Evaluation with Uncertainty

A thesis submitted for the degree of

Doctor of Philosophy

Gaya Kanishka Jayasinghe B.Sc. (Eng.),

School of Computer Science and Information Technology,

College of Science, Engineering and Health,

RMIT University,

Melbourne, Victoria, Australia.

28^{th} August, 2014

Declaration

I certify that except where due acknowledgement has been made, the work is that of the author alone; the work has not been submitted previously, in whole or in part, to qualify for any other academic award; the content of the thesis/project is the result of work which has been carried out since the official commencement date of the approved research program; any editorial work, paid or unpaid, carried out by a third party is acknowledged; and, ethics procedures and guidelines have been followed.

Gaya Kanishka Jayasinghe
School of Computer Science and Information Technology
RMIT University
28^{th} August, 2014

Acknowledgments

Many people have supported and encouraged me to undertake this life changing experience at RMIT University. First, I wish to thank my primary advisor Shane Culpepper for giving me this opportunity, constantly supporting me though out this journey, reading this thesis, having faith and being patient during various setbacks and personal struggles. I would also like to thank my thesis committee member Mark Sanderson for encouragement, reading this thesis, and constant support. The guidance, intellects and unfalteringly kept standards by both my advisors have always challenged me to find better solutions, and better face various milestones in this journey. Many thanks goes to William Webber, Panlop Zeephongsekul, Jamie Callan, David Lewis, Alistair Moffat, Peter Bertók and my wife Lasitha Dharmasena for many conversations that have helped me to better reshape this project. I would also like to thank administrative staff at School of Computer Science and Information Technology and School of Graduate Research for helping me with various administrative related matters. Another big thanks goes to my fellow graduate students for interesting conversations, supporting me to present my work at numerous occasions and enduring my constant interruptions. Finally, a huge thank you goes out to my parents, my wife and my wife's father for extra ordinary support and bearing the difficulties with me.

Dedicated in loving memory of my grandmother Dayawathi Nanayakkara

Publications arising from this thesis

Portions of the material in this thesis have previously appeared in the following publications:

Chapter 3 was presented in preliminary form at the 37th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '14) [Extending test collection pools without manual runs – G. K. Jayasinghe, W. Webber, M. Sanderson, and J. S. Culpepper, pages 915–918]; and 19th Annual Australasian Document Computing Symposium (ADCS '14) [Improving test collection pools with machine learning – G. K. Jayasinghe, W. Webber, M. Sanderson, and J. S. Culpepper, pages 2–9].

Chapter 4 was presented in preliminary form at the 37th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '14) [Evaluating Non-deterministic Retrieval Systems – Gaya K. Jayasinghe, William Webber, Mark Sanderson, Lasitha S. Dharmasena, J. Shane Culpepper, pages 911–914].

Chapter 5 was published in preliminary form in the Journal of Network and Computer Applications, Volume 38, February 2014 [Efficient and effective realtime prediction of drive-by download attacks – Gaya K. Jayasinghe, J. Shane Culpepper, Peter Bertók, pages 135–149].

Credits

This work was supported by an Australian Postgraduate Award (APA) and the Australian Research Council (DE140100275).

Authorship

The thesis was written in the `Vim` editor on Ubuntu GNU/Linux system, and typeset using the TEX, LATEX 2_ε and BIBTEX document preparation system. The packages: amsmath, amssymb, caption, dsfont, epsf, epsfig, fancyhdr, float, natbib, pagestyle, rotating, setspace, smallhead, url, verbatim, xr, dtklogos, pdflscape, titlesec, fncychap, color, times, multirow, algorithm2e, booktabs, and glossaries were employed. Graphics were prepared using the R, Dia, Dot, and GIMP software. Programs were authored in C, C++, python, R and Octave. Programs were ran on VPAC, and RMIT CSIT computing facilities. For document retrieval Indri search engine is used. Style information is derived from previous theses written by S. M. M. Tahaghoghi, Hugh E. Williams and Timo Volkmer.

Note

Unless otherwise stated, all fractional results have been rounded to the displayed number of decimal figures.

Contents

List of Figures

List of Tables

List of Algorithms

Abstract

Experimental uncertainty arises as a consequence of: (1) bias (systematic error), and (2) variance in measurements. Popular evaluation techniques only account for the variance due to different experimental units, and assume the other sources of uncertainty can be ignored. In this thesis, the impact of other sources of uncertainty on evaluating information retrieval (IR) and classification experiments are investigated. The uncertainty due to: (1) incomplete relevance judgements in IR test collections, (2) non-determinism in IR systems / classifiers, and (3) high variance of classifiers is analysed.

Effectiveness observed on test collections with incomplete relevance judgements can be biased. Probabilistic relevance of retrieved unjudged documents can be predicted based on judged documents, and can be used to reduce test collection bias. Many methods, each specific to a way of scoring retrieval results (evaluation metric) have been proposed to capture the uncertainty due to probabilistic judgements, and varying experimental units. In this work, a method independent of evaluation metrics to account for such uncertainty is proposed. Using the method, the feasibility of predicting the relevance of documents in the context of graded relevance judgements is shown.

To minimise the impact of bias due to incomplete relevance judgements, proper judgement coverage must be maintained. Traditionally, such coverage is achieved via manual retrieval runs that bring together many relevant documents would not otherwise be pooled. Fully automated approaches to generating a pool are studied. The methods are able to identify a large portion of relevant documents that would normally only be found through manual runs, by combining a simple voting approach with machine learning on documents retrieved by automatic runs. The proposed approaches are able to achieve a closer IR system ranking to the traditional approach which depended heavily on manual runs.

The output of IR systems can be non-deterministic. This can be due to the use of sampling within IR algorithms, or unpredictable inputs of users. Evaluating the effectiveness of such IR systems

is difficult because each run of the system will produce a different system instance with varying effectiveness scores, a situation not considered by current approaches in IR evaluation. Using the context of distributed information retrieval as a case study for the investigation, solutions to the problem of two dimensional significance testing where effectiveness vary across topics as well as different system instances are presented. Furthermore, a viable method for effectiveness-efficiency tradeoff comparisons for a non-deterministic IR system is presented.

To accommodate rapid shifts in the landscape of data, modern classifiers utilize automatically constructed feature spaces using a sample of data. Automatic feature generation results in a large feature space. A large feature space can lead to high classifier variance (commonly known as over-fitting), but effective applications can be found in text classification. Demonstrating how a similar concept leads to high variance classifiers and incorrect conclusions for intrusion detection, a classifier with low variance is proposed. By dynamically monitoring the bytecode stream generated by a web browser during rendering, the approach is able to detect previously unseen drive-by download attacks at runtime. The proposed method is effective, space efficient, and performs the analysis with a low performance overhead. With sampling of the bytecode stream effectiveness of the proposed approach can be traded off for time efficiency. How lessons learned in IR evaluation applies to classification problems in an effectiveness-efficiency tradeoff is shown.

Common abbreviations and Symbols

Notation	Description
$\mathcal{N}$	Normal distribution.
Δ	The difference in performance.
α	The level of significance.
β	The rate of type II error.
δ	The degradation in effectiveness as a result of sampling.
γ	Intercept of a linear model.
λ	A smoothing parameter.
μ	Mean.
ω	Weight vector representing the maximum-margin hyperplane of a linear *SVM* classifier.
ρ	A parameter.
σ	σ^2 – variance; σ – standard deviation.
τ	The Kendall's Tau correlation.
Γ	Test statistic.
ε	Error or residual uncaptured by a model.
$\mathcal{A}$	The first IR system/classifier.

Notation	Description
a	Effectiveness observed on IR system/classifier $\mathcal{A}$.
ANOVA	Analysis of variance.
$\mathcal{B}$	The second IR system/classifier.
b	Effectiveness observed on IR system/classifier $\mathcal{B}$.
BS	Number of bootstrap samples.
D	The depth to which ranked results are processed or evaluated.
d	A document.
E	Expected value.
f	Frequency function.
FN	False negative.
FP	False positive.
H	Hypothesis. H_0: Null hypothesis, and H_1: Alternative hypothesis.
HPD	Highest posterior density.
i	An index to traverse a list of items.
IR	Information retrieval.
j	A secondary index to traverse a list of items.
L	A sample.
LME	Linear mixed effect.
$\mathcal{M}$	Average effectiveness for a system on the population.
$\bar{\mathrm{m}}$	Average effectiveness for a system on a sample of data inputs.
MCMC	Markov chain Monte Carlo.
MSE	Mean squared error.
N	Number of sampled topics used for evaluation.
n	An index into topics, where $n \in 1, ..., N$.

Notation	Description
o	An index into repeated observations.
p	Probability.
PCC	Pearson's Correlation Coefficient.
S	Non-deterministic/Deterministic IR system or classifier.
SCC	Spearman's Correlation Coefficient.
sd	Sample standard deviation.
$\mathcal{T}$	Transpose of a matrix/vector.
t	Topic.
TN	True negative.
TP	True positive.
var	Sample variance.
X	A matrix of input data.
x	A random variable / input data.
Y	A vector of class labels.
y	Observed effectiveness / class label.
z	Difference between a paired effectiveness.

Chapter 1

Introduction – Embracing Experimental Uncertainty

Uncertainty is the only certainty there is, and knowing how to live with insecurity is the only security — John Allen Paulos.

True genius resides in the capacity for evaluation of uncertain, hazardous, and conflicting information — Winston Churchill.

Scientific truth is a never ending frontier. Experiments are an essential part of the process to objectively understand scientific truth. A key characteristic of experimentation is the examination of knowledge established so far – the *state of the art* for the problem, and formulation of a potential advancement – a *hypothesis*, which is put into an experimental test [Kempthorne, 1952]. A hypothesis validated via a suitable evaluation supersedes the state of the art, allowing knowledge to advance. In such a framework reproducing the state of the art is of utmost importance. Published research need not be replicable, but should at least lead to the same overall conclusions [Drummond, 2009; Dalle, 2012]. Repeating the state of the art not only validates prior research, but also provides new insights into how state of the art can be advanced [Moonesinghe et al., 2007; Fokkens et al., 2013]. The genesis of such evaluation methodology for *Information Retrieval (IR)* runs back to the Cranfield experiments by Cleverdon [1991] and Thorne [1955]. Establishing a similar evaluation strategy for the relatively new field of *Machine Learning* has commenced with the work by Kibler and Langley

[1988].

For such comparisons the average performance is quantified in terms of a suitable *evaluation metric*. The true value of a quantity, however it is quantified, differs from the corresponding observed value. Such errors cause uncertainty in experiments. Experimental uncertainty consists of two components, namely, (1) *bias* and (2) *variance*. Bias is systematic. For instance errors caused by incorrectly calibrated measuring equipment. In contrast, variation is a consequence of varying experimental conditions randomly, and therefore has no specific pattern in the distribution of errors. Experimental uncertainty can be attributed to many sources. While some are anticipated, others are not. Therefore, whether an experiment sufficiently reflects the reality when confronted by uncertainty is a question that depends on the circumstance, and is difficult to answer [Tannert et al., 2007].

A published non-reproducible result is not productive. Even worse an unreasonable percentage of published research findings are not true [Ioannidis, 2005]. Fokkens et al. [2013] and Ince et al. [2012] argue reproducing prior research is difficult due to incomplete documentation (uncertainty due to unspecified parameters, versioning of software, etc). The focus of this thesis is not on incompleteness of documentation, but rather ignoring bias and variability in results.

As the uncertainty increases in an experiment, the likelihood of deriving incorrect conclusions increases. Even with no malicious intent, no bias, no ignorance, and rigorously handled experimental uncertainty some findings can be incorrect purely as a consequence of chance. Not accounting for uncertainty can make the situation even worse. Hence, a prudent strategy is to first reduce uncertainty, and then account for the remaining uncertainty when evaluating experiments.

Traditionally experimental uncertainty is quantified using a *hypothesis test* or via a *confidence interval*. For a hypothesis test, the researcher plays the devil's advocate by pretending there is no significant difference between two methods - the hypothesised method and the state of the art. If evidence in favour of the false pretence is negligible, there is not much reason to believe that the two methods are similar. Hence the methods are statistically significantly different. Alternatively, a confidence interval, can be used to express an interval of uncertainty. A confidence interval for the difference in performance between two methods that does not include zero implies a significant difference between methods. When properly exercised such a practice minimises the chance of a system worse than the state of the art advancing.

Big datasets or large data populations is a norm in many scientific disciplines. Sampling is used to overcome difficulties associated with processing big datasets or large data populations [Chambers and Skinner, 2003; Cochran, 2007; Deming, 1966; Lohr, 2009; Stuart, 1976]. For example, quality is assessed, data segments (clusters) are recognised, systems are evaluated, knowledge is extracted using

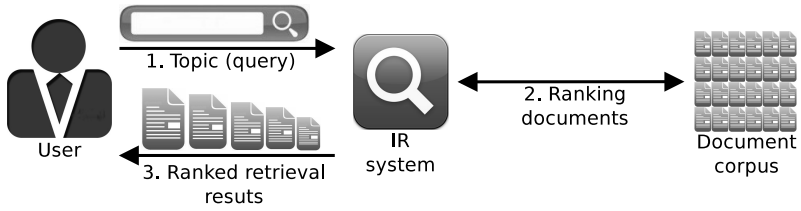


Figure 1.1: Schematic of the ranked document retrieval process. (1). Query is issued to the IR system; (2) Query is processed by scoring documents in the corpus based on probable relevance of each document to the topic; and (3) A ranked list of documents in descending order of scores is returned to the user.

a sample when big datasets or large data populations are involved. Furthermore, manual judgements to evaluate systems are produced only for a sample when test collections with big datasets are formed [Sanderson, 2010]. Sampling can cause experimental uncertainty. In this thesis, the likelihood of drawing incorrect conclusions due to such uncertainty, and hence non-reproducibility is investigated using example applications drawn from two seemingly non-related scientific disciplines, specifically adhoc ranked document retrieval and classification (from the domain of machine learning).

1.1 Uncertainty in Evaluating Ranked Document Retrieval

IR systems rank documents in a *corpus* in response to an information need expressed via a *topic (query)*. The schematic of the process is illustrated in Figure 1.1. Performance for each topic in a set can be evaluated using subject-based methods, where effectiveness is manually quantified, or using a standard *test collection* consisting of a corpus of documents, test topics, and *relevance judgements* for topics on the corpus. Uncertainty in an IR evaluation can be from many sources. In subject-based methods, uncertainty could be a result of user disagreement and/or sampling of topics. Incomplete relevance judgements, assessor disagreement, and sampling of topics causes uncertainty in test collection based evaluation. However, uncertainty due to sources other than sampling of topics is often ignored in the standard evaluation.

