Introducing an Object Oriented Design to the Ngram Statistics Package

A PROJECT REPORT

SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL

OF THE UNIVERSITY OF MINNESOTA

by

Saiyam Kohli

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

July 2006

UNIVERSITY OF MINNESOTA

This is to certify that I have examined this copy of master's project report by

Saiyam Kohli

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Dr. Ted Pedersen

Name of Faculty Adviser

Signature of Faculty Advisor

Date

GRADUATE SCHOOL

Acknowledgments

I would like to take this opportunity to acknowledge the people who have made this possible.

To, Dr. Ted Pedersen for his invaluable guidance and support, I am grateful for all the time and energy you have spent in making this project possible. To Dr. Doug Dunham and Dr. Ron Regal, I appreciate your thoughtful comments and guidance.

To the Department of Computer Science, specially Dr. Carollyn Crouch, Lori Lucia and Linda Kay Meek for their support.

I am also indebted to my parents, brother and friends for their endless encouragement and support. Without them nothing would have been possible.

Abstract

The Ngram Statistics Package is a freely available tool that counts sequences of words in text, and identifies those that have some added significance via a number of different measures of association. This package has been available since 2001, and has literally hundreds of users.

However, the internal organization of the package has shown signs of age and disrepair. This project was undertaken to make NSP more object oriented, in order to reduce code replication and to improve the maintainability of the package. A significant reorganization was undertaken that has greatly reduced the amount of duplication in the code, and improved the logical organization of the measures in the package. In addition, several new measures have been added to the package, and a significant optimization to Fisher's Exact test has been designed and implemented.

Table of Contents

Index of Tables

Index of Figures

1 Introduction

Ngram Statistics Package (NSP) is a collection of Perl modules and programs that aid in identifying Ngrams in text corpora. Identifying interesting Ngrams is one of the fundamental problems in computational linguistics. Ngrams are defined as a sequence of *N* tokens that occur in proximity to each other. Tokens are the smallest indivisible units of text. In NSP tokens can be alphanumeric characters, words or some combination of these.

Ngrams were first used by Claude E. Shannon [16] to compute the entropy of the English language. He devised an experiment wherein he used the previous *N* characters in a phrase to guess the $N+1^{st}$ character. Nowadays, Ngrams are used in several areas of statistical computational linguistics. For example, they are used in pattern recognition systems to compute the probability of a given word sequence appearing in text.

In speech recognition phonemes are modeled using Ngrams. Similarly Optical Character Recognition (OCR) systems use Ngrams to identify the next character or word. This is achieved by identifying and computing the probability of all the possible Ngrams. Then the most probable Ngram containing the already identified tokens (phonemes, character or words) is used to identify the next token.

Ngrams also find use in word sense disambiguation, machine translation and information retrieval. Most supervised word sense disambiguation applications use Ngrams features to identify the meaning of a given target word. In machine translation Ngrams are used to identify collocations.

Collocations are defined as sequences of words in which, if the constituent words are separated they would have a different meaning than the sequence itself. This is important because if such a sequence of words were translated verbatim the resulting sequence might not have the same meaning. Some examples of collocations are “World Cup”, “middle management” etc.

Some of the contributions of this project are the introduction of new measures for Ngram Statistics Package. In addition, the existing measures of association have been rewritten using a more modular approach, using object oriented concepts. The (*) in the Table of Contents of this report is used to identify the new measures of association or the ones to which some significant changes have been made.

The first chapter of this report provides a brief overview of the Ngram Statistics Package. In the second chapter we go on to provide the theoretical background of all the measures of association available in Ngram Statistics Package. Third chapter discusses the new design of measures of association in NSP. Then we go on to describe some of the Experiments performed and the results obtained. Finally we discuss some of the future work that can be done.

2 Overview of Ngrams Statistics Package

There are two programs in NSP that allow users to identify and analyze Ngrams in a text corpus. The program *count.pl* is responsible for identifying and counting the number of occurrences of Ngrams. *count.pl* takes a text file as input and outputs a list of all the Ngrams along with their frequency counts. Program *statistic.pl* allows the user apply a measure of association to the output of *count.pl.* These measures of association can be used to decide whether a given Ngram is significant or not.

This chapter provides a detailed introduction to the Ngram Statistics Package. The first few sections describe the process of identifying and counting Ngrams from a text file. Next we discuss the implementation and working of the program *statistic.pl.* Finally the program *rank.pl,* which can be used to compare two measures of association, is described

2.1 Tokenization

Generally tokenization is the first step in any natural language processing task. During tokenization the input text is broken down into the smallest indivisible units, known as tokens. NSP uses Perl regular expressions to define tokens in text. These regular expressions can either be specified by the user, or the defaults are used (Figure 1).

```
/\w+/
[\.,;:\?!]
```

Figure 1: Default regular expressions

The first regular expression defines a token to be a sequence of alpha-numeric characters, while the second regular expression defines a single punctuation mark to be a token.

These defaults more or less correspond to the idea of a token as a word, but in fact tokens can either be alphanumeric characters or any such combination.

Given a set of regular expressions, the input file is converted into one long input string by replacing all the new-line characters with a space character. This string is then matched against each of the regular expressions. If a substring, beginning from the first character of the input string, matches any regular expression it is identified as a token and is deleted from the input string. In case no match is found the first character in the input string is marked as a nontoken and is deleted.

This process is repeated until the input string is empty. For example, consider the input string "Heavy security was in place as U.S. President George W Bush landed in Delhi.". Table 1 lists all the tokens that would be identified if the default regular expressions are used.

Table 1: Tokens with default regular expressions

Heavy<>	security<>	was<>	in<>	place<>
as<>	U<>	.<>	S<>	.<>
President<>	George<>	W<>	Bush<>	landed<>
in<>	Delhi<>	.<>		

The meta-character <> is used to terminate a token. Such a representation is required because a token might be defined to consist of embedded white spaces. For example "President George W Bush<>".

Users can provide regular expressions by placing them in a single file and then passing the name of this file using the ***--token*** option in *count.pl*. NSP prioritizes the regular expressions in the order they are specified, and different orderings of regular expressions might produce different set of tokens.

For example, assume that the regular expressions **/United States/**, **/United States of America/** and **/\w+/** are used, in that order, to tokenize the input string “Ghana beat the United States of America in Group E decider”. Since the first three substrings (i.e. “Ghana”, “beat” and “the”) match the third regular expression, they are identified as valid tokens and are deleted from the input string. Even though the remaining string matches all three regular expressions, “United States” is identified as the next token.

In case the order of regular expressions is switched to **/United States of America/**, **/United States/** and **/\w+/** the resulting set of tokens would be slightly different, as “United States of America” will be identified as the fourth token instead of “United States”.

Often times it is required that certain sequence of character not be considered as tokens. This can be achieved by using the ***--nontoken*** option with *count.pl*. Users can define regular expressions to match the strings that should not be identified as tokens. The input data is first compared to the set of non-token regular expressions, if a matching string is found it is deleted from the input data. Once all the non-token strings have been removed from the input data, it is parsed to identify tokens. This can be useful when removing numeric data or HTML markup tags from the input text.

2.2 Identifying Ngrams

Once the text has been tokenized, the next step is to identify Ngrams. This is done by assembling sequences of *N* tokens. Generally these tokens are contiguous, that is they occur together in the text. Tables 2 lists all such bigrams in the string “Heavy security was in place as U.S. President George W Bush landed in Delhi.”.

Table 2: Bigrams with default window size

Heavy<>security<>	security<>was<>	was<>in<>	in<>place<>	place<>as<>
as<>U<>	U<>.<>	.<>S<>	S<>.<>	.<>President<>
President<>George<>	George<>W<>	W<>Bush<>	Bush<>landed<>	landed<>in<>
in<>Delhi<>	Delhi<>.<>			

NSP can also be used to identify Ngrams of non-contiguous tokens. This is done by allowing the user to define a window of k contiguous tokens, using the ***--window*** option in *count.pl*. The value of *k* should be greater than or equal to the value of *N* for the Ngrams. Now, Ngrams can be formed from any *N* tokens as long as all these tokens belong to a single window. Also, all the *N* tokens should occur in the Ngram in exactly the same order as they occur in the window. Table 3 shows all the bigrams in the "Heavy security was in place as U.S. President George W. Bush landed in Delhi." for a window of 3. In this case each bigram allows for up to one intervening token.

Table 3: Bigrams with window size 3

Heavy<>security<>	High<>was<>	security<>was<>	security<>in<>	was<>in<>
was<>place<>	in<>place<>	in<>as<>	place<>as<>	place<>U<>
as<>U<>	as<>.<>	U<>.<>	U<>S<>	.<>S<>
.<>.<>	S<>.<>	S<>President<>	.<>President<>	.<>George
President<>George<>	President<>W<>	George<>W<>	George<>Bush<>	W<>Bush<>
W<>landed<>	Bush<>landed<>	Bush<>in<>	in<>landed<>	in<>Delhi<>
landed<>Delhi<>	landed<>.<>	Delhi<>.<>		

2.3 Counting Ngrams

After all the Ngrams in a given text have been identified, the next step is to count the number of occurrences of each Ngram in the text. NSP not only counts the number of times a Ngram occurs in the text file, but it also counts all the unique combinations of the *N* constituent tokens, given that each of the tokens occurs in the same position as they occur in the Ngram. For example in the string "Heavy security was in place as U.S. President George W Bush landed in Delhi", for a window of size 3 "George<>Bush<>" is a valid bigram, but "Bush<>George<>" is invalid.

To represent a particular frequency combination NSP defines a function *f()*. This function takes a sequence of integer values as input. In this sequence each integer represents whether a particular token occurs in its position or not. The function *f()* returns the number of Ngrams in which these tokens occur in their respective positions. For example, *f(0,2)* returns the frequency count of the trigrams where only the first and third token occur in their respective positions. *f(0, 1, 2)* returns the count of the trigram in which all three tokens occur in their respective positions. By default all such frequency values for a Ngram are reported. Note that the positions are counted starting at zero. Thus first token is represented by zero and so on.

For example, let's consider that NSP is used to count bigrams in a text corpus. Figure 2 shows the output of *count.pl*. The first line of the output denotes the number of total bigrams in the corpus. In this case this number is 1630855. Please note that the total number of bigrams in a corpus is not same as the number of unique bigrams, that is this number also includes repetitions.

