	1	1	2	1	1	1	1	3	1	1	15	**1.25**
6	1	1	1	1	2	1	1	1	1	3	1	1	15	**1.25**
20	1	1	1	1	3	1	1	1	1	1	2	1	15	**1.25**
48	1	1	1	1	2	1	1	1	1	4	1	1	16	**1.33**
24	1	1	1	2	3	1	1	1	1	5	2	1	20	**1.67**
34	1	1	1	1	3	1	1	1	1	7	1	1	20	**1.67**
108	1	2	1	5	1	1	1	1	1	2	2	2	20	**1.67**
4	1	1	1	2	4	1	1	1	1	6	1	1	21	**1.75**
100	1	2	1	4	2	2	1	1	1	2	2	2	21	**1.75**
10	2	1	1	1	4	1	1	1	1	8	1	1	23	**1.92**
102	1	2	1	5	2	1	1	1	1	3	2	3	23	**1.92**
104	1	2	2	6	2	1	1	1	1	2	2	2	23	**1.92**
38	1	1	1	1	4	1	1	1	1	10	1	1	24	**2**
52	1	1	1	1	4	1	1	1	1	10	1	1	24	**2**
105	1	2	1	5	2	2	1	1	1	3	2	3	24	**2**
103	1	2	2	7	3	1	1	1	1	3	2	2	26	**2.17**
1	1	1	1	5	4	1	1	1	1	10	1	1	28	**2.33**
106	1	2	1	8	3	1	1	1	1	5	2	2	28	**2.33**
2	1	1	1	5	5	1	1	1	1	10	2	1	30	**2.5**
99	1	1	1	2	3	1	1	1	1	11	6	1	30	**2.5**
100	1	1	1	2	3	2	1	1	1	11	5	1	30	**2.5**

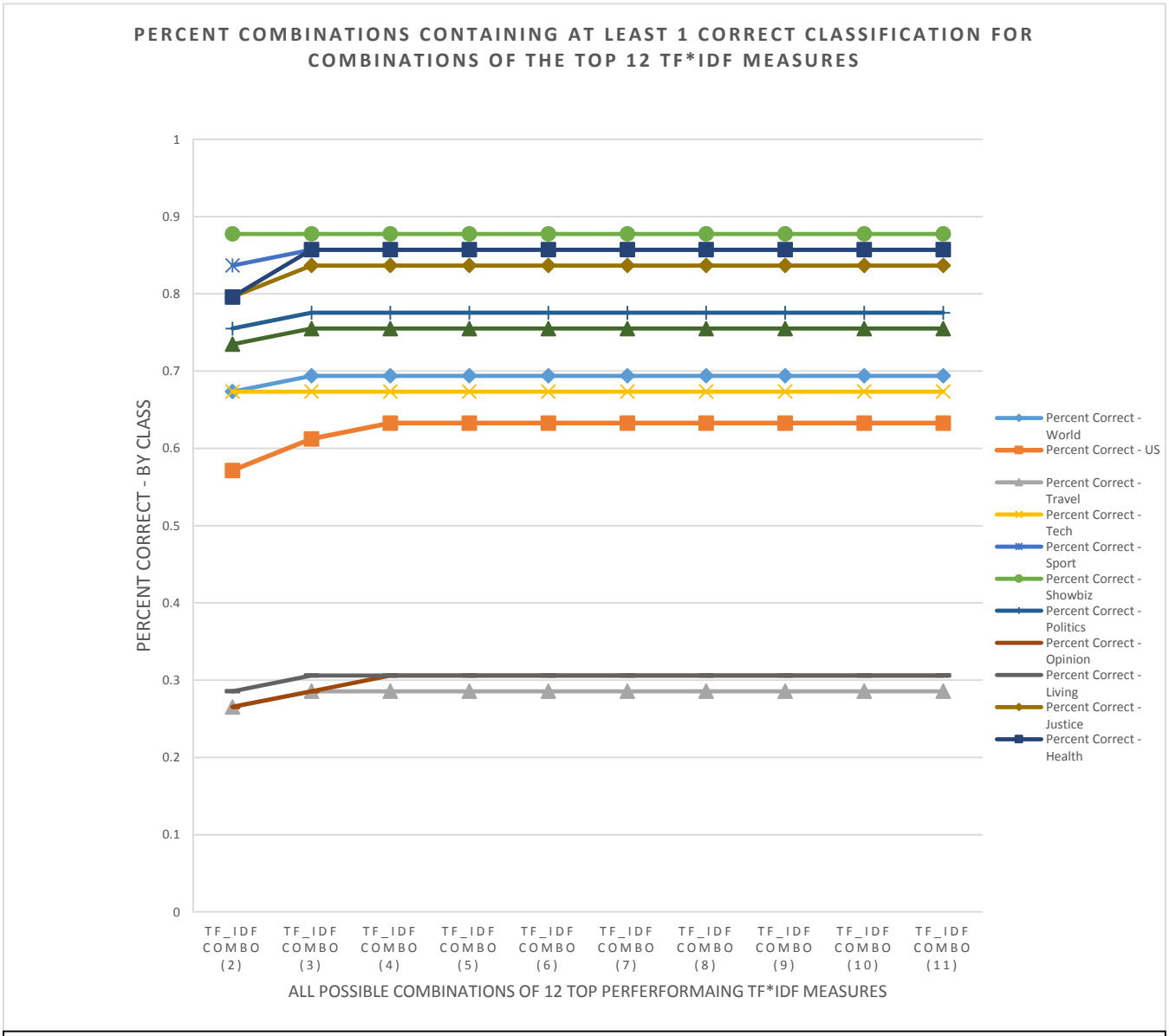

***Figure 1**. Percent Correct Classifications for combinations of the top 12 TF*IDF Measures*