1.2 Uncertainty in Evaluating Classification Experiments

A classifier *concept* (or just classifier) categorises new observations based on a prior categorisation, known as the *training dataset*. Here, each observation is transformed into a set of *features*, where each feature quantifies an aspect of the observations. Patterns in the form of features fulfilling a cer-

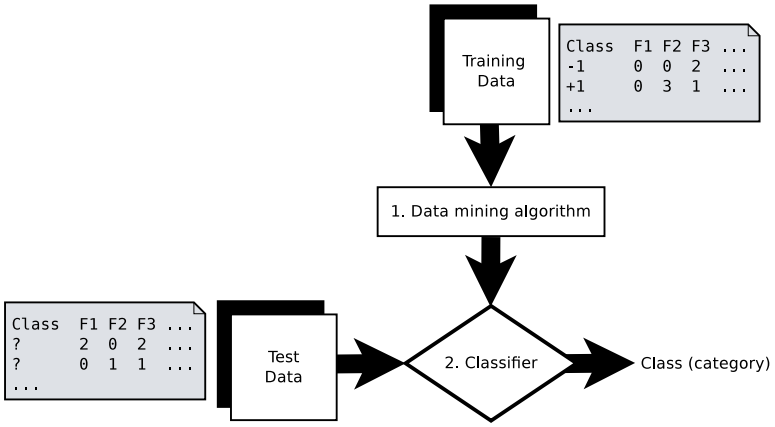


Figure 1.2: Schematic of the classification process. (1) A data mining algorithm extracts patterns in a labelled training dataset, and derives a classifier that is used to categorise new data. (2) The classifier is given a new dataset to classify. When the new dataset is independent of the training dataset, and the label is known is called a test dataset. Such a dataset can be used to measure effectiveness of the classifier.

tain criteria that are useful to make predictions on new observations are extracted from the training dataset. The knowledge extracted as patterns are embedded in a classifier concept. The schematic of a classification process is shown in Figure 1.2. Naturally, the effectiveness of a classifier can be measured on another labelled dataset independent from the training dataset referred to as a *test dataset*. Classifiers are assessed on a sample of varying training:test datasets [Witten and Frank, 2005; Japkowicz and Shah, 2011]. Here only uncertainty due to sampling of training:test datasets is considered. However, uncertainty can arise due to other factors such as assessor disagreement, use of non-deterministic classification algorithms, and the measurement bias. In a standard evaluation, uncertainty due to all other factors other than the variance on different training:test datasets are assumed to be negligible.

1.3 Research Questions

Both ranked document retrieval and classification are affected by many sources of uncertainty. Ignoring uncertainty can lead to incorrect conclusions. In this thesis the following research questions are answered:

1. IR test collections use a set of retrieval results (runs) from the state of the art IR systems to create a sample or *pool* of documents to be judged [Spärck Jones and Van Rijsbergen, 1975;

Spärck Jones and Bates, 1977]. The quality of the final judgements produced for a collection is a byproduct of the variety within each of the runs used, and the pool depth. Non-pooled documents are assumed to be non-relevant. As a consequence, evaluation metrics computed for topics on a new IR system can be biased [Zobel, 1998; Carterette et al., 2010a; Sakai et al., 2012; Sakai, 2013].

(a) To reduce bias, the relevance of unjudged documents in retrieval results can be predicted using machine learning techniques based on the existing relevance judgements produced for the sample or pool [Büttcher et al., 2007; Carterette and Allan, 2007; Carterette et al., 2010a]. Each prediction is the probability of an unjudged document being relevant to a topic. In prior research, uncertainty as a result of probabilistic relevance judgements, and variances due to topics is captured in a confidence interval using methods incorporated into an evaluation metric. Further, the analysis is limited to binary relevance assessments. How can one accurately account for uncertainty independent of an evaluation metric when predictive approaches are used to discover relevant, but unjudged documents? Can one use predictive approaches to minimise bias in the context of incomplete graded relevance judgements?

(b) Manual retrieval runs, where humans intervene with the objective of finding relevant documents that are not found by automatic runs [Voorhees and Harman, 1998; Buckley et al., 2007] are pooled as a way of "future proofing" or reducing bias of a test collection [Soboroff and Robertson, 2003]. Can one construct reliable (low biased) IR test collections using only automatic retrieval runs (a smaller sample of runs produced with less effort)?

2. The use of sampling, randomized algorithms, or unpredictable inputs such as user feedback in IR often leads to *non-deterministic* outputs [Aly et al., 2013; Callan et al., 1999; Callan and Connell, 2001; Kulkarni and Callan, 2010; Si and Callan, 2003; 2004; 2005; Shokouhi, 2007; Thomas and Shokouhi, 2009; Finkelstein et al., 2001; Lawrence, 2000; Liu et al., 2004]. The effectiveness of such IR systems differs not only for different topics, but also for different instances of the system. Current IR evaluation techniques do not address this problem.

   (a) How can one capture two dimensional variance in effectiveness due to sampling of topics and sampling of IR system instances of a non-deterministic IR system in evaluation?

   (b) Sample based non-deterministic IR algorithms are often used to improve search efficiency [Kulkarni and Callan, 2010; Si and Callan, 2003; 2004; 2005; Shokouhi, 2007; Thomas and Shokouhi, 2009]. Lowering the sample rate increases the efficiency at the

cost of effectiveness. How can one determine the minimum sample rate in an effectiveness-efficiency tradeoff that achieves a similar effectiveness to the deterministic full index search?

3. Domain expertise is required to select a small set of predictive features for a classification problem. However, domain specific knowledge is scarce [Witten and Frank, 2005]. Therefore, problems with highly dynamic data, such as intrusion detectors utilise feature spaces automatically derived from a sample of training data [Curtsinger et al., 2011; Ma et al., 2009; Rieck et al., 2010]. However, automatically generated feature spaces can be massive. Such feature spaces have successfully been used in text classification problems, where a large volume of training data is available [Joachims, 1998].

   (a) Classifiers with large feature spaces can demonstrate a high variance in performance, especially when the training dataset is small [Hawkins, 2004; Marsland, 2009]. Caution must be used when evaluating potential high variance classifiers. What impact do feature spaces automatically generated using a sample of training data have on evaluating intrusion detector classifiers when labeled training data for malicious class is scarce? How can one reduce the variance of such classifiers?

   (b) Can the lessons learned from IR evaluation be applied to evaluate classifiers with uncertainty due to more than one level of variance? Classifiers sample data streams to improve the efficiency of classification [Aggarwal, 2007]. How can one find the minimum acceptable sample rate that achieves an acceptable effectiveness degradation in an effectiveness-efficiency tradeoff?

1.4 Overview

This thesis investigates evaluation with multiple sources of uncertainty (bias and variance) that primarily arise as a consequence of sampling. Chapter 2 reviews the fundamentals of evaluation in the context of adhoc IR, and classification. How two systems are compared for superiority, as well as for similarity are discussed. Further, how two test environments can be compared is reviewed. Finally, data mining algorithms used in the thesis are presented. Handling uncertainty due to test collection bias that arise as a consequence of producing relevance judgements only for a sample when evaluating IR experiments is discussed in Chapter 3. First, the suitability of predicting relevance of retrieved unjudged documents based on a sample of judged documents in the context of graded relevance

judgements is assessed. Second, using a smaller sample of runs produced with less effort to form a low bias IR test collection is investigated. Chapter 4 presents evaluation methodologies for problems with a two dimensional variance taking a case study from distributed IR. How effectiveness-efficiency tradeoffs are evaluated in the context of two dimensional variance is also illustrated. Here the focus is on evaluating non-deterministic IR systems for significant differences and similarity when topics and IR system instances are sampled from the respective populations. The impact of large feature spaces that are automatically generated using a sample of training data on evaluating classifiers is discussed in Chapter 5. For experiments, a case study on classifying *drive-by download* attacks is used as motivation. Here, uncertainty arise as a consequence of sampling features using a training dataset, and evaluating classifiers on varying training:test data samples. Demonstrating how unbounded automatically generated large feature spaces can lead to high variance classifiers, especially when training data is limited, importance of minimising variance for evaluation is stressed. How variance of such classifiers can be reduced is discussed. Further, how lessons learned in IR evaluation can be applied to non-deterministic classification problems is also analysed. Here evaluating non-deterministic classifier algorithms when training:test data partitions and repeated observations on each training:test data partition are sampled from respective populations is investigated. The thesis concludes in Chapter 6.

1.5 Contributions of the Thesis

The contributions of this thesis are: An evaluation methodology independent of the evaluation metric for comparing predictive approaches that address bias in IR test collections is proposed in Section 3.2.1; Using the proposed evaluation methodology, existing predictive approaches are compared in the context of graded relevance assessments in Section 3.2.4; A fully automated approach to generating a pool for judging documents in IR evaluation is proposed in Section 3.3. Two separate tests of significance for a non-deterministic:deterministic comparison of IR systems is proposed in Section 4.1.1, and Section 4.1.2. A significance test for non-deterministic:non-deterministic comparison of IR systems is proposed in Section 4.2. The properties of the proposed significance test are analysed using a case study from distributed IR in Section 4.4. An approach for evaluating the effectiveness-efficiency tradeoff when at least one algorithm is non-deterministic is presented in Section 4. The impact of automatic feature generation and lack of training data in the context of classifying drive-by download attacks is presented in Section 5.4. A novel effective and space efficient algorithm for detecting drive-by download attacks is proposed in Section 5.3. Finally, how lessons learned in IR evaluation can be applied for evaluation, and quantifying effectiveness-efficiency tradeoffs is

presented in Section 5.5.

Uncertainty from many sources is inevitable in experiments with complex requirements where data is sampled at many different levels to efficiently process big datasets and large data populations. Standard evaluation practices alone may not be sufficient for evaluating such experiments. The best practices and findings on how uncertainty is embraced rather than ignored in IR and classification experiments is the emphasis of this thesis.

Chapter 2

Background

The previous chapter briefly described the experimental process for IR systems and classifiers. How these systems are evaluated and the challenges faced were also explained. In this chapter the evaluation process is reviewed in detail. How effectiveness is quantified for IR systems and classifiers is presented in Section 2.1 and Section 2.2 respectively. Basic concepts of statistical analysis is discussed in Section 2.3. Stressing the importance of testing for significance, how two systems are compared for superiority is reviewed in Section 2.4. Comparing two systems for similar performance is discussed in Section 2.5. How two different test environments are evaluated is presented in Section 2.6. The data mining algorithms used to form classifiers in this thesis are discussed in Section 2.7. Evaluating IR systems and classifiers is focused next.

In a classic experiment, a new system is evaluated against a state of the art baseline. During evaluation the output from each system for each unit of input data, referred to as an *experimental unit* is recorded. The quality of the output (performance) for the experimental unit is quantified using an evaluation metric. Often the criteria of performance is effectiveness, but can be efficiency or any other aspect of the system. Next, how an IR system is evaluated for effectiveness is discussed.

2.1 Evaluating Effectiveness of IR Systems

For an IR system, an experimental unit is a topic, and the output is a ranked list of documents. Two distinct approaches have been used to quantify the effectiveness of ranked retrieval results. The first method uses a group of users to quantify the effectiveness for each topic in a set. The second approach is to use a standard test collection, with a corpus of documents, a set of topics, and relevance

assessments. Test collection based evaluation is less expensive. Therefore, most of the IR evaluations are on a test collection [Sanderson, 2010]. The relevance assessments specify which documents from the corpus are relevant for a given topic. Relevance assessments can be binary or graded. Quantifying the effectiveness of ranked retrieval results is not trivial, as different aspects of user satisfaction needs to be considered. Therefore, a large number of evaluation metrics that capture various aspects of user satisfaction have been designed.

2.1.1 Evaluation Metrics

Two concepts fundamental to the evaluation of IR systems are *precision* and *recall*. Precision is the fraction of retrieved relevant documents out of the total retrieved documents. The recall is the proportion of relevant documents that are retrieved. While precision measures the accuracy of results, recall measures the completeness. Recall is rather slippery as the total number relevant documents in a corpus is unknown. Therefore it is always computed relative to the known relevant documents in the corpus. Retrieving more documents boosts recall at the expense of precision. The opposite is also true. That is, retrieving less documents improves precision, but harms recall. So, these metrics are in tension with each other. Although, the two metrics are in tension enhancing the retrieval method improves the both, as now more relevant documents are found at the top of the ranking [Spärck Jones, 1981]. Therefore, a number of evaluation metrics combine recall and precision into a single score, for example, *average precision* (*AP*).

AP **and** MAP

AP is the mean over precision scores computed at each rank where a relevant document is retrieved. The above definition does not specify a cutoff. The standard practice is to evaluate up to a depth of 1000. AP rewards more for ranking relevant documents higher in the ranked order. Mean AP computed over a set of topics is called *mean average precision* (*MAP*).

R-precision

Another metric that combines recall and precision is R-precision. Instead of computing precision for all retrieved results, one could stop at a depth equal to the total number of relevant documents for each topic. Both AP and R-precision consider results to higher depths when a large number of

relevant documents are available for a topic. The next evaluation metric is based on the argument a user will examine retrieved results only down to a certain depth.

P@D

P@D is the precision at fixed depth D. Often used values for D are 10, 20, 30 and 100. This evaluation metric ignores recall. Methods discussed so far only take advantage of binary relevance assessments.

NDCG

Järvelin and Kekäläinen [2002] proposed *normalized discounted cumulative gain* (NDCG) to take advantage of graded relevance assessments. Here discounted cumulative gain to a depth D (DCG(D)) is computed as follows:

$$\mathsf{DCG(D)} = rel(1) + \sum_{i=2}^{\mathsf{D}} \frac{rel(i)}{\log_2(i)}, \tag{2.1}$$

where $rel(i)$ is the relevance of the document in the $i{\cdot}th$ rank position of retrieval results. Normalizing against an ideal ranking of relevant documents gives the NDCG score.

$$\mathsf{NDCG\ (D)} = \frac{\mathsf{DCG(D)}}{\mathsf{Ideal\_DCG(D)}} \tag{2.2}$$

Burges et al. [2005] proposed another version of NDCG that emphasizes higher ranked relevant results as follows:

$$\mathsf{DCG(D)} = \sum_{i=1}^{\mathsf{D}} \frac{2^{rel(i)} - 1}{\log(1+i)} \tag{2.3}$$

Many other evaluation metrics for graded relevance assessments, such as RBP [Moffat and Zobel, 2008], and Q-measure [Sakai, 2004a] have been proposed in recent years. See Sakai [2004b] for a comprehensive survey.