The next line and onwards lists all the unique bigrams along with frequency counts associated with each of these bigrams. For each bigram three numbers are reported. The first of these numbers denotes the number of times that particular bigram occurred in the input text file. For example in our sample output the first bigram is “united<>states<>” and it occurred 3590 times in the input corpus. The next bigram, that is “atlanta<>journal<>”, occurred 2248 times. The second number denotes in how many bigrams, the token "united<>" occurs as the left most token. Similarly the third number is the count of the bigrams in which “states<>” occurs as the second token.

```
1630855
united<>states<>3590 4033 4234
atlanta<>journal<>2248 3634 2469
journal<>constitution<>2235 2438 2300
news<>service<>2110 4290 3727
sept<>11<>1912 2198 2916
world<>cup<>1658 4544 3173
white<>house<>1529 2875 2634
cox<>newspapers<>1465 2834 1652
optional<>trim<>1430 2741 1532
1<>2<>1423 10191 5677
story<>filed<>1319 2003 1684
los<>angeles<>1291 1532 1292
york<>times<>1107 3256 2154
coxnews<>com<>1074 1076 4302
palm<>beach<>1055 1127 1531
cox<>news<>996 2834 4090
president<>bush<>986 2629 2759
high<>school<>899 3718 2736
al<>qaida<>845 1874 908
sickle<>cell<>835 837 1106
```

Figure 2: count.pl output for bigrams

The output of *count.pl* gives a concise representation of a contingency table for the given bigram. Contingency tables can be used to analyze the co-occurrence data. The cell counts n_{11}, n_{12}, n_{21} and n_{22} are called the observed frequencies. They add up to the total number of bigrams, that is n_{pp}. The row totals n_{1p}, n_{2p} and column totals n_{p1}, n_{p2}, are referred to as marginal frequencies, since they are written in the margins of the table and represent the row and column sums.

Table 4 is an example of a partially filled contingency table. Only the values that are available in the output of *count.pl* have been filled. The cell n_{pp} denotes the total number of bigrams in the input corpus, which is the same as the first line in the *count.pl* output.

Table 4: A contingency table

	states<>	***!states<>***	
united<>	$n_{11} = 3590$	n_{12}	$n_{1p} = 4033$
!united	n_{21}	n_{22}	n_{2p}
	$n_{p1} = 4234$	n_{p2}	$n_{pp} = 1630855$

Notice that the values for the rest of the internal cells can be computed by using the existing values. For example we can get the number of times the token "united<>" occurred as the first token of a bigram that does not have the token "states<>" by subtracting the number of times "united<>" and "states<>" occurred together from the number of times "united<>" occurred as the first token.

Similarly using the total bigram count and the marginal totals we can compute the values for the rest of the cells. Table 5 shows the completed contingency table. To build a contingency table for a bigram, say "united<>states<>", first all the bigrams are extracted from the source corpus. They are then classified into the four cells of a contingency table, depending on whether the first token is "united<>" or not and similarly whether the second token is "states<>" or not.

Table 5: Completed contingency table

	states<>	***!states<>***	
united<>	$n_{11} = 3590$	$n_{12} = 443$	$n_{1p} = 4033$
!united	$n_{21} = 644$	$n_{22} = 1626178$	$n_{2p} = 1626822$
	$n_{p1} = 4234$	$n_{p2} = 1626621$	$n_{pp} = 1630855$

The output for trigrams (Figure 3) is also similar to the output for bigrams. As earlier the first line of the output represents the total number of trigrams in the input corpus. Next all the trigrams along with the respective frequency are reported. Only in this case there are seven numbers following each trigram instead of just three for the bigrams. The first number as before represents the number of times that particular trigram occurred in the input corpus. In this case the trigram “atlanta<>journal<>constitution” occurs 2235 times. This value corresponds to *f(0, 1, 2).*

```
717527
atlanta<>journal<>constitution<>2235 2958 2379 2243 2248 2235 2235
cox<>news<>service<>995 1864 3456 2713 996 995 2110
com<>story<>filed<>721 2580 1575 1477 721 721 1302
palm<>beach<>post<>669 1064 1312 832 1028 669 669
_<>atlanta<>_<>640 7002 1303 16299 677 1351 686
sickle<>cell<>disease<>625 767 970 728 767 625 625
optional<>material<>follows<>574 1714 669 779 574 725 574
newhouse<>news<>service<>509 601 3456 2713 509 509 2110
times<>news<>service<>470 912 3456 2713 470 470 2110
404<>526<>5456<>417 669 659 417 657 417 417
begin<>optional<>trim<>416 530 1248 764 417 417 683
1<>1<>2<>389 7234 6289 3673 642 450 1181
world<>trade<>center<>374 2329 789 1464 419 391 384
```

Figure 3: count.pl output for trigrams

The next three numbers denote the number of tokens in which the tokens “atlanta<>”, “journal<>” and “constitution<>” occur as the first, second and third tokens respectively, that is the values for *f(0)*, *f(1)* and *f(2)* respectively. Thus “atlanta<>” occurs as the first token in 2958 trigrams, journal<> occurs as the second token in 2379 trigrams and constitution<> occurs as the third token in 2243 trigrams. The fifth number represents the number of tokens in which the first and the second tokens (i.e. atlanta<> and journal<>) occur in their respective positions. In this case it happens 2248 times. Similarly atlanta<> and constitution<> occur as the first and the third tokens 2235 times, and in 2235 trigrams journal<> and constitution<> occur as the second and third tokens respectively. These values corresponds to *f(0,1)*, *f(0,2)* and *f(1, 2)* respectively.

For a Ngram, the first frequency value that is reported is the number of times the particular Ngram occurs in the text. The next ${}^{N}C_{1}$ (i.e. *N* choose *1*) frequency values are *f(0)*, *f(1)*, ..., *f(n-1)*. These are the frequency counts of those Ngrams in which only one of the N tokens occurs in its position. Similarly, next ${}^{N}C_{2}$ (i.e. *N* choose *2*) frequency counts denote the combinations *f(0, 1)*, *f(0, 2)*, ..., *f(0, n-1), f(1,2), f(1,3),......f(1,n-1),.....f(n-2,n-1)*. They represent all the possible Ngrams in which any two of the *N* tokens occurs in their respective positions. Similarly, all such possible combinations are reported. So for each Ngram we end up having a sequence of ${}^{N}C_{N} + {}^{N}C_{2} + {}^{N}C_{3} + {}^{N}C_{4} +$ .... $+ {}^{N}C_{N-1}$ numbers. Thus for every Ngram 2^{N}-1 frequency values are reported.

Since all these frequency combinations might not be useful to a user, NSP allows the user to specify a list of frequency combinations that must be reported by *count.pl*. The user can create a file with the inputs to the function *f(),* to represent which frequency combinations should be counted. The name of this file is passed to *count.pl* using the ***--set_freq_combo*** option.

2.4 Identifying Significant Ngrams

Association measures are used to interpret the co-occurrence frequency data for Ngrams. Program *statistic.pl* takes as input a list of Ngrams with their frequencies (*count.pl* output) and applies a user-selected statistical measure of association to compute a score for each Ngram.

This score can be used to judge whether the given Ngram is statistically significant or not. If the tokens of a Ngram do not occur together just by chance, we believe they have some added meaning or significance. For example “fine<>art<>” or “United<>Nations<>” have meanings that go beyond the combination of the meaning of their individual tokens.

The following measures of association are available in NSP:

- Measures to test bigram frequency counts:
    1. Dice Coefficient
    2. Fisher's Exact test – left sided
    3. Fisher's Exact test – right sided
    4. Fisher's Exact test – two-tailed
    5. Jaccard Coefficient
    6. Log-likelihood ratio
    7. Mutual Information
    8. Odds Ratio
    9. Pointwise Mutual Information
    10. Phi Coefficient
    11. Pearson's Chi Squared Test
    12. Poisson Stirling Measure
    13. T-score
- Measures of Association for trigram data
    1. Log-likelihood ratio
    2. Mutual Information
    3. Pointwise Mutual Information
    4. Poisson Stirling Measure

Apart from these measures, NSP also provides a basic framework for building new measures of association for Ngrams. These new measures should also be implemented as Perl modules. An implementation of a statistical measure is expected to have at least ones measure, that is **calculateStatistic()**. The program *statistic.pl* calls this method for every Ngram and passes it a hash containing the frequency values for that Ngram. The method **calculateStatistic()** is expected to return a (possibly floating point) value as the value of the statistical measure.

There are four other methods that are not mandatory, but may be implemented.

1. **initializeStatistic()**
2. **getErrorCode()**
3. **getErrorMessage()**
4. **getStatisticName()**

The program *statistic.pl* calls **initializeStatistic()** before calling any other method. This method is used to initialize the statistic. The **getErrorCode()** method is called by *statistic.pl* immediately after every call to method **calculateStatistic()**. This method is used by measures to report errors, if any, in the previous operation. In case of an error this method returns a integer value. Table 6 lists all the possible error codes. If there is no error a null value is returned.

Similarly, the method **getErrorMessage()** is used to return a string describing that describes the error. The fourth method that may be implemented is **getStatisticName()**. This method is expected to return a string containing the name of the statistic being used.

Table 6: Error Codes for Measures of Association.

Error Code	***Meaning***
101	calculateStatistic() not implemented by the measure.
200	one of the required values is missing.
201	one of the observed frequency comes out to be -ve.
202	one of the frequency values(n11) exceeds the total no of bigrams(npp) or a marginal total(n1p, np1).
203	one of the marginal totals(n1p, np1) exceeds the total bigram count(npp).
204	one of the marginal totals is -ve.
211 & 221	one of the expected values is zero.
212	one of the expected values is -ve.

To aid the implementation of the statistical measures, Text::NSP::Measures::2D and Text::NSP::Measures::3D provide three methods, namely:

1. **computeObservedValues()**
2. **computeMarginalTotals()**
3. **computeExpectedValues()**

These methods can be called from within the **calculateStatistic()** method of any bigram or trigram measure to compute the marginal totals, observed and expected frequency counts for each Ngram. These measures also perform basic error checks on these values, and in case of an error they return a null value, otherwise 1 is returned to indicate success. Appendix A provides a sample implementation of a measure.

Once the statistical scores for all the Ngrams have been computed, *statistic.pl* ranks them in the descending order of these scores. By default the scores are reported to a precision of 4 decimal places, but users can adjust this by using --precision option with *count.pl.* In case two Ngrams have the same score they are ranked in the descending order of their observed frequency counts. Also, users can specify a statistical score cutoff for Ngrams, by using the ***--score*** option with *statistic.pl.* Any Ngram that has a statistical score which is less than this cutoff will not be ranked. Similarly, a frequency cutoff may also be defined by using the ***--frequency*** option which will remove any Ngram with a frequency less than this value.