2.2 Evaluating Effectiveness of Classifiers

For a classifier system, a dataset takes the form of a set of examples where each example is expressed as a set of *features*, and a *label* denoting the class or the category. For evaluation, the dataset is randomly divided into at least two partitions [Witten and Frank, 2005]. One is referred to as the training data to train the classifier, and the other is known as the test data to test the classifier. Increasing

the size of the training dataset improves the classifier, but with small improvements after a certain volume of training data. Similarly, testing a classifier with additional test data improves the error estimation. In practice two thirds of the dataset is used for training as more training data tends to improve effectiveness, albeit with diminishing returns when the training dataset is large.

Some learning methods are trained in two-stages. In the first a learned concept is derived, and in the second the concept is optimised. In certain circumstances more than one classifier may need to be evaluated. In such situations the dataset is divided into three partitions: one to train, a second known as *validation data* to optimise the learned concept or select the most suitable classifier, and the last to test. However the data is partitioned, it is crucial that training, validation, and test datasets are disjoint. The above methods, known as *holdout* approaches are possible only when a large volume of labeled data is available. However, this is not the case for many classification problems.

A popular method for limited data is *10-fold cross validation*. Here, the dataset is randomly divided into 10-folds. Each fold is iteratively tested using a classifier trained on the remaining 9 folds. As every example in each fold in the dataset is tested, effectiveness can be assessed by computing an evaluation metric for all data, or for each fold which are averaged. Any number of folds can be used, but 10 is generally accepted as a good choice [Witten and Frank, 2005].

For very small datasets the *leave-one-out* method is used, which is a special case of many-fold cross validation. Here the number of folds is equal to the number of examples in the dataset. So, each example is tested using a classifier trained on all of the remaining examples. This method is computationally intensive for large datasets.

For most methods the data partitions can be *stratified*, by using the same proportion of examples from each class for each partition. This is good practice as it maintains similar training and test conditions, and avoids situations where all examples from a certain class are missing from the training data. Naturally, the leave-one-out method cannot be stratified.

A standard practice in classifier evaluation is to successively compute an evaluation metric for the dataset using one of the above methods. As examples from a dataset are randomly assigned to partitions, each repetition results in a different training-test dataset and a varying effectiveness. Here each repetition is considered an experimental unit. A $10\times$ stratified 10-fold cross validation is commonly used when data is limited [Bouckaert, 2003; 2004]. An alternative is to consider each fold of a many-fold cross validation as an experimental unit. Here, an appealing characteristic for evaluation is training-test data partitions are independent. However, reduced training data degrades the classifier effectiveness. So, this method is only suitable for large datasets.

		Predicted class: Positive	Predicted class: Negative
Actual class	Positive	True Positive (*TP*)	False Negative (*FN*)
	Negative	False Positive (*FP*)	True Negative (*TN*)

Table 2.1: A breakdown of a classifier predictions for a two class classification problem.

2.2.1 Evaluation Metrics

Evaluation of effectiveness for classifiers is done using standard confusion matrix metrics (See Table 2.1). The simplest to understand is the success rate. The success rate is the proportion of correctly classified examples out of total predictions:

$$\text{success rate} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{2.4}$$

The error rate is one minus the success rate. When the majority of examples in the dataset are from one class (unbalanced), a classifier always predicting the majority class will have a high success rate. Therefore, success rate is not a good evaluation metric for unbalanced datasets.

Precision and recall are two evaluation metrics also used in classification experiments. Both measures are redefined below in the context of classification using the standard confusion matrix metrics. Precision is the fraction of correctly identified positive samples out of the total positive predictions, where as recall, also called the true positive rate measures the proportion of positive cases that are correctly detected.

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2.5}$$

$$\text{recall}(\text{true positive rate}) = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2.6}$$

One may trade precision for recall, or vice versa. But, both can only be increased by improving the classifier, as now more instances are correctly predicted.

Another, evaluation metric is the false positive rate, which specifies the proportion of negative samples that are incorrectly classified as positive. This evaluation metric is important for problems where the cost of a false positive is high. In contrast to other evaluation metrics, a lower false positive rate is desirable.

$$\text{false positive rate} = \frac{\text{FP}}{\text{FP} + \text{TN}} \tag{2.7}$$

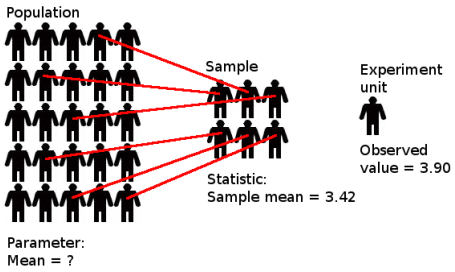


Figure 2.1: The model of statistical inference: Performance is observed for each experimental unit sampled from the population. Sample statistics, such as sample mean, and standard deviation are computed. Sample statistics are used to infer parameters, for example population mean.

Some measures combine precision and recall to a single value. Such measures are important when the application demands both accuracy and completeness. One such evaluation metric, called F-measure, is the harmonic mean of precision and recall.

$$\text{F-measure} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \tag{2.8}$$

How an experimental unit can be scored in IR and classifier system evaluations have been discussed. Next, the basic concepts essential for statistical analysis are reviewed.

2.3 Statistical Theory

A variable that can hold results, here performance scores for the *population* of experimental units is called a *random variable* (x). However, the population for system comparisons is not possible to construct. Hence, IR systems and classifiers are often evaluated using the performance observed on a random sample of experimental units. The sample results are summarised in order to compare two systems. A summarised attribute of a sample is called *sample statistic*. A sample statistic is an estimate for the corresponding parameter of the population. For example, the average performance ($\bar{\text{m}}$) computed on a sample output is a point estimate for the average performance on the population ($\mathcal{M}$). This model of statistical inference, where parameters for population are inferred using statistics computed on a random sample of experimental units drawn from the population is shown in Figure 2.1. Here the use of random sampling is crucial to infer population parameters from a sample.

A commonly used summary statistic for average is the sample *mean* $\bar{x}$, which is also the same as

the expectation $E(\cdot)$ of the random variable x.

$$\bar{x} = \frac{1}{|L|} \cdot \sum_{i=1}^{|L|} x_i \tag{2.9}$$

$$E(x) = \sum_{i=1}^{|L|} x_i \cdot p(x_i) \tag{2.10}$$

$$\bar{x} = E(x) \tag{2.11}$$

where $p(\cdot)$ is the *probability distribution* for x. So $p(x_i)$ is the probability of x having the value x_i. The mean is distorted by outliers or anomalies in data. The median is considered a better summary statistic for average when outliers are observed. The median is the value separating the lower and upper halves of sorted x. So the median is the $(|L|+1)/2{\cdot}th$ item for an odd number of experimental units and the mean of the $|L|/2{\cdot}th$ and $|L|/2+1{\cdot}th$ items for an even number of experimental units in the sorted x. Nonetheless, the mean is widely used for comparing IR systems and classifiers.

The mean indicates the central value, but does not show the scatter or distribution of values. For example, two random variables with different distributions can have a common mean. Hence, a better sense of the values in a random variable x can be obtained by defining the variance along with the mean. The sample variance computes the spread of values in the sample around its mean, and is given by:

$$var(x) = \hat{\sigma}^2 = \frac{1}{|L|-1} \cdot \sum_{i=1}^{|L|} (x_i - \bar{x})^2. \tag{2.12}$$

Here $\hat{\sigma}^2$ is used instead of σ^2 to indicate that the sample variance is an estimate for the population variance σ^2. Although, the variance is a useful concept, it is not comparable to mean as the unit of measure of the two quantities are different. Therefore, the *standard deviation*, denoted by $sd(x)$ or $\hat{\sigma}$, which is the square root of $var(x)$ is used instead.

The notion of a probability distribution for a random variable was introduced in this section. In addition certain theoretical probability distributions are central to the statistical data analysis. Each theoretical distribution defines what operations are permitted on data having a similar probability distribution. A few of these distributions are reviewed next.

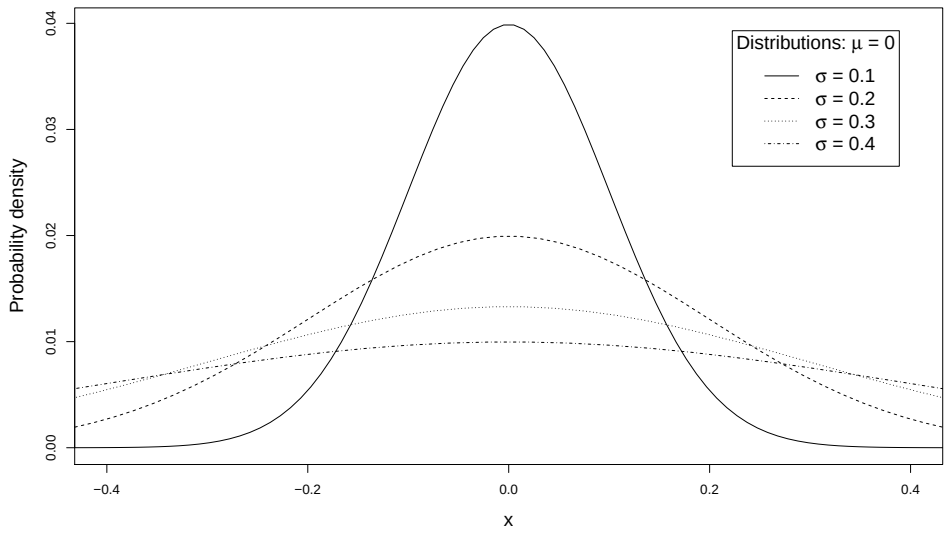


Figure 2.2: The impact of varying σ on normal distribution.

2.3.1 Theoretical Distributions

Among theoretical distributions the most widely used is the *normal distribution*. The normal distribution is modeled with the parameters mean (μ) and variance (σ^2) and is denoted by $\mathcal{N}(\mu, \sigma^2)$. A normal distribution takes the shape of a symmetric bell around the mean μ, whose width increase along with the remaining parameter σ as shown in Figure 2.2. One special normal distribution is the standard normal distribution of which $\mu = 0$ and $\sigma^2 = 1$.

Another theoretical distribution is the *binomial distribution*. The binomial distribution is used to model a random variable whose potential values are the binary outcomes of a trial, success or failure. Then the number of successes in a series of trials can be modeled using the binomial distribution of which the parameters are the probability of a success and the number of trials. Here each trial is assumed independent of the others.

Another widely used theoretical distribution is the uniform distribution whose values randomly vary within a given interval with equal probability. A number of other theoretical distributions such as the Poisson distribution, chi-squared distribution and the geometric distribution also exist. However, these distributions are not used in this thesis. Another distribution widely discussed in statistical theory is the *sampling distribution*.

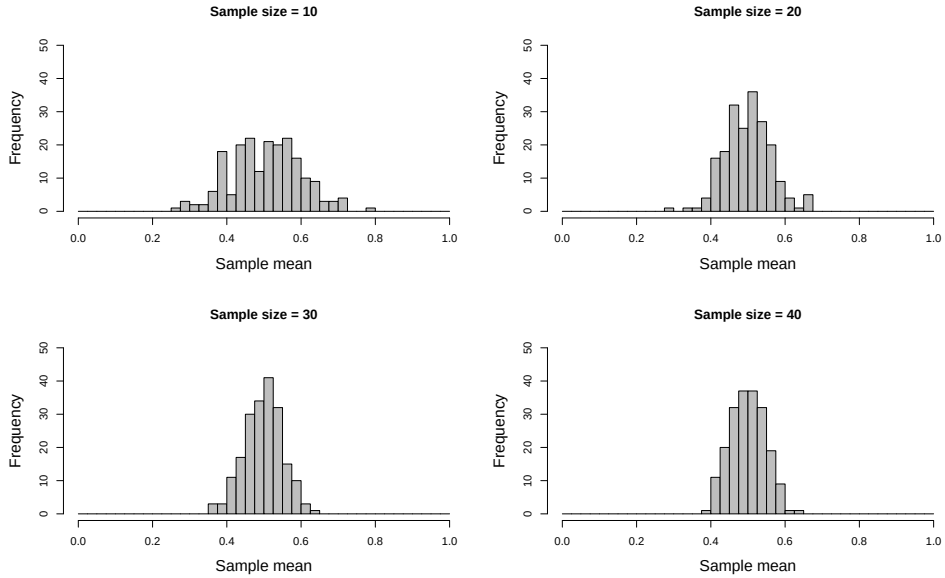


Figure 2.3: The distribution of sample means computed on different random samples.

2.3.2 Sampling Distribution

A sampling distribution is a probability distribution drawn for a statistic using repeated samples of the same size from the population. The probability statements about how close a sample statistic is to the population parameter can be made when the sampling distribution and its properties are known.

Among the theoretical distributions the most popular is the normal distribution. What's so important about the normal distribution? First, many datasets are normally distributed. Second, a large array of statistical tools are available when data is normally distributed. Recall that the mean is the most widely used statistic when evaluating IR systems and classifiers. The sampling distribution for the mean is considered normally distributed for large samples using a theorem known as the central limit theorem. The theorem is reviewed next.