2.5 Comparing Measures of Association

The program *rank.pl* allows a user to compare two measures of association. It uses the Spearman's Rank Correlation Coefficient to determine how different two ranked lists of Ngrams are. It is assumed that the two lists rank the same set of Ngrams. The Spearman's Rank Coefficient uses the ranks of the Ngrams to compute the Correlation between the two lists. The Spearman's Rank Correlation Coefficient is defined as:

$$r = 1 - 6 \frac{\sum_{i=1}^{i=n} D_i^2}{n(n^2 - 1)} \tag{1}$$

Here D is the difference between the ranking of Ngram *i*, n is the sample size, or the number of Ngrams in the list. The value of the Spearman Correlation Coefficient *r* ranges from +1 to -1. A value of 0 indicates no correlation between the lists, where as a value of ± 1 indicates complete correlation. A positive value indicates positive correlation and a negative value indicates negative correlation. For example the Rank Coefficient for the two lists of Ngrams shown in Table 7 is 0.0333 which indicates that there is very little correlation between them.

Table 7: Sample input for rank.pl

1922 entirely<>harmless<>1 9.3234 3 3 3 become<>admirable<>2 8.9084 1 4 1 human<>beings<>3 8.5865 3 5 3 afford<>her<>4 7.4490 1 1 11 actually<>prevent<>5 7.3234 1 2 6 independent<>person<>6 6.9084 2 8 4 contemporary<>life<>7 6.0015 5 6 25 convenient<>life<>8 5.5276 3 5 25 to<>be<>9 3.9311 8 63 16	1922 entirely<>harmless<>1 44.7704 3 3 3 contemporary<>life<>2 39.0979 5 6 25 human<>beings<>3 38.0403 3 5 3 to<>be<>4 34.0031 8 63 16 convenient<>life<>5 19.7402 3 5 25 independent<>person<>6 16.9398 2 8 4 become<>admirable<>7 12.6230 1 4 1 afford<>her<>8 10.4197 1 1 11 actually<>prevent<>9 8.9476 1 2 6

3 Association Measures

Association measures are used to interpret the co-occurrence frequency data for Ngrams. They assign a real valued score to each Ngram, which can be used to judge whether the tokens that make up the Ngram occur together more often than expected by chance. This is useful in identifying collocations and other interesting Ngrams.

3.1 Model of Independence

Most measures of association try to compare a contingency table of observed values with one of expected frequency counts. The expected frequencies for a contingency table are estimated based on a hypothesized model which in this case is the model of independence. The hypothesis for this model is that the tokens in a Ngram happen to co-occur purely by chance (i.e. are independent). For bigrams there is only one such hypothesis, that is the two tokens in the bigram have occurred together only by chance, or in other words the probability of the two tokens occurring together is equal to the product of their individual probabilities:

$$P(token1, token2) = P(token1) * P(token2) \tag{2}$$

Under this hypothesis the expected frequencies for bigrams can easily be computed using the marginal totals and the total Ngram count. Table 8 shows how expected values for a 2 x 2 contingency table are computed.

Table 8: Contingency table with expected values

	token2<>	***!token2<>***	
token1<>	$m_{11} = (n_{p1} * n_{1p})/n_{pp}$	$m_{12} = (n_{p2} * n_{1p})/n_{pp}$	n_{1p}
!token1<>	$m_{21} = (n_{p1} * n_{2p})/n_{pp}$	$m_{22} = (n_{p2} * n_{2p})/n_{pp}$	n_{2p}
	n_{p1}	n_{p2}	n_{pp}

For Ngrams with N>2 there is more than one possible model that can be used. For example for trigrams the hypothesis could be that all the three tokens are completely independent of each other (formulation 3), or it could be that two of the three tokens are dependent on each other but are independent of the third (formulations 4,5,6). All four hypothesis will result in different models. It is possible to use all of these models in NSP.

$$P(token1, token2, token3) = P(token1) * P(token2) * P(token3) \tag{3}$$

$$P(token1, token2, token3) = P(token1) * P(token2, token3) \tag{4}$$

$$P(token1, token2, token3) = P(token1, token2) * P(token3) \tag{5}$$

$$P(token1, token2, token3) = P(token1, token3) * P(token2) \tag{6}$$

3.2 Hypothesis testing

The measures that come under Hypothesis testing are Fisher's Exact tests, Pearson's Chi-squared test, Log-likelihood measure and the t-score. These measures formulate a null hypothesis that all the tokens of a Ngram are independent of each other. Then they test a Ngram to see if there is significant evidence to refute this hypothesis. If enough evidence is not found then the hypothesis is accepted, and the Ngram is said to be statistically insignificant and not interesting.

3.2.1 Fisher's Exact tests (*)

Pedersen[1] suggested to use Fisher's Exact test as an association measure for identifying collocations, arguing that it is more accurate than tests such as Pearson's Chi squared test and Log-likelihood ratio that make asymptotic assumptions. Fisher's Exact tests compute the probabilities of all possible contingency tables, whose internal cell counts (i.e. n_{11}, n_{12}, n_{21} and n_{22}) sum up to the observed marginal totals. These probabilities can then be used to check the significance of the observed contingency table.

Fisher's Exact tests are only computationally feasible for 2 x 2 contingency tables, so they are only available for bigrams in NSP.

In Fisher's Exact tests hypergeometric probabilities for all the possible contingency tables are computed by fixing the marginal totals (i.e. n_{1p}, n_{p1}, n_{2p}, n_{p2} and n_{pp}). The possible contingency tables are then generated by considering possible values for n_{11}. The value of n_{11} determines the values of n_{12}, n_{21} and n_{22}. Once all the values for a possible contingency table have been determined the probability of witnessing a contingency table whose internal cells sum up to the observed marginal totals can be computed using the following equation.

$$P = \frac{1}{n_{11}!n_{12}!n_{21}!n_{22}!} * \frac{n_{1p}!n_{p1}!n_{2p}!n_{p2}!}{n_{pp}!} \quad (7)$$

For example consider the following contingency table.

Table 9: A Contingency table

	token2<>	***!token2<>***	
token1<>	$n_{11} = 4$	$n_{12} = 2$	$n_{p1} = 6$
!token1<>	$n_{21} = 4$	$n_{22} = 10$	$n_{p2} = 14$
	$n_{1p} = 8$	$n_{2p} = 12$	$n_{pp} = 20$

For the given marginal totals, there are seven possible contingency tables, since the possible values for the cell n_{11} are 0, 1, 2, 3, 4, 5 and 6. Given the marginal totals ($n_{1p} = 8$, $n_{p1} = 6$, $n_{2p} = 12$, $n_{p2} = 14$ and $n_{pp} = 20$), table 9 lists the probabilities of finding these observed counts.

Table 10: Probabilities for the possible contingency tables.

Values in the internal cells	***Probability***
$n_{11}= 0, n_{12} = 6, n_{21} = 8, n_{22} = 6$	0.024
$n_{11}= 1, n_{12} = 5, n_{21} = 7, n_{22} = 7$	0.163
$n_{11}= 2, n_{12} = 4, n_{21} = 6, n_{22} = 8$	0.357
$n_{11}= 3, n_{12} = 3, n_{21} = 5, n_{22} = 9$	0.318
$n_{11}= 4, n_{12} = 2, n_{21} = 4, n_{22} = 10$	0.119
$n_{11}= 5, n_{12} = 1, n_{21} = 3, n_{22} = 11$	0.017
$n_{11}= 6, n_{12} = 0, n_{21} = 2, n_{22} = 12$	0.001

There are three Fisher's Exact tests (left sided, right sided and two-tailed), the first two of these are one sided, while the third is a two sided test.

The left sided test is calculated by adding the probabilities of all the possible two by two contingency tables where the count in cell n_{11} is less than or equal to the observed value. A left sided Fisher’s Exact test tells us how likely it is to randomly sample a table where n_{11} is less than observed value. In other words, it tells us how likely it is to sample a contingency table where the two tokens are less dependent than currently observed. A high probability value indicates that the particular bigram is significant, and hence we can dismiss the null hypothesis. In our example this value comes out to 0.981, which suggests that the two tokens occur more frequently than expected by chance.

The right sided Fisher's test is calculated by adding the probabilities of all the possible two by two contingency tables where the cell count n_{11} is greater or equal to the observed value. A right sided Fisher's Exact test tells us how likely it is to randomly sample a table where n_{11} is greater than the observed frequency count, that is it tells us how likely it is to sample an observation where the two words are more dependent than currently observed.

If this probability values is small it indicates that the two tokens are dependent on each other hence they do not occur just by chance. For our example the right fisher value is 0.137. Since this value is very small it indicates that the two tokens occur together more frequently than expected by chance and hence we can dismiss the null hypothesis. Thus this bigram is considered to be interesting.

The two-tailed Fisher's test is calculated by adding the probabilities of all the contingency tables whose probabilities are less than or equal to the probability of the observed table. This test tells us how likely it would be to observe a contingency table which is less probable than the observed table under the null hypothesis. Although the value given by this measure cannot be directly used to test the null hypothesis, it still gives us some information about the pair of tokens. If this value is very high, it indicates that the probability of witnessing the observed table under the null hypothesis is very high, that is the two tokens are independent of each other. Thus a high value might suggest that there is not enough evidence against the null hypothesis.

Implementation of Fisher's Exact tests

A simple implementation of Fisher's Exact tests generates all possible contingency tables for the given marginal totals and computes the probability for each of these tables using the formulation given in equation (7). Since the complexity of computing factorials increases exponentially, a simple implementation of Fisher's Exact tests is computationally intensive specially if the number of possible contingency tables is high.

The new implementation of Fisher's Exact tests in NSP done for this project overcomes this problem by using the probability of the contingency table with $n_{11} = i$ to compute probability of contingency table where $n_{11} = i+1$.

First, the probability of the contingency table with smallest possible n_{11} count is calculated. The probability value is then used to compute the probability of rest of the contingency tables. The expression $max\{0, n_{1p}+n_{p1}-n_{pp}\}$ gives the smallest value that can be assigned to n_{11} for the given marginal totals. Once probability of the first contingency table has been computed, it is used to compute probability of rest of the possible tables. This is achieved by incrementing n_{11} by a value of 1 and adjusting the rest of the observed frequency values accordingly. Now the probability of the previous table is multiplied with $n_{12}*n_{21}/n_{11}*n_{22}$ to get the probability of the new table. This process is repeated until the probabilities for all the possible contingency tables have been computed.

```
sub computeDistribution()
{
   $n11_start = shift @_;        #get smallest n11 value
   $final_limit = shift @_;      #get maximum n11 value

   #compute probability for first table.
   my $product = computeHypergeometric($d);

   #hash to store all probability values
   my %probability;
   $probability{$d} = $product;

   #compute values for remaining contingency tables
   for ($i = $n11_start+1; $i <= $final_limit; $i++ )
   {
      $subproduct += log $n12;
      $n22++;
      $subproduct -= log $n22;
      $subproduct += log $n21;
      $n12--;
      $n21--;
      $subproduct -= log $i;
      $probability{$i} = $product+$subproduct;
      $product = $product+$subproduct;
      $subproduct=0;
   }
   return \%probabiliy;
}
```

Figure 4: Function to compute probability distribution for all possible tables

Since values associated with bigram data are generally very large, direct division of terms can result in an underflow error causing the intermediate value to be rounded down to zero. Hence any further multiplications or divisions to this value will also result in a zero. Similarly, multiplication can cause an overflow, resulting in an invalid value. To avoid this all computations are done using logarithms to base ***e*,** and all multiplication and division operations are replaced with addition and subtraction operations respectively.

```
sub computeHypergeometric( )
{
  my $d = shift @_;         # n11 value for the contingency table
  my $probability = 0      #initialize the variable to hold the result

  #initialize and sort the numerator and denominator arrays
  my @numerator = sort { $b <=> $a } ($n1p, $np1, $n2p, $np2, 1);
  my @denominator = sort { $b <=> $a } ($npp, $n22, $n12, $n21, $d);

  #compute probability
  for ( my $i = 0; $i <= 4; $i++ )
  {
    if ( $numerator[$i] > $denominator[$i] )
    {
      for (my $j = $denominator[$i]+1; $j <= $numerator[$i], $j++)
      {
       $probability += log($j);
      }
    }
    else
    {
      for (my $j = $numerator[$i]+1; $j <= $denominator[$i], $j++)
      {
       $probability -= log($j);
      }
    }
  }
  return $probability;
}
```

Figure 5: Function to compute hypergeometric probability

Figure 5 lists the implementation of a function to compute the hypergeometric probability. To reduce the amount of computations required to calculate hypergeometric probability for the first table, the factorial terms in numerator and denominator of equation (5) are canceled out.

Figure 6 shows a chart that compares the performance of our implementation with a simple implementation of Fisher's Exact test. As it can be seen in the chart there is a significant improvement in performance.

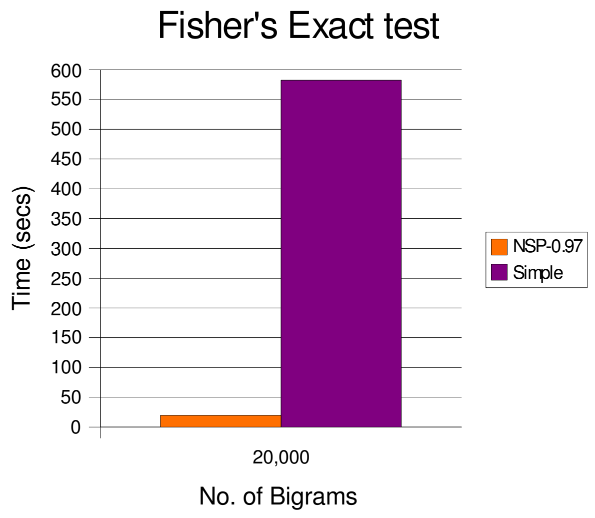


Figure 6: Performance of left sided Fisher's Exact test

3.2.2 Pearson's Chi squared test

In mathematical statistics, the standard asymptotic test for independence is Pearson's Chi Squared [3] test. It compares the observed frequencies n_{ij} with the expected frequencies m_{ij} under the null hypothesis of independence. If the difference between observed and expected frequencies is large, then the null hypothesis is rejected. The Pearson's Chi squared test is defined as

$$X^2 = \sum_{i,j} \frac{(n_{ij} - m_{ij})^2}{m_{ij}} \qquad (8)$$

For large samples, the test statistic of Pearson's Chi-squared test has an asymptotic X^2 (Chi-squared) distribution with one degree of freedom. The Pearson's Chi squared measure can easily extended to Ngrams of any size. Equation (9) shows the formulation for trigrams. The Pearson's Chi Squared test is more accurate than the Log-likelihood measures for sparse contingency tables.

$$X^2 = \sum_{i,j,k} \frac{(n_{ijk} - m_{ijk})^2}{m_{ijk}} \tag{9}$$

3.2.3 Log-likelihood ratio

The Log-likelihood [2] ratio checks the probability of sampling the observed contingency table under the null hypothesis, that is the tokens occur together by chance and are independent of each other. The more unlikely it is to witness the given table, the more evidence against the null hypothesis. The log-likelihood measure can be computed as:

$$\text{ll} = 2 * \sum_{i,j} n_{ij} * \log \frac{n_{ij}}{m_{ij}} \tag{10}$$

Note that the the logarithm is undefined when there are empty cells ($n_{ij} = 0$). For such cells, the entire term evaluates to zero, because 0*log(0) = 0, and can simply be omitted from the summation.

Dunning (1993) [2] showed that the accuracy of Log-likelihood measure is better than Pearson's Chi Squared test for highly skewed contingency tables, like the ones observed in Ngram co-occurrence data. The Log-likelihood can also easily extended to Ngrams of any size. The following equation gives the computation for trigrams.

$$\text{ll} = 2 * \sum_{i,j,k} n_{ijk} * \log \frac{n_{ijk}}{m_{ijk}} \tag{11}$$

Similarly the Log-likelihood measure can be extended for longer Ngrams.

3.2.4 T-score

The t-score [4] is a heuristic variation of z-score which itself is an approximation of the Binomial Exact test. The z-score measure uses the normal distribution to approximate the binomial probability of all possible contingency tables for which the expected value m_{11} is greater than the observed frequency count n_{11}.

$$z-score=\frac{(n_{11}-m_{11})}{\sqrt{m_{11}}} \tag{12}$$

However the z-score tends to overestimate Ngrams with low expected values. The cause of this problem is the presence of m_{11} in the denominator. Here $\sqrt{m_{11}}$ is used to approximate the variance of the distribution under the null hypothesis. The t-score solves this problem by estimating this variance using the observed sample. The t-score is computed as:

$$t-score=\frac{(n_{11}-m_{11})}{\sqrt{n_{11}}} \tag{13}$$

The t-score measure can also be extended to Ngrams of any size. For trigrams the computation is:

$$t-score=\frac{(n_{111}-m_{111})}{\sqrt{n_{111}}} \tag{14}$$

3.3 Mutual Information based measures

The Mutual information based measures are motivated from Information Theory. They try to measure the reduction in uncertainty of one random variable due to the knowledge about another random variable. In other words, they try to measures the amount of information one random variable provides about the other. Here we can assume the tokens in a Ngram being represented by variables

3.3.1 Pointwise Mutual Information (*)

The use of Pointwise Mutual Information(PMI) as a measure to identify Ngrams was proposed by Church and Hanks [6]. Pointwise Mutual Information measures the overlap between two events. In other words it measures the amount of information the occurrence of an event provides about the occurrence of another event. For example, consider the bigram united<>states<>. Pointwise Mutual Information will quantify the increase in the amount of information we have about the occurrence of states<> at position *i+1* in the corpus, if we know that united<> occurs at position *i*. The Pointwise Mutual Information is defined as:

$$PMI = \log \frac{P(token1, token2)}{P(token1) * P(token2)} \tag{15}$$

The probability *P(token1, token2)* can be computed as:

$$P(token1, token2) = \frac{n_{11}}{n_{pp}} \tag{16}$$

and the probabilities *P(token1)* and *P(token2)* are given by n_{1p}/n_{pp} and n_{p1}/n_{pp} respectively. Under null hypothesis the expected value can be computed as:

$$m_{11} = \frac{(n_{1p} * n_{p1})}{n_{pp}} \tag{17}$$

Substituting equations (16) and (17) in equation (15) gives us the following computation for PMI.

$$PMI = \log\left(\frac{n_{11}}{m_{11}}\right) \tag{18}$$

The Pointwise Mutual Information tends to overestimate Ngrams with low observed frequency counts. To prevent overestimation of low frequency data, the PMI measure can be modified by increasing the influence of the observed co-occurrence frequency n_{11} in the numerator. This is done by raising the observed co-occurrence frequency by some exponent *e,* where *e* > 1. Daille(1994) [12] considered versions of PMI with *e* = 2....10, obtaining the best performance for *e* = 3.

$$PMI = \log \frac{n_{11}^{e}}{m_{11}} \quad (19)$$

The implementation of Pointwise Mutual Information measure was modified in NSP to accept a floating point parameter *e*. If no value for *e* is provided, the measure computes the standard PMI statistic. Otherwise the value *e* is used to compute the modified Pointwise Mutual Information. The program *statistic.pl* was also modified to provide a new command-line option ***--pmi_exp***. This option is only available for the PMI measure and can be used to specify the *e* value to compute the modified PMI statistic.

3.3.2 True Mutual Information

True Mutual Information tries to measure the amount of information one random variable provides about the other. True Mutual Information can also be defined as the weighted average of the pointwise mutual information for all the observed and expected value pairs.

$$TMI = \sum P(xy) * \log \frac{P(xy)}{P(x) * P(y)} \quad (20)$$

Under our hypothesized model the above mentioned computation can be simplified to

$$TMI = \sum_{i,j} n_{ij} * \log \frac{n_{ij}}{m_{ij}} \quad (21)$$

The true mutual information measure is nearly identical to the Log-likelihood ratio test. In fact they only differ by a factor of 2 and as such produce identical rankings.

3.4 Measures of association strength

These measures test the degree of association between two random variables. They are not measures of association in the statistical sense as they do not assign a significance score. Instead they try to measure how correlated two or more variables are. In our case these variables represent tokens in a Ngram.

Measures of degree of association are not effected by the sample size where as the measures belonging to other families are effected by the sample size. That is, The measures of degree of association are not affected if the values in the contingency table are modified, while maintaining the proportions of the contingency table. For example the values generated by these measures will remain the same even if we multiply all the cells in a contingency table by a factor of 10. Thus these measures must be used for the data where the user wants to nullify the effect of the sample size.

3.4.1 Dice Coefficient

The Dice Coefficient was first used by Smadja(1999) [3] for the extraction of collocations from text corpora. The Dice Coefficient computes the harmonic mean of the marginal totals n_{1p} and n_{p1}. A higher value for the Dice Coefficient indicates that the two tokens do not occur together by chance and hence the bigram is significant.

$$Dice = \frac{2 * n_{11}}{n_{1p} + n_{p1}} \tag{22}$$

The Dice coefficient can be easily extended to Ngrams. The computation for trigrams is:

$$Dice = \frac{3 * n_{111}}{n_{1pp} + n_{p1p} + n_{pp1}} \tag{23}$$

3.4.2 Phi Coefficient

The Phi Coefficient is a measure of the degree of association between two binary variables. For bigrams, these variables represent tokens in a bigrams and indicate whether or not a particular token occurs at its position or not.