2.3.3 Central Limit Theorem (CLT)

Theorem 2.3.1 *The distribution of the sample mean for a independent and identically distributed random samples of size $|L|$ with finite expectation μ and variance σ^2 is approximately normal in its distribution with expectation μ and variance $\sigma^2/|L|$ for large values of $|L|$ irrespective of the underlying distribution of the random samples.*

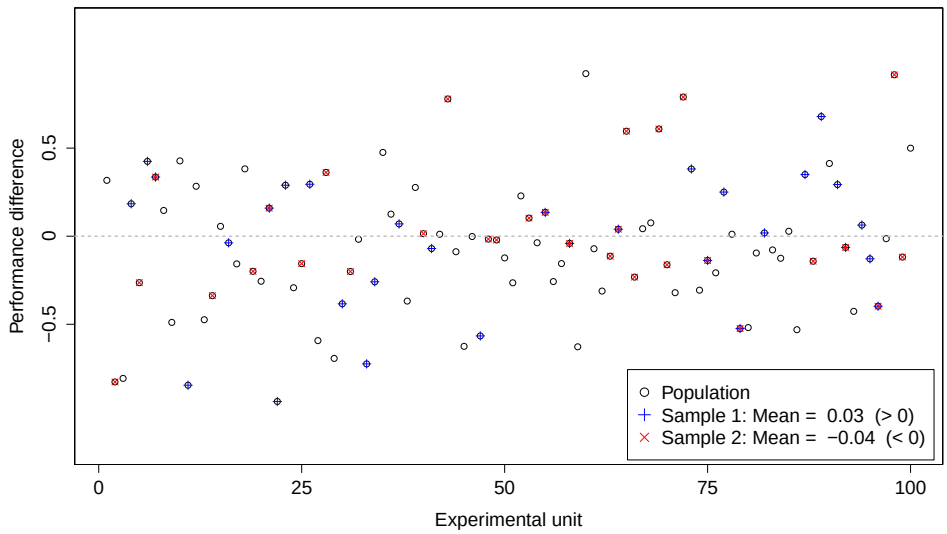


Figure 2.4: Comparing systems $\mathcal{A}$ and $\mathcal{B}$ using the mean of a sample. Each point is the performance difference for systems $\mathcal{A}$ and $\mathcal{B}$ for an experimental unit. $\mathcal{A}$ is better than $\mathcal{B}$, when evaluated on sample 1, and vice versa on sample 2.

The above theorem is illustrated in Figure 2.3 using a simple simulation. A limited population of 10000 items is generated as a random sample from a uniform distribution between 0 and 1. The histograms show the frequency distribution of 200 sample means computed using random samples when for each sample $10, 20, 30,$ and 40 items are drawn from the population. While the frequency distribution for mean approximates a normal distribution, variance reduces along with increasing sample sizes. Note the average performance computed on one random sample may differ from another. In the next section, how two systems are compared using a sample when the performance computed on one sample very from another is reviewed.

2.4 Evaluating Systems for Superiority

The varying estimates for average performance computed on different samples presents a dilemma: How can two systems accurately be compared? Envisage two systems $\mathcal{A}$ and $\mathcal{B}$, where performance for a simulated experimental unit is sampled from a uniform distribution between 0 and 1. Figure 2.4 illustrates the performance difference for a set of synthetic experimental units. The sample mean for system $\mathcal{A}$ is better than $\mathcal{B}$, when evaluated using sample 1, that is $\bar{\mathrm{m}}_{\mathcal{A}} > \bar{\mathrm{m}}_{\mathcal{B}}$. Similarly, the sample

mean for system $\mathcal{A}$ is worse than system $\mathcal{B}$, when evaluated using the sample 2, that is $\bar{\mathrm{m}}_{\mathcal{A}} < \bar{\mathrm{m}}_{\mathcal{B}}$. Sometimes sample statistics are equivalent, that is $\bar{\mathrm{m}}_{\mathcal{A}} = \bar{\mathrm{m}}_{\mathcal{B}}$. This exemplifies that for two equivalent systems $\mathcal{A}$ and $\mathcal{B}$, one may conclude $\mathcal{A}$ is better, equivalent, or worse than $\mathcal{B}$ depending on the sample. Hence, point estimates for average performance derived from a single sample is not a reliable indicator of performance when comparing two systems as the observed difference in average performance is specific to the sample chosen. Fortunately, a hypothesis test [Fisher, 1934], or interval estimates for average performance (confidence interval) that take variance in average performance into account can be used to improve the reliability of comparisons.

2.4.1 Hypothesis Testing

Let $\bar{\mathrm{m}}_{\mathcal{A}}$ and $\bar{\mathrm{m}}_{\mathcal{B}}$ be the average performance scores for systems $\mathcal{A}$ and $\mathcal{B}$ on two random samples of experimental units $L_{\mathcal{A}}$ and $L_{\mathcal{B}}$. Given $\bar{\mathrm{m}}_{\mathcal{A}} > \bar{\mathrm{m}}_{\mathcal{B}}$, a hypothesis test questions how confident one can be that a similar behaviour is observed on the population, that is $\mathcal{M}_{\mathcal{A}} > \mathcal{M}_{\mathcal{B}}$. The above question is answered via two competing hypotheses: one, referred to as the *null hypothesis* or H_0, assumes the performance scores for $\mathcal{A}$ and $\mathcal{B}$ on the population of experimental units are equal; and the other, called the *alternative hypothesis* or H_1 assumes they are not equal.

$$H_0 : \mathcal{M}_{\mathcal{A}} = \mathcal{M}_{\mathcal{B}} \tag{2.13}$$

$$H_1 : \mathcal{M}_{\mathcal{A}} \neq \mathcal{M}_{\mathcal{B}} \tag{2.14}$$

A hypothesis test estimates the probability of satisfying the null hypothesis using evidence gathered from a sample of data. Only substantial evidence against the null hypothesis is considered as not supporting it. The probability of two systems being hypothetically equal on the population is called the p *value* of the test [Fisher, 1934]. A p value below a commonly accepted *level of significance* (α) implies a systematic difference that cannot be attributed purely to the chance in selection of experimental units – that is, the alternative hypothesis is accepted rejecting the null hypothesis. Vice versa, with a large p value one fail to reject the null hypothesis, hence any difference in average performance cannot be confidently confirmed, and therefore is not statistically significant. The way a conclusion is derived from a hypothesis test makes an analogy to the judicial process in the USA and many other countries [Feinberg, 1971]. A defendant is considered innocent until proven guilty beyond a reasonable doubt using the evidence established against the defendant. The values 0.1, 0.05, and 0.01 are commonly used levels of significance. A number of alternative tests are available to compute the p value for a hypothesis test.

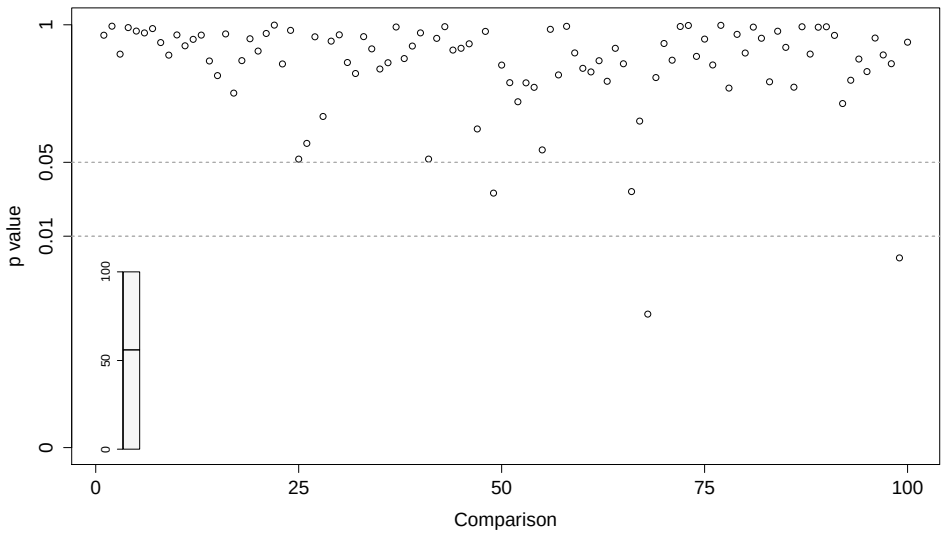


Figure 2.5: Evaluating two equal systems $\mathcal{A}$ and $\mathcal{B}$ for mean performance on 100 random samples using a hypothesis test. The barplot within shows the proportion of conclusions ($\mathcal{A}$ is better than $\mathcal{B}$ and vice versa) that were derived when sample means were compared.

The above hypothesis test is a *two-tailed* test, where evidence from both directions are considered. Another is *one-tailed* test, where only evidence in one direction is analysed. For a one-tailed test:

$$H_0 : \mathcal{M}_{\mathcal{A}} \leq \mathcal{M}_{\mathcal{B}} \tag{2.15}$$

$$H_1 : \mathcal{M}_{\mathcal{A}} > \mathcal{M}_{\mathcal{B}} \tag{2.16}$$

Some argue a one-tailed test is more natural and suitable for system comparisons [van Rijsbergen, 1997; Croft et al., 2010]. However, it should be noted that a two-tailed test is more stringent than a one-tailed test. In fact, the p value for a two-tailed test is twice as that of for a one-tailed test for a symmetrical distribution. A two-tailed test is more widely used than a one-tailed test in most comparisons today.

Reconsider previous simulation where two equivalent systems $\mathcal{A}$ and $\mathcal{B}$ are compared on a random sample of simulated experimental units. Performance for $\mathcal{A}$ and $\mathcal{B}$ is sampled from a uniform distribution between 0 and 1 for each experimental unit. The earlier example showed that comparing sample means is not always reliable. Therefore, a hypothesis test is used for this comparison. Results for many such comparisons, are shown in Figure 2.5. The bar plot within illustrates results for comparing sample means. System $\mathcal{A}$ is better than system $\mathcal{B}$ and the opposite is true on many

samples when comparing sample means. No equal sample means are observed in this experiment as performance scores are real valued. However, with hypothesis testing only a small number of samples have a p value less than 0.05. This number is even smaller for a p value less than 0.01. The majority of comparisons fail to reject the null hypothesis. Hence, on many samples no statistically significant differences are observed between systems $\mathcal{A}$ and $\mathcal{B}$. Unfortunately, the accurate conclusion is not always derived. Occasionally a sample for which one system consistently beats the other is observed, and two systems appear to be statistically significantly different even though the systems are equivalent. These are the samples for which the p value is extremely low ($p < \alpha$). Such errors where the null hypothesis is rejected when in fact it is true are called *type I errors*. A hypothesis test attempts to minimise type I errors. The smaller the value of α the lower the probability of making a type I error. Another class of errors occur when the test fails to reject null hypothesis when it is false. Such errors are called *type II errors*. The rate of type II error is depicted with β. Here the probability of rejecting null hypothesis or the *power of a test* is defined as $1 - \beta$. Type I errors and type II errors are in tension. Hence, one can reduce Type I errors at the expense of power, and this is what happens when α is lowered.

How important is a result showing a significant improvement is determined by the *effect size*. Instead of considering the average performance difference, one may use Cohen's d statistic which is the performance score difference normalized by the standard deviation to quantify the effect size [Cohen, 2013].

$$\text{effect size} = \frac{\bar{\mathrm{m}}_{\mathcal{A}} - \bar{\mathrm{m}}_{\mathcal{B}}}{\hat{\sigma}} \tag{2.17}$$

Here $\hat{\sigma}$ is the standard deviation for performance score differences. Common rules of thumb when classifying results based on effect size are: 0.8 and above is a large effect, from 0.5 to 0.8 is a medium effect, and between 0.2 and 0.5 a small effect.

2.4.2 Confidence Interval

Point estimates for a population parameter derived across samples vary. Therefore, a population parameter is inferred as an interval estimate. However, an interval estimate from a sample cannot be derived with certainty. Therefore, a confidence interval is defined for a given *level of confidence* $(1 - \alpha)\%$. How frequently the population parameter is included in the defined interval is expressed by the confidence level.

Two systems are compared by drawing a confidence interval for the performance differences. The expectation of performance difference is 0 for two similar systems. Therefore, if 0 is not within the

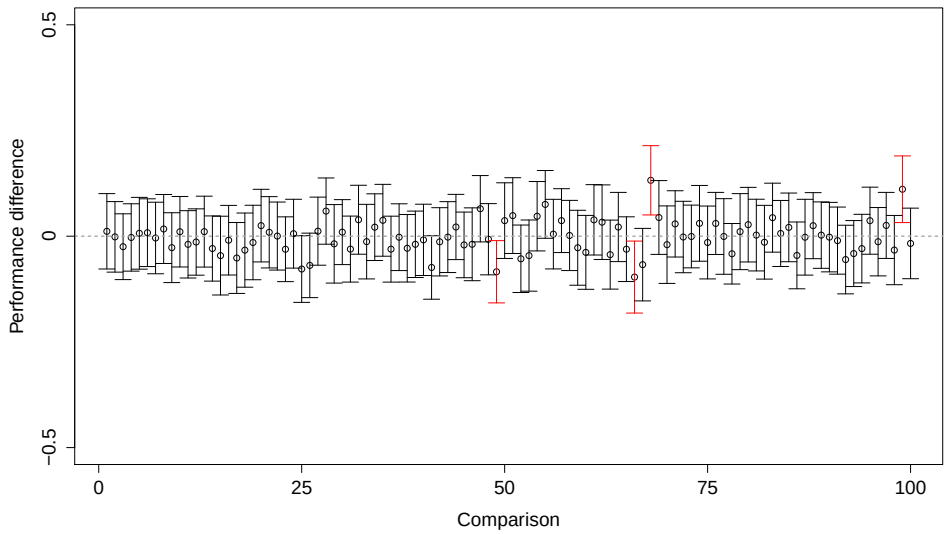


Figure 2.6: Evaluating two equal systems $\mathcal{A}$ and $\mathcal{B}$ for mean performance on random samples using confidence intervals. The plot shows results for the same 100 *samples compared using a significance test for the data points shown in Figure 2.5.*

confidence interval the systems are significantly different at the given level of confidence $(1-\alpha)\%$. The commonly used levels of confidence include 99%, 95%, and 90%. Higher level of confidence implies a more accurate comparison.

Figure 2.6 shows the corresponding 95% confidence intervals for the mean performance difference computed on the same simulated samples used to compare the equivalent systems $\mathcal{A}$ and $\mathcal{B}$ using a *significance test* in Figure 2.5. When 0 is not included within the confidence interval, the two systems are incorrectly identified as being significantly different. Note the similarity between conclusions derived using both approaches at an α of 0.05.