The Phi Coefficient is computed as:

$$phi = \frac{n_{11} * n_{22} - n_{12} * n_{21}}{\sqrt{n_{1p} * n_{p1} * n_{2p} * n_{p2}}} \tag{24}$$

Church(1991)[11] used phi^2 instead of phi to identify collocations in text corpora.

$$phi^2 = \frac{(n_{11} * n_{22} - n_{12} * n_{21})^2}{n_{1p} * n_{p1} * n_{2p} * n_{p2}} \tag{25}$$

The phi^2 measure is equivalent to Pearson's Chi-Squared test multiplied by the sample size.

$$X^2 = n_{pp} * phi^2 \tag{26}$$

3.4.3 Odds Ratio

For a bigram this measure computes the ratio of odds of witnessing the second token with the first token to the odds of witnessing second token occurring with some other token. The odds of witnessing the second token with the first token are given by n_{11}/n_{12}. Similarly the odds of witnessing the second token with some other token are given by n_{21}/n_{22}.

$$odds = \frac{n_{11} * n_{22}}{n_{12} * n_{21}} \tag{27}$$

This measure is not very effective in identifying bigrams, this is so because one of the numerators is the value in cell n_{22}, which is generally very large for bigram data. Also, intuitively this measure may not be suited for identifying collocations, because it measures the chances of witnessing the second token with the first token rather than measuring the chances of the two tokens occurring together.

The value of odds-ratio is undefined when one of the non-diagonal cells has a count of zero, $n_{12} = 0$ or $n_{21} = 0$. To avoid this, the NSP implementation of the odds ratio adds 1 to each of the cell counts, before the ratio is calculated.

3.4.4 Jaccard Coefficient (*)

The Jaccard Coefficient computes the ratio of the bigram frequency count to the number of times at least one of the constituent tokens occurs in the correct position.

The Jaccard Coefficient is computed as:

$$Jaccard=\frac{n_{11}}{n_{11}+n_{12}+n_{21}} \qquad (28)$$

The Jaccard and Dice Coefficients are similar to each other. In fact, there is a monotonic transformation from Dice Coefficient to Jaccard Coefficient.

$$Jaccard=\frac{Dice}{2-Dice} \qquad (29)$$

The Jacccard Coefficient is frequently used in Information Retrieval as a measure of association. It is used to measure the degree of association between two variables.

3.5 Likelihood Measures

Likelihood measures try to estimate the probability of seeing a observed contingency table under the null hypothesis that the tokens in a Ngram occur together by chance and are independent of each other.

The smaller this probability, the more unusual the observed outcome and, consequently more evidence against the null hypothesis. These measures use the probability of the observed frequency table as a measure of the evidence against the null hypothesis.

3.5.1 Poisson-Stirling Measure(*)

Poisson Stirling [13] is a Likelihood measure. The Poisson Stirling measure uses the Stirling's formula to approximate the negative logarithm of the Poisson-likelihood measure. It is defined as follows:

$$Poisson-Stirling = n_{11} * (\log \frac{n_{11}}{m_{11}} - 1) \qquad (30)$$

The Poisson likelihood measures uses Poisson distribution to approximate the binomial distribution of witnessing the tokens in a Ngram together.

$$Poisson-likelihood = e^{-m_{11}} * \frac{m_{11}^{\,n_{11}}}{n_{11}!} \qquad (31)$$

The results calculated using Poisson-Stirling measure are comparable to the Log-likelihood ratio, even though it is computationally similar to the Pointwise Mutual Information measure.

4 Redesign of NSP

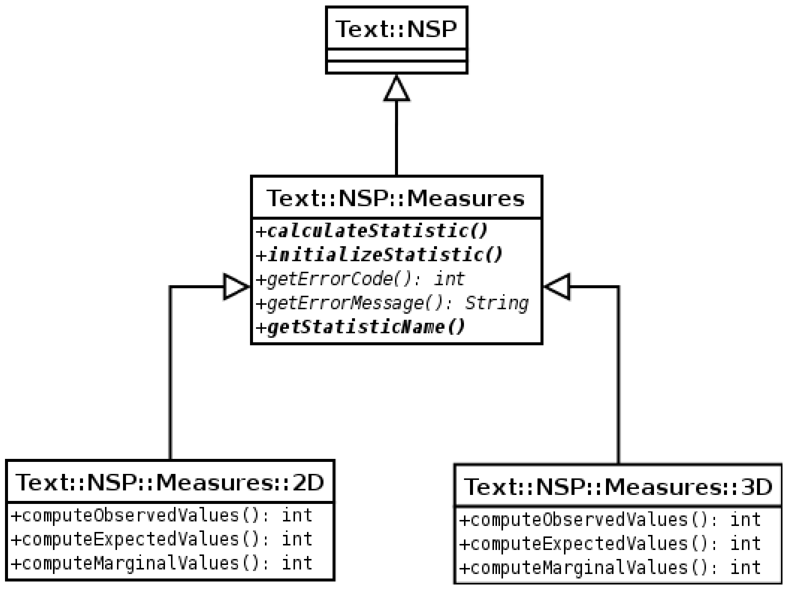
Significant changes have been made to how measures of association are implemented in NSP. There were three main goals for this redesign, first to make the implementation of measures in NSP conform to CPAN[1] guidelines for writing modules. CPAN is Comprehensive Perl Archive Network, a repository for Perl software. The second goal was to improve the maintainability of the implementations by reducing the amount of code duplication. The third goal was to makes it easier for users to implement new measures of association. The measures in NSP were originally designed to be used only with *statistic.pl*. This redesign also makes it possible for users to use the measures of association directly from within their programs. Since many applications have been built over NSP, special care was taken to keep these changes transparent to the existing users of NSP. So significant changes have been made to statistic.pl to allow a smooth transition to the new implementation. Also, we did not want any decline in the performance of the measures.

As stated earlier, in NSP all measures are implemented as Perl modules. CPAN requires that all modules are implemented in the name space declared by the software package. To minimize name space collisions, Perl provides a hierarchal name space for modules. Components of a module name are separated by double colons (::). For NSP the name space is Text::NSP. In the previous implementation (NSP-0.73 and before) all measures were implemented directly under the global name space. Apart from making the implementation non compliant to CPAN this also resulted in significant amount of code duplication. Also, it made it difficult for users to implement new measures. The previous implementation (NSP-0.73 and before) provided two modules, measure2d and measure3d, that implemented methods to compute the observed and expected frequency counts for bigrams and trigrams respectively.

(1) CPAN website: http://www.cpan.org

4.1 Design of NSP-0.95

In this implementation all the measures of association were rewritten using Object Oriented Concepts. Text::NSP::Measure is the base class for all the measures. It lays down the groundwork for implementing measures of association. It implements the framework that allows measures to report errors to *statistic.pl*. This module also provides an abstract implementation of methods that must be overridden by all the measures. Two modules: Text::NSP::Measure::2D and Text::NSP::Measures::3D, are derived from Text::NSP::Measures. These modules provide the basic framework for implementing measures for bigrams and trigrams respectively.



Text::NSP
Text::NSP::Measures
+calculateStatistic()
+initializeStatistic()
+getErrorCode(): int
+getErrorMessage(): String
+getStatisticName()
Text::NSP::Measures::2D
+computeObservedValues(): int
+computeExpectedValues(): int
+computeMarginalValues(): int
Text::NSP::Measures::3D
+computeObservedValues(): int
+computeExpectedValues(): int
+computeMarginalValues(): int

Figure 7: Class diagram of basic framework

Both Text::NSP::Measures::2D and Text::NSP::Measures::3D implement methods to compute the marginal, observed and expected frequencies for a given Ngram. They also implement some basic error checks that are common for all measures. For example Text::NSP::Measures::2D checks whether the computed marginal totals are valid, that is no marginal total is less than zero or greater than the sample size.

All bigram measures inherit from Text::NSP::Measures::2D, while the trigram measures inherit from Text::NSP::Measures::3D. Both of these classes are abstract and hence cannot be instantiated.

4.1.1 Bigram Measures

There are thirteen measures of association for bigrams available in NSP. Measures that are computationally similar have been grouped together under one family. The computations and error checks that are common to each family of measures have been implemented in the base class for that family.

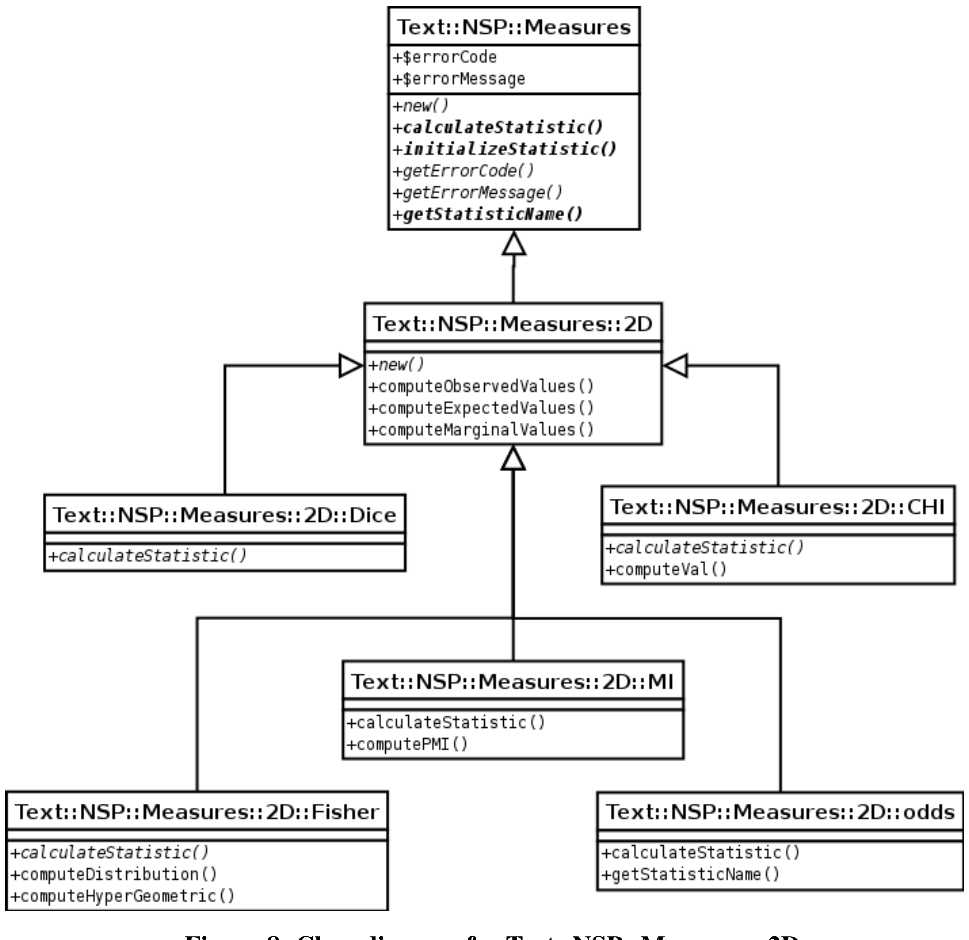


Figure 8: Class diagram for Text::NSP::Measures::2D

The measures that did not belong to any family (like the Odds Ratio) have been implemented directly under Text::NSP::Measures::2D.