The way the p value is derived or the confidence interval is computed differs from one approach to another, but the principle is similar. First, the sampling distribution for a statistic of interest under the null hypothesis that is for a two equivalent systems is determined. The probability that a sample statistic is drawn from the sampling distribution can be determined when the sampling distribution and its properties are known. For a hypothesis test the p value is the probability of observing a *test statistic* at least as extreme as the one that was observed on the sample. In practice only a sample is available to determine the sampling distribution. Therefore, the sampling distribution is derived via theoretical or resampling techniques. Each test computes the test statistic and the sampling dis-

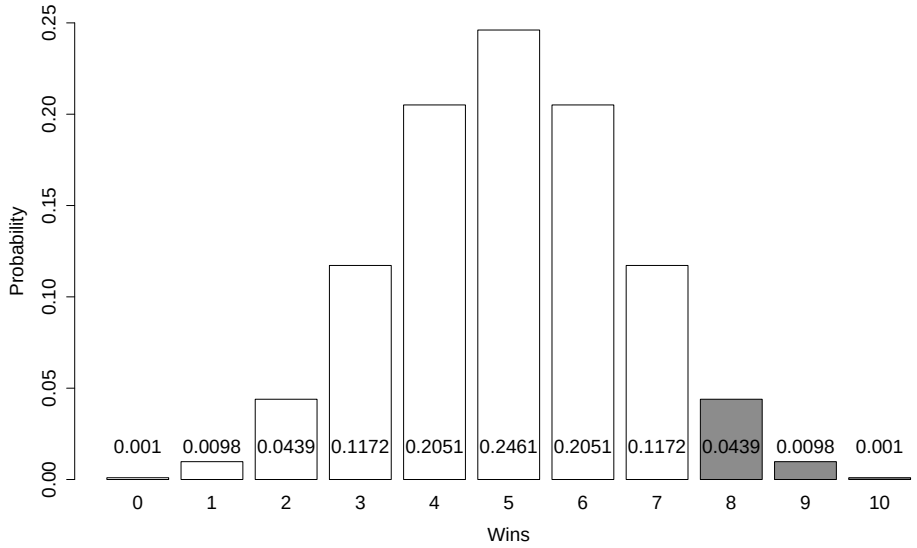


Figure 2.7: Binomial probabilities for 10 *trials when the expectation of a win in a trial is* 0.5.

tribution in a different manner. Widely used approaches for comparing IR systems and classifiers are discussed next. A more reliable comparison is possible when both systems are evaluated on the same sample of experimental units ($L_{\mathcal{A}} = L_{\mathcal{B}}$) [Witten and Frank, 2005], and is the widely used standard for comparing IR systems and classifiers [Smucker et al., 2007; Japkowicz and Shah, 2011]. Therefore, approaches for comparing systems on independent samples are not discussed here.

2.4.3 Comparing on a Matched Sample

Let a_i and b_i be the respective performance scores on systems $\mathcal{A}$ and $\mathcal{B}$ for the $i{\cdot}th$ experimental unit from a sample L, and z_i be the difference in performance for matched output pairs.

$$z_i = a_i - b_i \tag{2.18}$$

Consider $\mathcal{A}$ is the new system, and $\mathcal{B}$ is the baseline. Commonly used matched sample significance tests include: the sign test, the Wilcoxon signed-rank test [Wilcoxon, 1945], the paired t-*test* [Student, 1908; Fisher, 1934], the bootstrap test [Efron, 1979; 1982; Efron and Tibshirani, 1993], Fisher's randomization test [Fisher et al., 1935], and the multivariate linear model test [Carterette, 2011; Carterette et al., 2011; Robertson and Kanoulas, 2012].

Sign Test

The sign test is the simplest among all tests for significance, but provides an elegant illustration of the workings associated with a hypothesis test. The sign test compares two systems for median performance by estimating the number of times one system wins over another. Therefore the performance difference for each experimental unit is considered a trial whose outcome is a win if $z_i > 0$ and a loss if $z_i < 0$. A tie can be dealt with in two different ways. One simple option is to eliminate them, and consider only the reduced sample [Anderson et al., 2008]. This is the option most widely used. The other is to split ties equally between the counts for wins and losses, except when the number of ties are an odd number, in which case one tie is ignored [Japkowicz and Shah, 2011]. In IR and classification experiments real valued performance scores (not rounded off) can be used to minimise ties [van Rijsbergen, 1997]. Each individual fold of a multi-fold cross validation is considered an experimental unit when evaluating classifiers using the sign test [Japkowicz and Shah, 2011]. The expectation of a win in a trial is 0.5 when the null hypothesis is true. The number of wins provides the test statistic, which follows a binomial distribution as the sampling distribution with a $p = 0.5$ and a number of trials equal to the reduced sample size. The sign test estimates the p value as the likelihood of observing a number of wins at least as extreme as the actual number of wins observed for the given number of trials. For example, assume 8 wins are observed from 10 trials. Now the p value for a one-tailed test is the probability of observing at least 8 wins. The binomial probabilities for a given number of wins in 10 trials are shown in Figure 2.7. For example, the probability of observing 8 wins is 0.0439. The probability of observing at least 8 wins is $0.0439 + 0.0098 + 0.001 = 0.0547$. The result is significant when an α of 0.1 is considered. However, the p value is twice that of a one-tailed test for a two-tailed test as now probability of observing at most 3 wins are also considered. Now the p value is 0.1094, hence the result is not significant at an α of 0.1. The sign test is simple and easy to compute, but ignores magnitude and variability in performance differences, which can be addressed with a Wilcoxon signed-rank test.

Wilcoxon Signed-rank Test

The Wilcoxon signed-rank test also compares two systems for median performance. First, all ties ($z_i = 0$) are dropped. Then the experimental units are ranked by the absolute value of performance difference in ascending order. For instance, the experimental unit with the smallest absolute value of performance difference is ranked 1. Experimental units with a tying rank are assigned an average

rank. Each rank is assigned the sign of the original performance difference. The sum of positive signed-ranks gives the test statistic (Γ^+_{wilcox}) for the test. When the number of experimental units in the reduced sample is 10 or more Γ^+_{wilcox} follows a normal distribution with:

$$\text{mean } \mu_{\Gamma^+_{wilcox}} = \frac{|L|(|L|+1)}{4}, \text{ and}$$
$$\text{standard deviation } \sigma_{\Gamma^+_{wilcox}} = \sqrt{\frac{|L|(|L|+1)(2|L|+1)}{24}}.$$

The p value corresponding to the

$$\text{z score} = \frac{\Gamma^+_{wilcox} - \mu_{\Gamma^+_{wilcox}}}{\sigma_{\Gamma^+_{wilcox}}}$$

is computed from the standard normal distribution. Here $|L|$ is the size of the reduced sample. The p value for a two-tailed test is twice of that of the p value for a one-tailed test. Both the sign test and the Wilcoxon signed-rank test are easy to compute. However the test only compares for the median and not the mean performance.

Paired t-test

For the sign test and the Wilcoxon signed-rank test no parameters for the sampling distribution are estimated. Alternatively, the sampling distribution can be determined with estimated parameters and is the basis for the most widely used paired t-test. The idea here is to compare the mean performance score difference between two systems against per unit variance. The test computes a *test statistic* Γ as follows:

$$\Gamma(z) = \frac{\overline{z}}{\sqrt{\hat{\sigma}(z)^2/|L|}} \tag{2.19}$$

Here $\overline{z}$ is the mean performance score difference between systems $\mathcal{A}$ and $\mathcal{B}$ on a sample L, $|L|$ is the size of the sample, and $\hat{\sigma}(z)$ is the sample standard deviation. The sampling distribution for the mean is a normal distribution. However, sample statistics (mean and standard deviation) are used here as the best estimates for the population parameters when deriving the sampling distribution. Therefore, Γ follows a *t distribution*, which accounts for more uncertainty than a normal distribution as illustrated in Figure 2.8. The t distribution approaches a normal distribution with increasing sizes of the sample. The p value corresponding to the test statistic and the degrees of freedom ($|L|-1$) is

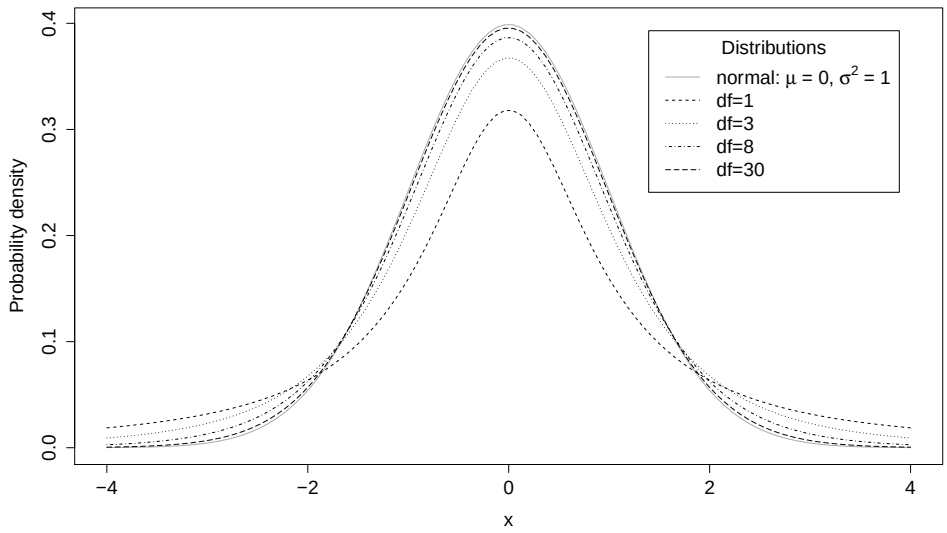


Figure 2.8: The normal distribution, and t-distributions for varying degrees of freedom (df).

obtained from the t distribution. Again due to the symmetrical sampling distribution the p value for a two-tailed test is twice as that for a one-tailed test.

The same conclusions are derived with a confidence interval for mean performance score difference and estimated as follows:

$$\overline{z} \pm t_{\alpha/2}\sqrt{\frac{\hat{\sigma}^2}{|L|}} \tag{2.20}$$

Here $t_{\alpha/2}$ is the t value corresponding to an area of $\alpha/2$ in the upper tail of the t distribution with $|L| - 1$ degrees of freedom. Note the similarity between computing the t-statistic and the confidence interval.

Recall that the sampling distribution for the above test is derived using theoretical methods and estimated parameters, hence a number of parametric assumptions about the distribution must be met. First, the test requires the two populations being compared to be normally distributed, and have a similar variance. Furthermore, the test expects sampled experimental units to be independent. However, a paired t-test and the associated confidence interval method were found to be robust to violations of normality assumptions for large samples which is not surprising given the central limit theorem [Anderson et al., 2008; Hull, 1993; Smucker et al., 2007; 2009].

Although, experimental units for IR system evaluation (topics) can assumed to be independent, repeated runs with cross validation for classifier evaluations are not. Overlapping training sets in

repeated cross validations result in an underestimation of variance, and a higher likelihood of type-I errors. The problem is exacerbated when increasing the number of repeated cross validations. If each individual fold in a multi-fold cross validation is considered an experimental unit, they are independent. However, this reduces the training dataset for each experimental unit resulting in an underestimation of effectiveness for classifiers on small datasets. Bouckaert [2003; 2004] recommended the use of $10\times$ 10-fold cross validation for small datasets, and is now the de facto standard for evaluations on small datasets with t-tests. Here each 10-fold cross validation is an experimental unit.

An alternative to build sampling distribution is to use simulations. Here, the intuition is to empirically estimate the unknown sampling distribution via repeated sample variations formed using the sample. Therefore these methods make fewer assumptions about the sampling distribution. Further, these methods are not tied to inference of a particular parameter, and therefore can be used to compare systems for mean or median performance. However, the methods are computationally expensive. Two such methods, the Randomization test and the Bootstrap test are discussed next.

Randomization Test

If systems $\mathcal{A}$ and $\mathcal{B}$ are similar as stated for the null hypothesis, another third system can also be assumed to be similar to both systems $\mathcal{A}$ and $\mathcal{B}$. Hence, for the $i{\cdot}th$ experimental unit one may observe a pair of effectivenesses a_i and b_i from the third system. As all three systems are similar under the null hypothesis, either effectiveness could have come from system $\mathcal{A}$ or system $\mathcal{B}$. Therefore, one of these effectivenesses is arbitrarily assigned to system $\mathcal{A}$, and the other to system $\mathcal{B}$ as produced by the respective systems. For $|L|$ experimental units the third system can produce $2^{|L|}$ such permutations. Let $\bar{\mathfrak{m}}^j_{\mathcal{A}}$ and $\bar{\mathfrak{m}}^j_{\mathcal{B}}$ be the mean computed for systems $\mathcal{A}$ and $\mathcal{B}$ on $j{\cdot}th$ permutation, where $j = 1, ..., 2^{|L|}$. Similarly let $\bar{\mathfrak{m}}_{\mathcal{A}}$ and $\bar{\mathfrak{m}}_{\mathcal{B}}$ be the mean computed for the original assessment of systems $\mathcal{A}$ and $\mathcal{B}$. The number of times the difference $\bar{\mathfrak{m}}^j_{\mathcal{A}} - \bar{\mathfrak{m}}^j_{\mathcal{B}}$ is greater than or equal to the difference $\bar{\mathfrak{m}}_{\mathcal{A}} - \bar{\mathfrak{m}}_{\mathcal{B}}$ per comparison is the p value for one-tailed randomization test.

$$p \text{ value}_{\text{one-tailed}} = \frac{count[(\bar{\mathfrak{m}}^j_{\mathcal{A}} - \bar{\mathfrak{m}}^j_{\mathcal{B}}) \geq (\bar{\mathfrak{m}}_{\mathcal{A}} - \bar{\mathfrak{m}}_{\mathcal{B}})]}{2^{|L|}} \tag{2.21}$$

Similarly, the p value for the two-tailed test is computed by comparing the absolute value of the

performance differences.

$$p \text{ value}_{\text{two-tailed}} = \frac{count[abs(\bar{\mathrm{m}}^j_{\mathcal{A}} - \bar{\mathrm{m}}^j_{\mathcal{B}}) \geq abs(\bar{\mathrm{m}}_{\mathcal{A}} - \bar{\mathrm{m}}_{\mathcal{B}})]}{2^{|L|}} \tag{2.22}$$

However, computing all $2^{|L|}$ permutations is computationally expensive. Therefore, the p value is often computed using a random sample from the $2^{|L|}$ permutations. The method presented above is a variant of the randomization test used by Smucker et al. [2007] to compare IR systems. Instead of the mean performance difference the *t-statistic* can be compared as illustrated with the bootstrap method next.