The module Text::NSP::Measures::2D::MI is the base class for Log-likelihood, Total Mutual Information, Pointwise Mutual Information and Poisson-Stirling measures. Text::NSP::Measures::2D::MI is a abstract class, hence it cannot be instantiated. This class implements a method to test the values computed by Text::NSP Measures::2D for errors that are specific to these measures.

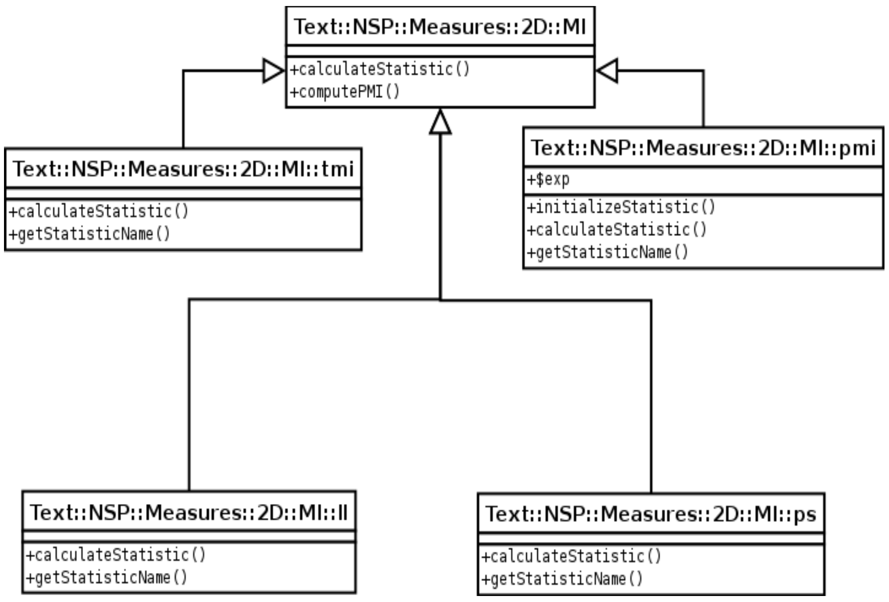


Figure 9: Class diagram for Text::NSP::Measures::2D::MI

Text::NSP::Measures::NSP::MI also implements a method to compute the log of the ratio of a pair of values. This is the same as computing Pointwise Mutual Information for a given cell in the contingency table. This computation is common in all the measures implemented under this class.

The module Text::NSP::Measures::2D::Fisher is the base class for the Fisher's Exact tests. It implements methods to compute the probabilities of all possible contingency tables for the observed marginal totals. This method returns a hash containing all these probability values.

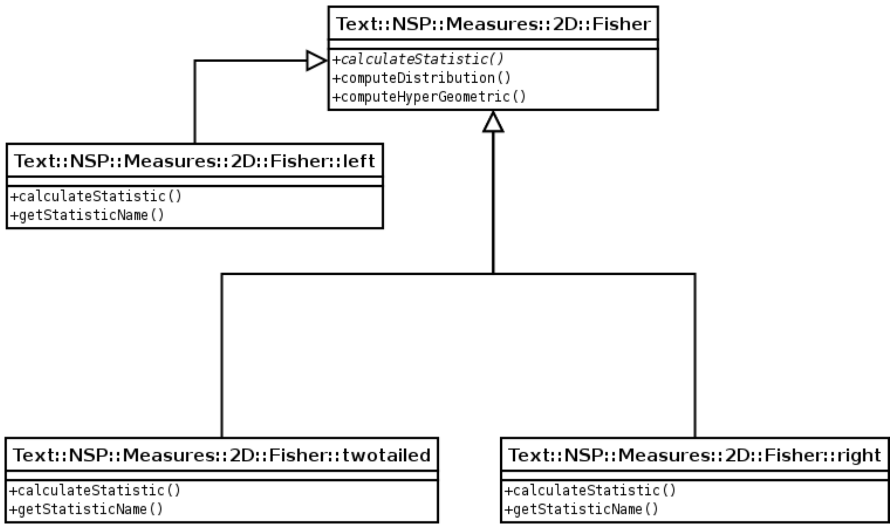


Figure 10: Class Diagram for Text::NSP::Measures::2D::Fisher

Text::NSP::Measures::2D::Dice is the base class for implementing the Dice and Jaccard Coefficient measures. This module provides a method to compute the value for the Dice Coefficient. Text::NSP::Measures::2D::Dice::dice simply calls this method and returns the values computed in Text::Measures::2D::Dice, while the implementations of the Jaccard coefficient, Text::NSP::Measures ::2D::Dice::jaccard, applies a transformation to the value of Dice coefficient to get the result.

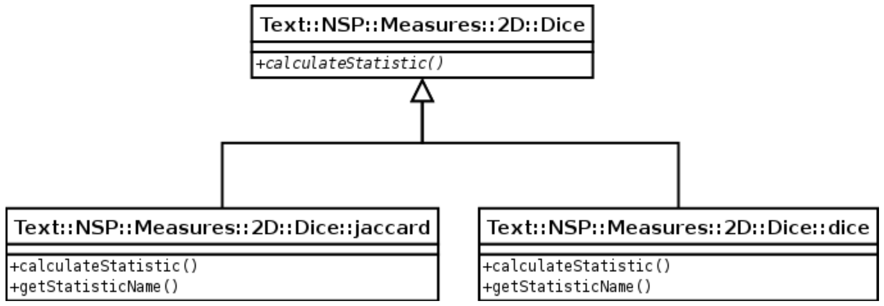

Text::NSP::Measures::2D::Dice
+calculateStatistic()
Text::NSP::Measures::2D::Dice::jaccard
+calculateStatistic()
+getStatisticName()
Text::NSP::Measures::2D::Dice::dice
+calculateStatistic()
+getStatisticName()

Figure 11: Class diagram for Text::NSP::Measures::2D::MI::Dice

Text::NSP::Measures::2D::CHI is the base class for Pearson's CHI squared, Phi coefficient and the t-score measures. It implements some error checks that are common for these set of measures.

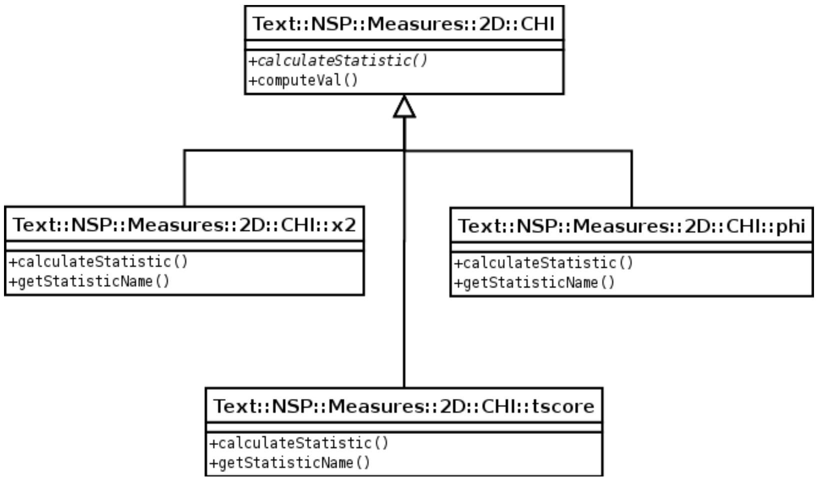


Figure 12: Class diagram for Text::NSP::Measures::2D::MI::CHI

4.1.2 Trigram Measures

In NSP, there are four measures of association available for trigrams. All these measures inherit from a single class, Text::NSP::Measures::3D::MI, which in turn inherits the functionality implemented in Text::NSP::Measures::3D. As in the bigram implementation Text::NSP::Measures::3D::MI is the base class for Log-likelihood, Total Mutual Information, Pointwise Mutual Information and Poisson-Stirling measures. This class provides error checks that are common for these measures.

4.2 NSP-0.97

Perl originated in 1987 and was not originally designed as an objected oriented language. Support for object oriented programming was introduced in Perl5 which was released in 1994. This support was build on top of the existing Perl components.

As a result of this, object-oriented Perl isn't as fast as non object-oriented Perl [14]. Calling a method through an object is significantly slower than calling a regular Perl subroutine. A single method call can be about 30 percent slower than a regular call to the same subroutine. Since all measures in NSP-0.95 were implemented using object oriented Perl, this resulted in a significant slowdown of performance when compared to the earlier versions.

NSP-0.97 addressed this issue by modifying the implementation of measures to not use object oriented features available in NSP. This paradigm shift required some major changes to the implementation of the measures, since object oriented features like polymorphism and inheritance could not be used any longer. This was achieved by importing the required subroutines and variables from the modules higher in the hierarchy. This basic functionality for implementing measures is implemented in the higher modules. If a measure needs to extend this functionality. It can redefine the subroutine that implements this functionality.

We compared the performance of Log-likelihood implementation in NSP-0.73, NSP-0.95 and NSP-0.97. For this comparison we ranked 5 lists of bigrams consisting of 20000, 40000, 60000, 80000 and 100000 bigrams respectively. Figure 13 shows the profile results thus generated.

To generate these profile results we used the Devel::DProf Perl module. This module collects information on the execution times of Perl scripts and subroutines. This information is then written to a file named *tmon.out.* This data can then be further analyzed using the program *dprofpp*. Appendix C shows a sample output from dprofpp.

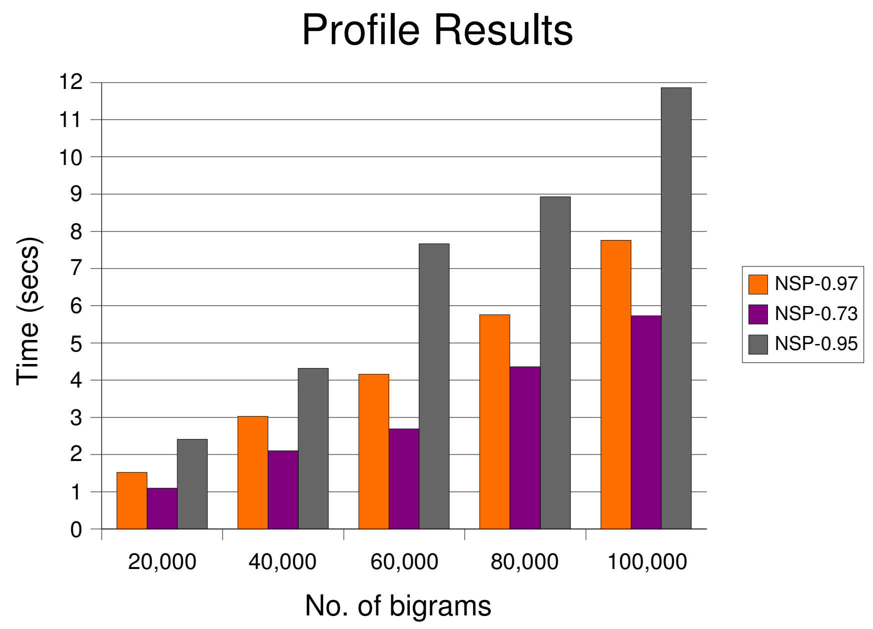


Figure 13: Profile results for Log-likelihood measure in NSP-0.95 and NSP-0.97

As seen in Figure 13 there is a significant improvement in the performance of NSP-0.97 as compared to NSP-0.95. But there is still some slowdown in the performance of the measures. This is because the current implementation requires more subroutine calls, which causes some extra overhead. This cannot be avoided because these subroutines were defined to reduce the amount of code duplication across the measures.

5 Experimenal Results

The algorithms and new measures described in this project report were tested using a subset of the New York Times Newswire Service data available from the English Gigaword Corpus, which was produced by the Linguistic Data Consortium. It consists of newswire text data collected from four newswire services, the Agency France Press English services (AFPE), Associated Press Worldstream English Service (APW), Xinhua News Agency English Service (XIE) and the New York Times Newswire Service (NYT).

For our experiments we used the New York Times data from the year 2002. It has 200,000 lines of text and approximately 8,000,000 tokens. We used *count.pl* to count all the bigrams in this corpus. We used a standard English stop list to remove the non-content words in this corpus. This list can be found in Appendix B.

5.1 Spearman's Rank Correlation Coefficient

We compared the new measures of association that have been incorporated in NSP, by comparing their results with those of the Log-likelihood measure. This measures was used as the standard because it is one of the most popular measures of association, thus comparing the new measures of association with Log-likelihood measure will allow us to make some observations about the how the measures perform and also about how these new measures can be useful. For this purpose we used the program rank.pl to compute the Spearman's Rank Coefficient between the output of the new measure and output from the Log-likelihood measure.

5.2 Two-tailed Fisher's Exact Test

Table 11 shows the top ten bigrams identified by the two-tailed Fisher's Exact test. None of the bigrams in this list seem interesting at least in an intuitive way.

Table 11: Top ten bigrams identified by two-tailed Fisher's Exact test

_<>1<>1 1.0000 52 11288 7599
art<>_<>1 1.0000 20 1288 26470
_<>bush<>1 1.0000 19 11288 2759
play<>_<>1 1.0000 14 915 26470
administration<>_<>1 1.0000 13 861 26470
internet<>_<>1 1.0000 12 775 26470
minute<>_<>1 1.0000 9 603 26470
board<>_<>1 1.0000 9 588 26470
x_<>life<>1 1.0000 9 11288 1349
record<>_<>1 1.0000 9 591 26470

The value for Spearman's Rank Correlation coefficient for the two-tailed test is -0.7444. which indicates that there is a negative correlation between the ranks of the bigrams generated by the two-tailed test and the Log-likelihood measure.

5.3 Pointwise Mutual Information

We compared variations of Pointwise Mutual Information measure. Table 12 shows the top 10 bigrams identified by Pointwise Mutual information for the values e = 1, 2, 3. As it can be seen there are some significant changes in the list of bigrams. Thus by changing the value of e users can identify different sets of bigrams. Intuitively the list of bigrams for e=3 is appealing as this corresponds most closely to what we would think of as collocations.

Table 12: Top ten bigrams identified by variations of PMI

PMI for e=1	*PMI with e=2*	*PMI with e=3*
jeroboam<>methuselah<>	5887<>toder<>	united<>states<>
pteridologists<>botanically<>	8334<>andya<>	journal<>constitution<>
uptempo<>polyrhythmic<>	8320<>chuckh<>	atlanta<>journal<>
steerage<>cavernous<>	8338<>artd<>	sept<>11<>
jeev<>milkha<>	8348<>rickm<>	los<>angeles<>
3s<>ae32i<>	7282<>agordon<>	news<>service<>
buyable<>guilder<>	kollar<>kotelly<>	optional<>trim<>
caine<>mutiny<>	hospitalization<>nns14<>	cox<>newspapers<>
algernon<>moncrieff<>	amorim<>sicherle<>	story<>filed<>

We also compared variations of the Pointwise Mutual Information with the Log-likelihood ratio, by computing the Spearman's Rank coefficient. As it can be seen from the list of bigrams, different values of *e* can be used to identify different sets of bigrams. Smaller *e* values, like 1 and 2, can be used to identify rare bigrams, since these variations tend to overestimate bigrams with lower frequency counts where as the values 3 and 4 may be used to identify significant bigrams that occur more regularly.

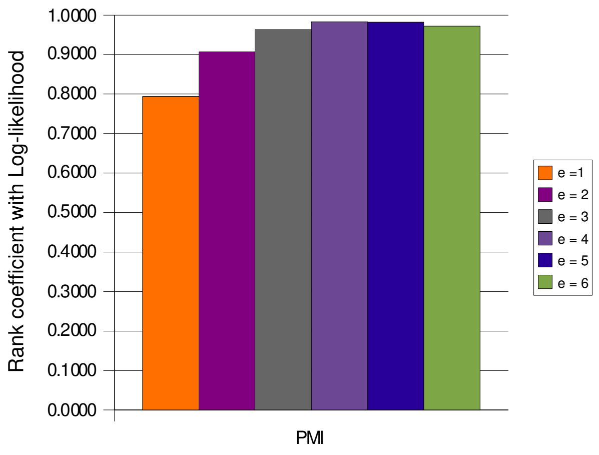


Figure 14: Spearman's Rank Coefficient

5.4 Jaccard Coefficient

The Spearman's Rank Coefficient between the output of Jaccard Coefficient and Dice Coefficient is 0.9998. Such a high value indicates that these measures are almost identical and could be used interchangeably. This is not surprising given the the formulation of the Jaccard coefficient. The Spearman's Rank Coefficient for Log-likelihood measure and Jaccard was 0.8008, which is fairly high and the positive correlation means that Jaccard Coefficient is quite similar to the Log-likelihood measure. This is not entirely unexpected, even though the formulation of Log-likelihood and Jaccard are quite different.

5.5 Poisson-Stirling Measure

The Rank Correlation Coefficient for Poisson-Stirling measure and the Log-likelihood measure is 0.9968. Such a high positive correlation indicates that the two measures are quite similar. Table 12 shows the top ten bigram identified by the Poisson-Stirling measure.

Table 13: Top ten bigrams identified by Poisson-Stirling

united<>states<>1 17366.1016 3590 4033 4234
journal<>constitution<>2 12241.1258 2235 2438 2300
atlanta<>journal<>3 11268.6698 2248 3634 2469
sept<>11<>4 9918.0198 1912 2198 2916
news<>service<>5 9224.1969 2110 4290 3727
los<>angeles<>6 7706.6391 1291 1532 1292
cox<>newspapers<>7 7669.3323 1465 2834 1652
optional<>trim<>8 7607.0813 1430 2741 1532
white<>house<>9 7334.4866 1529 2875 2634
story<>filed<>10 7198.9907 1319 2003 1684

Since Poisson-Stirling measure is computationally quite similar to the Pointwise Mutual information. We compared it with Pointwise Mutual Information as well. The correlation between these two measures was 0.7974. Although this value is fairly high, it still indicates that these two measures are not identical and hence can be used to identify different sets of Ngrams.

6 Future Work and Conclusions

In this section we discuss some of the features that can be added to NSP in the near future.

Program *count.pl* can be implemented in a more modular fashion, thus reducing the amount of code duplication. Threads can also be used to speed up the process of identifying and counting bigrams in large corpora of text. Also, work can be done to make *count.pl* more memory efficient.

The program *count.pl* can also be extended to get counts of Ngrams from the World Wide Web. It can also be extended to provide support for the Unicode character set. Also, count.pl can be modified to allow frequency cutoff for frequently occurring tokens. This can be done to automatically identify and remove stop words.

Some of the existing measures can be extended for trigrams and if possible 4-grams. Some more measures of association can be implemented. Program *statistic.pl* can be modified to allow sorting in ascending order of scores. This might be useful for certain measures such as the right sided Fisher's Exact test.

To conclude, we summarize some of the accomplishments of this project. We have redesigned Ngram Statistics Package using object-oriented concepts without any significant slowdown in performance. As part of this redesign we have grouped the various measures into families based on their computations. We have also implemented some new measures such as Poisson-Stirling, Jaccard Coeffcient and two-tailed Fisher's Exact test.

We have designed and implemented an optimization to the left and right sided Fisher's Exact tests. We have also implemented a variation to the Pointwise Mutual Information measure that reduces the overestimation of low frequency Ngrams by PMI.

References

[1] Pedersen, Ted (1996). Fishing for Exactness. In *Proceedings of the South-Central SAS Users Group Conference*. Austin, TX.

[2] Dunning, Ted (1993). Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics* **19**(1), 61-74.

[3] Manning, Christopher D. and Schütze, Hinrich (1999). *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA.

[4] Church, Kenneth W.; Gale, William; Hanks, Patrick; Hindle, Donald (1991). Using statistics in lexical analysis. In *Lexical Acquisition: Using On-line Resources to Build a Lexicon*, Lawrence Erlbaum, pages 115-164.

[5] Evert, Stefan and Krenn, Brigitte (2001). Methods for the qualitative evaluation of lexical association measures. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*. Toulouse, France, pages 188-195.

[6] Church, Kenneth W. and Hanks, Patrick (1990). Word association norms, mutual information, and lexicography. *Computational Linguistics* **16**(1), 22-29.

[7] Moore, Robert C (2004). On Log-Likelihood-Ratios and the Significance of Rare Events, *Proceedings of EMNLP 2004*, 333-340.