Bootstrap Test

Bootstrap methods are widely used to evaluate IR systems [Wilbur, 1994; Savoy, 1997; Sakai, 2006], and classifiers [Chernick et al., 1985; Japkowicz and Shah, 2011]. The test uses resampling to derive the sampling distribution under a null hypothesis. A bootstrap approach for evaluating two systems for mean performance is presented in Algorithm 1. The test creates BS bootstrap samples of size $|L|$ by repeatedly resampling with replacement. Performance for each experimental unit in the bootstrapped samples are recentered by deducting the mean deviation of the mean performance for bootstrap samples from 0. So the recentered bootstrap samples are assumed to be from a pair of equivalent systems or the sampling distribution under the null hypothesis. Now the evidence to support a null hypothesis is observed by comparing each individual recentered bootstrap sample with the original sample. Let $t(z)$ and $t(z^*_j)$ be the t-statistic for comparing the original sample z with the $j{\cdot}th$ recentered bootstrap sample z^*_j. Evidence in favour of null hypothesis for a one-tailed test is found each time the value of the t-statistic for the $j{\cdot}th$ recentered bootstrap sample $t(z^*_j)$ is greater than or equal to the value of the t-statistic for the original sample. Similar to the randomization test evidence in favour of null hypothesis for a two-tailed test is found when absolute values are used in each comparison. Hence, the p values for the test are given by:

$$p \text{ value}_{\text{one-tailed}} = \frac{count[t(z^*) \geq t(z)]}{BS}$$
$$p \text{ value}_{\text{two-tailed}} = \frac{count[abs(t(z^*)) \geq abs(t(z))]}{BS}$$

The confidence interval for the mean or median can also be computed using the bootstrap method. A bootstrap approach for estimating the confidence interval for the population mean is presented in Algorithm 2. The $\alpha/2$ and $1 - \alpha/2$ percentiles of the sorted bootstrap sample means are the lower

ALGORITHM 1 Bootstrap algorithm for comparing two systems $\mathcal{A}$ and $\mathcal{B}$.

$BS \leftarrow$ Number of bootstrap samples;
$|L| \leftarrow$ Number of experimental units;
$a_i \leftarrow$ Performance score for system $\mathcal{A}$ on experimental unit x_i;
$b_i \leftarrow$ Performance score for system $\mathcal{B}$ on experimental unit x_i;

// Compute $t(z)$.
$[z_i] \leftarrow [a_i - b_i], \forall i = 1, ..., |L|$
$t(z) \leftarrow \overline{z}/\sqrt{\hat{\sigma}_z^2/|L|}$;

// Compute p value.
$count \leftarrow 0$;
$bootstrap\_samples \leftarrow []$;
$total\_shift \leftarrow 0$;
for $j = 1$ *to* BS **do**
    $z_j^* \leftarrow$ Resample $|L|$ items with replacement from z;
    $total\_shift \leftarrow total\_shift + mean(z_j^*)$;
    $bootstrap\_samples.append(z_j^*)$;
end
$shift \leftarrow total\_shift/BS$;
for $j = 1$ *to* BS **do**
    $z_j^* \leftarrow bootstrap\_samples[j]$;
    for $i = 1$ *to* $|L|$ **do**
        $z_j^*[i] \leftarrow z_j^*[i] - shift$;
    end
    $t(z_j^*) \leftarrow \overline{z_j^*}/\sqrt{\hat{\sigma}_{z_j^*}^2/|L|}$;
    if $abs(t(z_j^*)) \geq abs(t(z))$ **then**
        $count \leftarrow count + 1$;
    end
end
$p\_value \leftarrow count/BS$;

and upper bounds for a confidence interval with $(1 - \alpha)$ levels of confidence.

All of the tests so far can handle only one variation, that is the variation due to experimental units. How can a system be evaluated if performance observed for an experimental unit varies with each repeated measurement? This question can be answered using multivariate linear model tests. The discussion commences by illustrating the multivariate linear model test for the standard case.

ALGORITHM 2 Bootstrap algorithm for estimating confidence interval for population mean.

$BS \leftarrow$ Number of bootstrap samples;
$|L| \leftarrow$ Number of experimental units;
$a_i \leftarrow$ Performance score for system $\mathcal{A}$ on experimental unit x_i;
$b_i \leftarrow$ Performance score for system $\mathcal{B}$ on experimental unit x_i;

// Compute $t(z)$.
$[z_i] \leftarrow [a_i - b_i], \forall i = 1, ..., |L|$

// Construct bootstrap samples using resampling with replacement.
$bootstrap\_sample\_means \leftarrow []$;
for $j = 1$ *to* BS **do**
    $z_j^* \leftarrow$ Resample $|L|$ items with replacement from z;
    $bootstrap\_samples\_means.append(mean(z_j^*))$;
end
$sort(bootstrap\_sample\_means)$;
$confidence\_interval \leftarrow \alpha/2$ and $1 - \alpha/2$ percentiles of $bootstrap\_sample\_means$;

Multivariate Linear Model Test

When the variance is due solely to the experimental units, the test applies the following multivariate linear regression model.

$$y_{ij} = \gamma + S_i + x_j + \varepsilon_{ij}, \tag{2.23}$$

Let y_{ij} be the performance modeled by two factors, also known as effects: the $i{\cdot}th$ system effect S_i, and the effect due to $j{\cdot}th$ experimental unit x_j. Let one of the systems and one of the experimental units respectively be the *reference levels* for the system effect and experimental unit effect. Now, the system effect S_i is the effect of the $i{\cdot}th$ system relative to the reference system, and x_j is the easiness or the difficulty of $j{\cdot}th$ experimental unit compared to the reference experimental unit. The intercept γ in the model is the performance of the reference system for the reference experimental unit, as explained by the model. The discrepancy between the observed performance and that explained by the model (*residual* or *error*) resides in ε_{ij}. The p value for the system effect can either be computed with $\Delta_S / \sqrt{\hat{\sigma}^2 / |L|}$ or a t-statistic using ANOVA, which follows Student's t distribution with $|L| - 1$ degrees of freedom. Here Δ_S is the difference between the two system effects, $\hat{\sigma}^2$ is the variance of the residuals, and $|L|$ is the size of the sample.

Alternatively, the same can be modeled using a *linear mixed effect (*LME*)* model [Stroup, 2012], consisting of fixed (non-random) and random effects [Carterette et al., 2011; Robertson and Kanoulas, 2012]. For this scenario, sampled experimental units produce a random effect, and systems produce a

fixed effect. Though not apparent at this stage LME can be used to build complex models that capture repeated measurements and hierarchical groupings. For large samples, the p value can be computed from a t-statistic obtained from the LME and the degrees of freedom of sample size less the number of fixed effect parameters ($|L| - 1$ for the above scenario) [Baayen et al., 2008].

Instead, simulation techniques can be applied on the LME model using *Bayesian inference* and *Markov chain Monte Carlo (*MCMC*)* sampling to compute the p value [Baayen et al., 2008]. MCMC sampling is derived from the posterior probability distribution of the parameter subsets of the LME model (σ, parameters defining the variance-covariance for random effects, fixed effects, and random effects) as a random traversal of the parameter space [Andrieu et al., 2003]. Here, posterior probability distributions are derived with repeated and cyclic sampling from each parameter subset conditioned on other two parameter subsets and on the data, thus making variance of all other parameter subsets reflect the variance for each parameter subset and data. Use of a Markov chain forces the sampling to spend more effort on important regions of the posterior probability distribution [Andrieu et al., 2003]. The posterior probability distribution for the system effect is expected to follow a near symmetric normal distribution from which p value is derived. Relaxed assumptions about the distribution of population is an advantage of using Bayesian inference and MCMC to compute the p value.

The posterior probability distribution for the system effect can be used to derive the highest posterior density (HPD) interval which is analogous to the standard confidence interval. A $(1-\alpha)\ \%$ HPD interval represents the shortest interval enclosing $(1-\alpha)\ \%$ of the posterior probability mass of the distribution. Therefore, the HPD interval is considered a better estimate of a population parameter than a standard confidence interval [Chen and Shao, 1999].

With repeated observations the variance can be thought of as combining two components: first, measurement or model error; and second, the fact that some systems do better on some experimental units than others (both systems and experimental units). One cannot separate these two factors with only one observation per experimental unit, but the two factors can be separately estimated with repeated observations. Taking system effect as fixed, and experimental unit and system–experimental unit interaction effects as random, the above can be modeled with LME, as follows:

$$y_{ijo} = \gamma + S_i + x_j + Sx_{ij} + \varepsilon_{ijo} \tag{2.24}$$

Here y_{ijo} is the effectiveness on the $o{\cdot}th$ observation of system i on topic j, Sx_{ij} is the system–experimental unit interaction effect, and ε_{ijo} represents the (random) error of a single observation.

However, the model only makes sense if different observations on the same system–experimental unit interaction pair lead to different scores. In Robertson and Kanoulas [2012], this variability in topic-system scores is observed over different document sets, whereas in Carterette et al. [2011] the variability is in different user types.

2.4.4 Achieving a Significant Difference

What can a researcher do if an experiment strongly believed to improve the baseline turns out not be significant? This can be answered easily using the formulation for a paired t-test. A higher t-statistic is required to reduce the p value for the test. Although, a higher mean difference, a lower variance, or a higher sample size can be used to reduce the p value, only the sample size is under the researcher's direct control. Therefore, researchers must increase the sample size to show a significant difference. Here how two systems are evaluated for a difference in performance was discussed. Showing two systems are not different is not the same as two systems being equal when inferred from a sample. So, how two systems can be compared for a similar performance is reviewed next.

2.5 Evaluating Systems for Similarity

The exact identity of two systems can only be established after seeing the population. Therefore, assessing similarity of two systems using the performance observed for a sample of experimental units is not as straightforward as it might initially seem. Even if two systems are believed to be equal on average, the sample averages may not be equal, as was observed in Figure 2.4. On the other hand equal sample averages may come from two non-similar systems.

An unconventional and incorrect method to prove similarity between two systems $\mathcal{A}$ and $\mathcal{B}$ is to setup the following experiment. Performance scores for the two systems are observed on a sample of experimental units and tested for a significant difference. The test finds no significant difference at the highest level of significance ($\alpha = 0.1$). Based on the result, the researcher accepts the null hypothesis ($\mathcal{M}_{\mathcal{A}} = \mathcal{M}_{\mathcal{B}}$). In reality, the null hypothesis is a statistical straw man. Even when two systems are identical, using the failure of a statistical significance test to reject the null hypothesis as a proof for equivalence is bad practice. The way forward is obvious for a researcher willing to find no significant difference between two systems in order to justify an experimental objective. Reduce the sample size so that you minimise the likelihood of finding a significant difference. In this section, while clarifying

Practice	Null hypothesis (H_0)	Desired outcome	Inference	Encouraged to
Incorrect	$\mathcal{M}_\mathcal{A} = \mathcal{M}_\mathcal{B}$	Accept H_0	Not "statistically different"	Reduce sample
Correct	$\lvert\mathcal{M}_\mathcal{A} - \mathcal{M}_\mathcal{B}\rvert \geq \delta$	Reject H_0	Difference is significantly less than consequential	Expand sample

Table 2.2: Incorrect and correct practices of testing for "statistically significant equivalence" of systems $\mathcal{A}$ and $\mathcal{B}$.

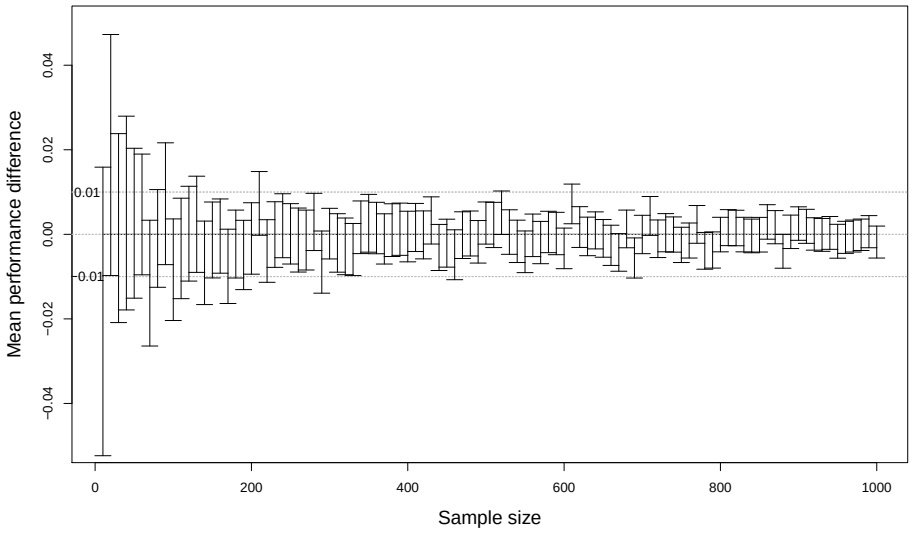


Figure 2.9: The 95% confidence interval for mean performance difference between two equal systems $\mathcal{A}$ and $\mathcal{B}$ with increasing sample sizes.

statistical best practices, a methodology for establishing similarity between two systems is discussed.

Inevitably, an average estimated on a sample is subject to variance, and hence uncertainty. Therefore, if hypothesis testing is used to recognise "statistically significant equivalence", a better practice is to consider the least difference in mean performance, δ, that one would consider as being consequential [Ahn et al., 2013]. Here, $|\mathcal{M}_\mathcal{A} - \mathcal{M}_\mathcal{B}| \geq \delta$, becomes the null hypothesis; and $|\mathcal{M}_\mathcal{A} - \mathcal{M}_\mathcal{B}| < \delta$ is the alternative hypothesis. The alternative hypothesis is accepted when the null hypothesis is rejected, that is the difference between system $\mathcal{A}$ and $\mathcal{B}$ is no more than δ and the systems are effectively equivalent. This process not only makes statistical sense, it also drives good

practice – the experimentalist is encouraged to increase the sample size and thus the accuracy of the average estimates. The pros and cons of these two methods of testing for statistically significant equivalence are summarized in Table 2.2.

The above principle can be indirectly tested using confidence intervals. If the confidence interval is above $-\delta$ and below δ, one can conclude that any difference between the two systems are not more than δ at a given level of confidence. Hence, one method is not consequently different than the other method. Performance differences between two systems are simulated to demonstrate the approach. Consider the case, where the mean effectiveness of systems $\mathcal{A}$ and $\mathcal{B}$ are equal, such that the expectation $\bar{\mathrm{m}}_{\mathcal{A}} - \bar{\mathrm{m}}_{\mathcal{B}}$ is 0. Assume that the difference in effectiveness between the two systems for a synthetic set of topics is sampled from a normal distribution $\mathcal{N}(0, \sigma^2)$. Assume δ to be 0.01. The 95% confidence interval for increasing values of $|L|$ is shown in Figure 2.9. Increasing the sample size causes the confidence intervals to narrow, and fit within $\pm\delta$. This exemplifies the encouraged behaviour of expanding the sample to find "statistically significant equivalence".

So far how systems are evaluated for differences as well as for similarity were discussed. Supporting only one source of variability is a key limitation in standard evaluation practices. Recall that how special analysis is required when evaluating IR systems with repeated measurements [Robertson and Kanoulas, 2012; Carterette et al., 2011]. Similarly, an increased variance is observed when effectiveness is repeatedly measured on a non-deterministic data mining algorithm [Granichin et al., 2014] or when effectiveness is assessed on different instantiations of a non-deterministic IR system [Aly et al., 2013]. Evaluating such systems require special treatment. In addition to considering variance in effectiveness, it is vital to maintain an unbiased test environment for system evaluation. Comparing two test environments is the focus of next section.

2.6 Comparing Test Environments

Experimentalists are compelled to evaluate systems in non-ideal test environments. A fair evaluation is expected to maintain the same system ranking as produced by the ideal test environment rather than the exact performance scores for the systems. The impact a non-ideal environment has on evaluation can indirectly be measured by assessing the correlation between the system rankings produced in the two environments for several systems. Correlation metrics for assessing the similarity between two system rankings are discussed next.

2.6.1 Pearson's Correlation Coefficient (PCC)

The similarity between two system rankings can be assessed by computing the correlation between the system scores that led to the rankings. Let x_{1i} and x_{2i} be the scores for $i{\cdot}th$ system of $|L_S|$ systems, and $\bar{x}_1$ and $\bar{x}_2$ be the respective mean scores for ideal and non-ideal test environments. The Pearsons correlation coefficient is given by:

$$PCC = \frac{\sum_{i=1}^{|L_S|}(\bar{x}_1 - x_{1i})(\bar{x}_2 - x_{2i})}{\sqrt{\sum_{i=1}^{|L_S|}(\bar{x}_1 - x_{1i})^2}\sqrt{\sum_{i=1}^{|L_S|}(\bar{x}_2 - x_{2i})^2}}.$$

The measure is sensitive to the magnitude of score differences. So the PCC is penalised for score differences even when there are no swaps in the system rankings. Therefore, methods that only consider ranking differences are widely used.

2.6.2 Spearman's Correlation Coefficient (SCC)

Spearman's correlation coefficient is based on the squared distance between ranks from the two test environments, and calculated as follows:

$$SCC = 1 - \frac{6\sum_{i=1}^{|L_S|}(x_{1i} - x_{2i})^2}{|L_S| \cdot (|L_S|^2 - 1)} \tag{2.25}$$

Here x_{1i} and x_{2i} are the rankings for $i{\cdot}th$ system of $|L_S|$ systems, rather than scores. A more popular alternative rank correlation measure is Kendall's Tau [Kendall, 1948].

2.6.3 Kendall's Tau (τ)

This measure is computed by counting the number of concordant and discordant pairs between the two system rankings. A pair of systems are concordant when the two systems are in the same relative order in the both rankings, and discordant when the relative order differs. When the number of concordant and discordant pairs are f_{con} and f_{dis}, Kendall's τ is:

$$\tau = \frac{f_{con} - f_{dis}}{f_{con} + f_{dis}} \tag{2.26}$$

When the two rankings are identical all pairs are concordant and $\tau = 1$. Similarly, all pairs are discordant and $\tau = -1$ when one ranking is the reversed order of the other. When the number of concordant and discordant pairs are the same, the Kendall's τ is 0.

Both Spearman's correlation coefficient and Kendall's τ correlation results in similar decisions [Gibbons, 1985]. However, the sampling distribution for Kendall's τ approaches a normal distribution more rapidly than Spearman's correlation coefficient as the number of systems increases [Gilpin, 1993]. Kendall's τ is widely used for this reason. However, ranking inversions at the top matters more than at the bottom when evaluating systems. This is addressed in AP correlation [Yilmaz et al., 2008a].

2.6.4 AP Correlation (τ_{ap})

The AP correlation is a weighted variant of the Kendall's τ correlation, and is computed as follows:

$$\tau_{ap} = \frac{2}{|L_S| - 1} \sum_{i=2}^{|L_S|} \frac{|con(i)|}{i-1} - 1 \tag{2.27}$$

Here $|con(i)|$ is the number of concordant pairs above rank i. The method penalizes discordant pairs like the Kendall's τ and the penalty is weighted using a scheme similar to the average precision. When the rank inversions are on the top or bottom of the rankings the AP correlation is lower or higher than the Kendall's τ correlation, and equivalent if inversions are uniformly spread.

2.6.5 Rank Correlation for IR systems

The above rank correlation methods assume systems are randomly sampled. However, IR presents a special scenario where not only systems but also topics are randomly sampled. When evaluating rank correlation with methods discussed so far system effectiveness is the average over a set of topics, hence the topical variance is ignored. Another concern is the significance: is the ranking on a non-ideal test environment significantly similar to the ranking with the ideal environment. While tests for significance are available for previously discussed methods [Cliff, 2014], they are rarely used due to wider confidence bounds and hence cause difficulty in finding significance [Carterette, 2009]. A more sophisticated method that considers topical effect is proposed by Carterette [2009]. The method severely penalises swaps in ranking between dissimilar systems more than similar systems. The method infers the significance in similarity using a bootstrap method.

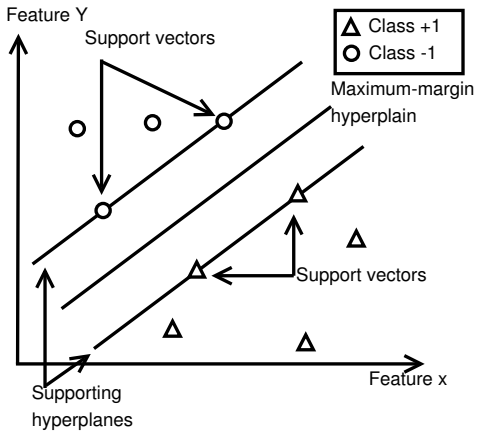


Figure 2.10: The maximum-margin hyperplane for a classifier concept trained with SVM.

Despite the lack of ambiguity in new methods for assessing rank correlation for IR experiments Kendall's τ is most widely used. A commonly used convention by Voorhees [2000] is that a τ greater than 0.9 is considered a similar ranking, similarities between 0.8 and 0.9 are as highly correlated, and below 0.8 as containing noticeable differences. No such convention exists for newer methods. So far evaluation in the context of IR and classification experiments were discussed. The data mining algorithms that are used to form classifier concepts in this thesis are discussed next.

2.7 Standard Data Mining Algorithms

In this section, several state of the art data mining algorithms for supervised machine learning are reviewed. They are: *Naïve Bayes*, *decision trees*, *support vector machines*, (SVM), and *logistic regression*.

2.7.1 Naïve Bayes

This simple probabilistic data mining algorithm applies Bayes' theorem using a strong independence assumption between features, and estimates the likelihood of each class (cls) given a set of feature values ($\{ft_i\}$) as follows:

$$p(cls/ft_1, ft_2, ...) = \frac{[p(ft_1/cls) \times p(ft_2/cls) \times ...] \times p(cls)}{p(ft_1) \times p(ft_2) \times ...} \tag{2.28}$$

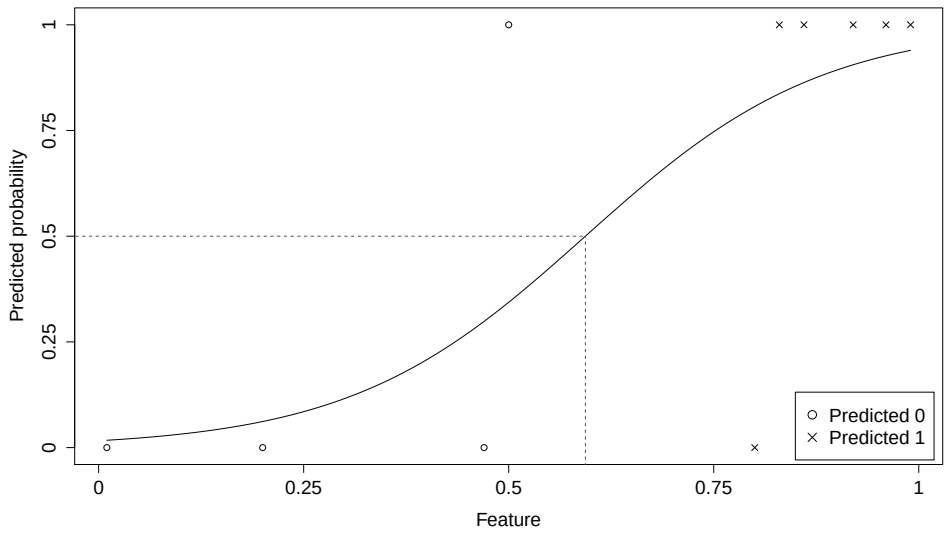
Here ft_i represents the value for the $i{\cdot}th$ feature in the dataset. The probabilities $p(ft_i/cls)$ and $p(ft_i)$, and $p(cls)$ for each ft_i and cls are estimated on a training dataset [Bishop, 2006]. Finally, the class with the maximum likelihood is predicted. The simplicity of the algorithm makes the naïve Bayes an attractive solution for classification tasks with large features spaces. This simple algorithm outperforms more advanced data mining algorithms in many datasets [Witten and Frank, 2005]. However, the naïve Bayes does not perform so well on all datasets where the assumption of independence between features is not valid.

2.7.2 Decision Trees

Decision trees transform the classification problem into a series of choices for each feature in the feature space [Marsland, 2009]. The most informative feature is used for the root node. The dataset is then partitioned by the category at the root node. The process is recursively applied to the child nodes for subsequent categorizations. Pruning the subtrees based on evaluation of a validation dataset is used to avoid overfitting in the classifier concept [Quinlan, 1993]. Traversing to the leaves of the decision tree provides a definitive classification for each test instance. The criteria for prioritising a feature can be *information gain*, *Gini Impurity* or any other suitable measure. Information gain is the most frequently used, and is the choice in the *C5.0* and *J48* algorithms [Witten and Frank, 2005].

2.7.3 Support Vector Machines (SVM)

Given a training dataset consisting of examples of two classes, a SVM produces a maximum-margin hyperplane separating instances from the two classes. The maximum-margin hyperplane is positioned to maximise the distance from data points of each class. The construction of the maximum-margin hyperplane requires the supporting hyperplanes to lie on a (usually sparse) subset of training examples from each class called support vectors as shown in Figure 2.10. The maximum-margin hyperplane is robust against slight variations in feature vectors [Hamel, 2009], and resilient to overfitting [Ma et al., 2009]. Further, the number of support vectors is independent to the number of the features, and makes a SVM more robust when using large feature spaces [Ayat et al., 2005]. This flexible generalization capability has been validated in various theoretical and empirical studies, and makes the SVM a popular candidate for classification over large feature spaces [Schölkopf and Smola, 2002]. In fact, the SVM is often the preferred choice for high dimensional classification problems, such as text classification, where the number of possible features can exceed the number of distinct instances



Example dataset:

Feature	0.01	0.2	0.47	0.5	0.8	0.83	0.86	0.92	0.96	0.99
Class	0	0	0	1	0	1	1	1	1	1

Figure 2.11: The logistic regression function for classifying examples in an example dataset containing a single feature.

available in the training data samples [Drucker et al., 1999; Joachims, 1998].

2.7.4 Logistic Regression

On a training dataset with examples from two classes (0 and 1) and a set of features ($\{ft_i\}$), a logistic regression algorithm computes a weight vector (ω) to fit the following model, so that the model defines the probability of an example belonging to class 1 [Witten and Frank, 2005].

$$p(1|ft_1, ft_2, ...) = \frac{1}{1 + exp(-\omega_0 - \omega_1 \times ft_1 - \omega_2 \times ft_2 - ...)} \tag{2.29}$$

The method measures the goodness of fit using the squared error of the *log-likelihood* function. The weight vector ω can be regularised on a validation dataset to prevent overfitting. The workings of a logistic regression classifier concept using an example dataset with a single feature is shown in Figure 2.11. The curve shows the logistic regression function fitted for the dataset. When the predicted probability is above 0.5 the classifier concept predicts the class 1 and vice versa.

2.8 Summary

In this chapter, evaluation metrics, and current practice of evaluating IR systems and classifiers are reviewed. Showing the difficulty of comparing systems on a sample, the importance of hypothesis testing is discussed. The necessity to avoid the common pitfall of misusing a lack of significance as a proof that two systems have equivalent effectiveness is clarified. How statistics can safely be used to establish that two system have equivalent effectiveness is illustrated. How test environments are evaluated is also discussed. Finally, several state of the art data mining algorithms used in the thesis are presented.

Abbreviations and Symbols used in Chapter 3

Notation	Description
κ	The depth to which a ranking is possessed to form the second pool for assessment (see Section 3.3).
ARJ	Automatic runs judged.
ARU	Automatic runs unjudged.
BC	Borda Count.
CBC	Combined - Borda Count.
CCMNZ	Combined - CombMNZ.
CMNZ	CombMNZ.
ML	Machine Learning.
MRJ	Manual runs judged.
Q	Number of documents retrieved per topic in a retrieval run.
q	The depth to which retrieval runs are pooled for manual assessment (pool depth).
$\acute{r}$	Judged non-relevant.
r	Relevant.

Chapter 3

Handling Bias Due To Incomplete Relevance Judgements

Successful evaluation and reproducibility of experiments in IR depends on building reusable test collections composed of a corpus of documents, topics, and relevance judgements. Ideally every document in the corpus would be assessed against each topic, but this approach does not scale. So relevance judgements are normally produced for a sample of the corpus, known as a pool and all unjudged documents are assumed to be not relevant. This sample needs to be representative of the entire corpus and robust enough to evaluate entirely new search algorithms. The genesis of pooling dates back to the 1970s [Spärck Jones and Van Rijsbergen, 1975; Spärck Jones and Bates, 1977].

To produce relevance judgements, the organizers of the TREC, CLEF, NTCIR, and other conferences invite researchers to submit ranked retrieval results produced by state of the art IR systems (*automatic run*s) up to a depth of Q for a set of topics on a specified corpus (typically $Q = 1000$) [Voorhees, 2002; Voorhees and Harman, 2005]. The top-q (*pool depth*) ranked documents in these runs for each topic are then gathered for assessment. Such a practice seems to identify many relevant documents, but provides no guarantee on the judgement coverage for previously unseen approaches [Zobel, 1998; Carterette et al., 2010a; Sakai et al., 2012; Sakai, 2013]. Test collections tend to have a bias towards IR systems contributing to the pool, and may not reliably evaluate novel IR systems that retrieve unjudged but relevant documents.