[8] Smadja, Frank (1993). Retrieving collocations from text: Xtract. *Computational Linguistics* **19**(1), 143-177.

[9] Dias, Gaël; Guilloré, Sylvie; Lopes, José G. P. (1999). Language independent automatic acquisition of rigid multiword units from unrestricted text corpora. In

Proceedings of Traitement Automatique des Langues Naturelles (TALN), Cargèse, France.

[10] Kuhns, J. L. (1965). The continuum of coefficients of association. In *Proceedings of the Symposium on Statistical Association Methods For Mechanized Documentation*, Washington, DC, pages 33-39.

[11] Church, K. (1991) Concordances for Parallel Text, In *proceeding of the Seventh Annual Conference of the UW Centre for the New OED and Text Research*, Oxford, England.

[12] Daille, Béatrice (1994). *Approche mixte pour l'extraction automatique de terminologie: statistiques lexicales et filtres linguistiques.* PhD thesis, Université Paris 7.

[13] Quasthoff, Uwe and Wolff, Christian (2002). The Poisson collocation measure and its application. In *Workshop on Computational Approaches to Collocations*. Vienna, Austria.

[14] Conway Damian (1999). *Object Oriented Perl.* Manning Publications Co., Greenwich, CT.

[15] Banerjee S. and Pedersen T. (2003). The Design, Implementation and Use of the Ngram Statistics Package, In *Proceedings of Conference on Intelligent Text Processing and Computational Linguistics.*

[16] Shannon C.E. (1948). A Mathematical Theory of Communication, *Bell System Technical Journal*, vol. 27, pp. 379-423, 623-656, July, October, 1948.

[17] Wall Larry, Christiansen Tom and Orwant Jon (2000). *Programming Perl*, O'Reilly Media Inc., Cambridge, MA.

Appendix A

Here is a sample implementation of a measure that computes the sum of all the observed frequency cells (i.e. n_{11}, n_{12}, n_{21} and n_{22}).

```
package Text::NSP::Measures::2D::sum;
use Text::NSP::Measures::2D::MI::2D;
use strict;
use Carp;
use warnings;
no warnings 'redefine';
require Exporter;

our ($VERSION, @EXPORT, @ISA);

@ISA  = qw(Exporter);

@EXPORT = qw(initializeStatistic calculateStatistic
             getErrorCode getErrorMessage getStatisticName);

$VERSION = '0.01';

sub calculateStatistic
{
    my %values = @_;

    # computes and returns the marginal totals from the frequency
    # combination values. returns undef if there is an error in
    # the computation or the values are inconsistent.
    if(!(Text::NSP::Measures::2D::computeMarginalTotals($values)) ){
        return;
    }
    # computes and returns the observed and marginal values from
    # the frequency combination values. returns 0 if there is an
    # error in the computation or the values are inconsistent.
    if( !(Text::NSP::Measures::2D::computeObservedValues($values)) )
{
          return;
    }


    #  Now for the actual calculation of the association measure
    my $NewMeasure = 0;

    $NewMeasure += $n11;
    $NewMeasure += $n12;
    $NewMeasure += $n21;
    $NewMeasure += $n22;

    return ( $NewMeasure );
}

sub getStatisticName
{
    return "Sum";
}
```

Appendix B

The following is the list of stop words that were removed from the input text.

a	did	i	not	sixteen	various
aboard	do	idem	nothing	sixth	versus
about	doe	if	notwithstanding	so	very
above	does	ii	now	some	vi
across	doing	iii	ns	somebody	via
after	done	ilk	nt	someone	vii
again	dont	in	o	something	viii
against	down	include	of	sometimes	viiii
all	dr	included	off	somewhat	vis-a-vis
along	during	including	often	soon	w
alongside	e	indeed	on	sooner	wa
already	each	inside	once	sr	want
also	ec	instead	one	such	wanted
although	ee	into	oneself	suchlike	wants
always	eight	is	only	suddenly	was
am	eighteen	it	onto	sundry	we
amid	eighth	its	opposite	t	well
amidst	either	itself	or	take	went
among	eleven	iv	other	ten	were
amongst	else	j	others	tenth	what
an	end	jr	otherwise	than	whatall
and	enough	just	ought	that	whatever
another	especially	k	our	the	whatsoever
anti	etc	kept	ourself	thee	when
any	even	know	ourselves	their	where
anybody	ever	l	out	theirs	whereas
anyone	every	last	outside	them	whereby
anything	everybody	late	over	themselves	wherewith
are	everyone	later	own	then	wherewithal
around	except	less	p	then	which
as	excepting	let	part	there	whichever
astride	excluding	like	particular	they	whichsoever
at	f	little	past	thine	while
aught	few	m	pe	third	who
away	fewer	made	pending	thirteen	whoever
b	fifteen	make	per	this	whole
back	fifth	making	perhaps	those	whom
bar	first	many	plenty	thou	whomever
barring	five	may	plus	though	whomso
be	following	me	probably	three	whomsoever

because	for	might	puts	thrice	whose
become	four	mine	q	through	whosoever
becomes	fourteen	minus	quite	throughout	will
becoming	fourth	mm	r	thus	with
been	from	more	rather	thyself	within
before	g	most	really	till	without
behind	get	mostly	recent	to	wont
being	give	mr	regarding	too	worth
below	go	mrs	relate	totally	would
beneath	going	much	round	tother	x
beside	good	must	s	toward	y
besides	got	my	said	towards	ye
between	h	myself	save	twain	year
beyond	ha	n	saw	twelve	years
both	had	naught	say	twenty	yes
but	hardly	near	says	twice	yet
by	has	need	second	two	yon
c	have	needed	see	u	yonder
called	he	needs	seem	under	you
can	held	neither	seems	underneath	you-all
cannot	her	never	seen	unless	your
cant	here	new	self	unlike	yours
certain	hers	next	seven	until	yourself
circa	herself	nhs	seventeen	up	yourselves
cm	hes	nine	seventh	upon	z
concerning	him	nineteen	several	upper	\\.
considering	himself	ninth	shall	us	\\,
contain	his	no	she	use	:
could	hisself	nobody	short	used	;
d	hm	non	should	usually	\\"
de	how	none	since	ux	\\'
despite	i	nor	six	v	\\`

Appendix C

Sample output from ***dprofpp***:

```
Total Elapsed Time = 9.609147 Seconds
  User+System Time = 7.759147 Seconds
Exclusive Times
%Time ExclSec CumulS #Calls sec/call Csec/c  Name
 19.4   1.510  1.510      1   1.5100 1.5100  main::unformattedPrinting
 15.5   1.210  4.410 100000   0.0000 0.0000  Text::NSP::Measures::2D::MI::ll::calculateStatistic
 12.5   0.970  0.970 400000   0.0000 0.0000  Text::NSP::Measures::2D::MI::computePMI
 10.1   0.790  0.790 100000   0.0000 0.0000  Text::NSP::Measures::2D::computeMarginalTotals
 7.86   0.610  2.230 100000   0.0000 0.0000  Text::NSP::Measures::2D::MI::getValues
 5.54   0.430  0.430 100000   0.0000 0.0000  Text::NSP::Measures::2D::computeExpectedValues
 5.16   0.400  0.400 100000   0.0000 0.0000  Text::NSP::Measures::2D::computeObservedValues
 1.29   0.100  0.100 100000   0.0000 0.0000  Text::NSP::Measures::getErrorCode
 0.26   0.020  0.020      4   0.0050 0.0049  main::BEGIN
 0.00   0.000  0.000      4   0.0000 0.0000  Exporter::Heavy::heavy_export
 0.00       - -0.000      1        -      -  Config::TIEHASH
 0.00       - -0.000      1        -      -  Config::import
 0.00       - -0.000      1        -      -  Getopt::Long::Configure
 0.00       - -0.000      1        -      -  warnings::BEGIN
 0.00       - -0.000      1        -      -  Getopt::Long::ConfigDefaults
```

Appendix D

The following table shows the N x N Correlation matrix for all the measures implemented in Ngram Statistics Package.

	Log-likelihood	TMI	PMI	Poisson-Stirling	Chi Squared	T-Score	PHI Coefficient	Dice Coefficient	Jaccard Coefficient	left Fisher's	right Fisher's	two-tailed Fisher's	Odds Ratio
Log-likelihood	1	-0.8872	0.7941	0.9968	0.9049	0.9129	0.8208	0.7997	0.8008	0.4435	-0.7429	-0.7444	0.8006
TMI	-0.8872	1.0000	-0.8882	-0.8871	-0.8876	-0.8749	-0.2259	-0.8654	-0.8440	-0.3691	-0.6056	-0.6053	-0.8881
PMI	0.7941	-0.8882	1.0000	0.7974	0.9725	0.5830	0.8795	0.8189	0.8200	0.4019	-0.5910	-0.5897	0.9987
Poisson-Stirling	0.9968	-0.8871	0.7974	1.0000	0.9063	0.9068	0.8208	0.8128	0.8139	0.4490	-0.7437	-0.7425	0.8037
Pearson's Chi Squared	0.9049	-0.8876	0.9725	0.9063	1.0000	0.7276	0.9012	0.8605	0.8613	0.4503	-0.6730	-0.6723	0.9743
T-Score	0.9129	-0.8749	0.5830	0.9068	0.7276	1.0000	0.6917	0.6778	0.6799	0.2833	-0.6894	-0.6881	0.5907
PHI Coefficient	0.8208	-0.2259	0.8795	0.8208	0.9012	0.6917	1.0000	0.8067	0.8117	0.4489	-0.1891	-0.1887	0.8812
Dice Coefficient	0.7997	-0.8654	0.8189	0.8128	0.8605	0.6778	0.8067	1.0000	0.9998	0.2866	-0.5734	-0.5721	0.8103
Jaccard Coefficient	0.8008	-0.8440	0.8200	0.8139	0.8613	0.6799	0.8117	0.9998	1.0000	0.2927	-0.5612	-0.5599	0.8115
left Fisher's	0.4435	-0.3691	0.4019	0.4490	0.4503	0.2833	0.4489	0.2866	0.2927	1.0000	-1.5609	-1.5590	0.4030
right Fisher's	-0.7429	-0.6056	-0.5910	-0.7437	-0.6730	-0.6894	-0.1891	-0.5734	-0.5612	-1.5609	1.0000	0.9989	-0.5941
two-tailed Fisher's	-0.7444	-0.6053	-0.5897	-0.7425	-0.6723	-0.6881	-0.1887	-0.5721	-0.5599	-1.5590	0.9989	1.0000	-0.5928
Odds Ratio	0.8006	-0.8881	0.9987	0.8037	0.9743	0.5907	0.8812	0.8103	0.8115	0.4030	-0.5941	-0.5928	1.0000