The bias in test collections can be counteracted by: (1) providing new evaluation methods that account for incomplete judgements, or (2) finding relevant documents until the test collection is sufficiently complete for accurate evaluation. In this chapter, the following contributions are made to

reduce test collection bias:

1. Evaluation methods robust against incomplete judgements minimise and/or account for uncertainty due to unjudged documents. One such class of methods predict probabilistic relevance of retrieved unjudged documents [Büttcher et al., 2007; Carterette and Allan, 2007; Carterette et al., 2010a]. These methods capture the uncertainty in mean effectiveness due to the sampling of topics and probabilistic relevance judgements as a confidence interval incorporated into an evaluation metric [Carterette and Allan, 2007; Carterette et al., 2010a]. In this chapter, a bootstrap algorithm for constructing confidence intervals independent of the evaluation metric is proposed in Section 3.2. The proposed evaluation methodology is used to compare and contrast existing methods that predict probabilistic relevance of retrieved unjudged documents. In contrast to prior research, which examined predictive methods using binary relevance judgements, predictive methods are analysed and compared in the context of graded relevance judgements in Section 3.2.4.

2. In order to "future proof" test collections, the organizers of TREC style conferences encourage submissions of *manual run*s, where humans can reformulate queries and/or merge results from multiple queries [Voorhees and Harman, 1998; Buckley et al., 2007] before submitting their final set of top-Q documents. Such runs are generally highly effective and contribute many unique relevant documents to the judgement pool [Soboroff and Robertson, 2003]. Manual runs are not always available when building a test collection. A methodology to construct reliable IR test collections with low bias is proposed using only automatic runs in Section 3.3.

Prior work in investigating uncertainty due to incomplete relevance judgements is reviewed next.

3.1 Related Work

The suitability of a test collection to evaluate novel IR systems in the presence of incomplete judgements has been previously studied. To assess the impact of bias due to incomplete judgements, Zobel [1998] used *leave one run out* simulations. If a pooled run had not been available, unique relevant documents contributed by the run would not have been judged for relevance, and assumed not relevant. The effectiveness for the system with and without the run in the pool is used to assess the impact of incomplete judgements on evaluation. On early test collections, Zobel [1998] found no evidence against reuse.

Presuming multiple submissions from the same group retrieve similar documents, Voorhees [2002]

adapted the above method to exclude all runs from a group (*leave one group out*) rather than one run. In this formulation a group of runs are assessed using relevance judgements produced with and without the group of runs being in the pool. Since then, leave one group out methodology has become the de facto standard for evaluating test collections for reusability. The more stringent leave one group out method also did not find any evidence against reuse of early test collections.

Evidence against reusability of test collections started to appear with larger test collections. Buckley et al. [2007] reported an unusual manual run, which was scored lower than expected if the run had not been pooled. Digging into the problem revealed that large test collections contain many more documents with query terms than a practically assessable pool can hold. Hence, relevant documents without query terms are often left out from the pool. Further evidence appeared in an experiment where runs were evaluated with holding out manual runs from the pool [Büttcher et al., 2007]. The new system ranking was found to be different to the ranking produced with full relevance judgements. Whether the test collection bias causes a considerable impact on ranking of brand new systems is unknown when a very small number of examples found to be affected and finding all affected instances is not possible without extra judgements. As such, IR researchers began searching for evaluation methods robust against incomplete judgements, and efficient and effective ways of locating relevant documents in test collections.

3.1.1 Robust Evaluation Methods

Buckley and Voorhees [2004] showed that traditional evaluation metrics such as AP, R-precision, and P@D are not sufficient when judgements are incomplete. Therefore, evaluation methods robust against incomplete judgements are required.

When relevance of a document is a binary decision, the deviation of a retrieval run from a perfect run can be computed based on how many non-relevant documents are retrieved before a relevant one, and is the basis for the first evaluation metric proposed for this problem. The method, called Bpref, computes the effectiveness score for a topic as follows [Buckley and Voorhees, 2004]:

$$\text{Bpref} = \frac{1}{|d_r|} \sum_{\forall \{d_r^\star\}} 1 - \frac{|d_{\acute{r}}^\star \text{ ranked before } d_r^\star|}{min(|d_r|, |d_{\acute{r}}|)}. \tag{3.1}$$

Here $|d_r|$ and $|d_{\acute{r}}|$ denote the number of relevant and judged non-relevant documents for the topic, $d_r^\star$ is a retrieved relevant document, and $d_{\acute{r}}^\star$ is a document among the top $|d_r|$ retrieved non-relevant documents. The deviation from a perfect run is scaled to a value between 0 and 1 by dividing it from

the lower of $|d_r|$ or $|d_{\acute{r}}|$. The authors pointed out that the stability of the metric is due to ignoring unjudged documents in the ranked retrieval results which was earlier observed by Hersh et al. [1994]. System rankings for Bpref and MAP are highly correlated when judgements are complete. But as the number of unjudged documents in the collection increased, Bpref maintained a similar system ranking to an original system ranking than as seen with MAP. De Beer and Moens [2006] generalised Bpref to handle graded relevance judgements in Rpref by altering the penalty to consider weighted counts of number of less relevant documents ranked before more relevant documents.

A more robust evaluation metric was subsequently proposed by Yilmaz and Aslam [2006]. The metric, called induced AP (indAP) is computed as follows:

$$\text{indAP} = \frac{1}{|d_r|} \sum_{\forall \{d_r^\star\}} 1 - \frac{|d_{\acute{r}}^\star \text{ ranked before } d_r^\star|}{rank(d_r^\star)}. \tag{3.2}$$

Note that the metric indAP differs from Bpref only by the scaling factor.

Recall that the robustness in the above metrics is due to ignoring unjudged documents rather than considering them as non-relevant. Hence, an alternative is to use traditional evaluation metrics: Q-measure, NDCG, and AP with a condensed retrieval run – each run only includes judged documents [Sakai, 2007; Sakai and Kando, 2007; 2008]. The study further illustrated that a boost in robustness is observed when judgements are graded. Hence, Q-measure and NDCG are more robust than MAP on a condensed run.

Rather than ignoring unjudged documents, information carried with them such as ranked positions of retrieved relevant documents are used to better estimate AP in *inferred AP (*infAP*)* [Yilmaz and Aslam, 2006]. If $rank(d_r^\star)$ is the rank of the relevant document $d_r^\star$, a randomly chosen document from retrieved results up to rank $rank(d_r^\star)$ may yield the document $d_r^\star$ with the probability of $1/rank(d_r^\star)$ or any other document up to $rank(d_r^\star) - 1$ with the probability of $[rank(d_r^\star) - 1]/rank(d_r^\star)$. Hence, the expected precision at rank $rank(d_r^\star)$ can be recursively computed as follows:

$$E[\text{P@}rank(d_r^\star)] = \frac{1}{rank(d_r^\star)} \cdot rel(d_r^\star) + \frac{rank(d_r^\star) - 1}{rank(d_r^\star)} \cdot E[\text{precision above } rank(d_r^\star)]. \tag{3.3}$$

Here $d_r^\star$ being a relevant document, $rel(d_r^\star) = 1$. Assuming any document not included in the depth 100 pool is not relevant, and the probability of any other document up to $rank(d_r^\star) - 1$ being relevant is the proportion of judged relevant documents to total judged documents, the method computes the

E[precision above $rank(d_r^\star)$] as follows:

$$E[\text{precision above } rank(d_r^\star)] = \frac{|\{d_{100}^{'}\}|}{rank(d_r^\star) - 1} \cdot rel(d_{100}^{'}) + \frac{|\{d_{100}\}|}{rank(d_r^\star) - 1} \cdot \frac{|\text{Relevant}| + \lambda}{|\text{Total Judged}| + 2\lambda}. \tag{3.4}$$

Here d_{100} is a document from the depth 100 pool found above the rank $rank(d_r^\star)$ in retrieval results, $d_{100}^{'}$ is a document not in the depth 100 pool, and λ is a smoothing parameter [Chen and Goodman, 1996]. The $rel(d_{100}^{'})$ is 0. Hence,

$$\mathsf{infAP} = \frac{1}{|d_r|} \sum_{\forall \{d_r^\star\}} E[\mathsf{P@}rank(d_r^\star)] = \frac{1}{|d_r|} \sum_{\forall \{d_r^\star\}} \left(\frac{1}{rank(d_r^\star)} + \frac{|\{d_{100}\}|}{rank(d_r^\star)} \cdot \frac{|\text{Relevant}| + \lambda}{|\text{Total Judged}| + 2\lambda} \right). \tag{3.5}$$

When the judgement pool is under represented as a result of a lack of variety in participating systems or low pool depths, all of the above evaluation metrics overestimate the effectiveness of new systems, while the standard evaluation metrics underestimate it [Sakai, 2008a;b; 2009]. Therefore, more accurate and robust alternatives for evaluation are still desirable.

The next class of methods require fewer extra judgements for evaluation. One such method recognises the minimum additional judgement effort required for accurate evaluation [Carterette et al., 2006; Carterette, 2007]. For example, to objectively compare two IR systems using P@D, only judgements for uniquely retrieved unjudged documents to a depth of D are required.

Effectiveness for a topic can also be statistically inferred from a random sample of retrieved documents. When a random sample is used exactness in a measurement is traded off for efficiency. Hence, effectiveness for each topic is observed as a confidence interval, but does not require judgements for all retrieved documents. For example, infAP can be inferred on a random sample of retrieved documents [Yilmaz et al., 2008b]. Here, a set of relevant documents $\{d_r^\star\}$, and documents above $rank(d_r^\star)$ for computing E[precision above rank $rank(d_r^\star)$] are randomly sampled. Using formulae specific to infAP, confidence bounds for mean effectiveness are estimated.

Standard evaluation metrics such as AP, R-precision, and P@D can also be inferred from a random sample of retrieved documents [Aslam et al., 2006; Aslam and Pavlu, 2007]. Each evaluation metric is estimated as the expectation of a random variable x_i with respect to a probability distribution

p_i. For example, to compute P@D, let:

$$x_i = \begin{cases} 0 & \text{if } d_i \text{ is non-relevant, and} \\ 1 & \text{otherwise;} \end{cases} \tag{3.6}$$

and

$$p_i = \frac{1}{\mathsf{D}} \qquad \forall 1 \leq i \leq \mathsf{D}. \tag{3.7}$$

L be a sample drawn from the probability distribution p_i. Now the expected value of P@D is:

$$E[\mathsf{P@D}] = \frac{1}{|L|} \sum_{\forall i \in L} x_i. \tag{3.8}$$

With uniform random sampling, unbiased estimates for the $E[\mathsf{P@D}]$ can be obtained. As for the statistical theory the variance in the above estimate is $var(x_i)/|L|$, where $var(x_i)$ is the variance in x_i and $|L|$ is the sample size. The variance in the estimate can be minimised by reducing $var(x_i)$ or increasing the sample size. The latter requires more relevance judgements, hence the former is preferred. The $var(x_i)$ can be reduced by sampling more at the top of the retrieval results than the tail as this is the region which is expected to contain more relevant documents. The x_i is adjusted accordingly to count less for over sampled regions and count more for under sampled regions. Low variance results in a narrow confidence interval, and a more accurate mean estimate.

These methods require new judgements for unbiased evaluation. No matter how small the judgement effort is, researchers are reluctant to judge more documents due to (1) resource and time constraints, and (2) potential acceptability issues.

Another category of approaches learn a concept per topic using existing relevance judgements, and predict relevance of unjudged documents in retrieval runs. One such approach represents documents in a TF · IDF vector space, where each term is a unique dimension [Büttcher et al., 2007]. A classifier is trained using a suitable data mining algorithm such as SVM, with judged documents as the training dataset where binary relevance form the labels. The trained classifier is then used to predict the binary relevance of retrieved unjudged documents. This method is referred to as Predictive Method 1 henceforth.

Another method, referred to as Predictive Method 2 in this chapter, outputs the probability of relevance of an unjudged document to a topic [Carterette and Allan, 2007; Carterette et al., 2010a]. The method uses a vector space with a dimension for each judged document. Documents are represented in the vector space using *cosine* similarity. The judged documents for each topic represented

in the above manner form the training dataset, of which the probability of relevance for a relevant and non-relevant document is one and zero respectively. The learned concept is then derived using a regularized logistic regression classifier. The method utilises mathematically proven formulae specific to an evaluation metric to infer effectiveness and the confidence intervals. Such confidence intervals account for the uncertainty due to the sample of topics chosen, and probabilistic judgements. To narrow the confidence interval, the experimenter is required to judge retrieved unjudged documents, and/or increase the number of topics.

Predictive methods minimise the bias in test collections, but are not free from errors. A machine learning concept recognises relevant documents based on a training dataset, and therefore can embed a bias due to the lack of variety in training data.

Increasing the number of topics improves the accuracy of evaluation. As most of the relevant documents are found at the top of the rankings, a lower pool depth can be used with a larger topic set. Hence, such a strategy can be implemented by test collection curators at a no extra expense [Sanderson and Zobel, 2005; Jensen, 2006; Bodoff and Li, 2007; Carterette and Smucker, 2007; Carterette et al., 2008; Webber et al., 2008; Sakai and Mitamura, 2010]. Although such test collections are better at evaluation, insufficient judgements can make failure analysis and training machine learning algorithms difficult. Therefore, sufficiently completing the test collection is a vital task. Such effort is carried out via effective and efficient approaches to build pools. Related work is reviewed next.

3.1.2 Completing Test Collections

Zobel [1998] showed that some topics have many more relevant documents than others, and therefore assessors should focus their effort on judging topics with more relevant documents. For each topic, the number of relevant documents found were used to estimate the expected ratio of relevant documents in the remaining unjudged block. Each topic would be assessed until relevant documents were depleted beyond a limit economically viable to assess the block.

The idea of focusing assessor effort on the most fruitful sources of relevant documents was also applied to IR systems that contribute to a pool. Just as some topics have more relevant documents than others, some systems retrieve more relevant documents than others. Using this insight, Cormack et al. [1998] describe a move-to-front pooling system which ensures that documents from the IR systems producing the most relevant documents are moved to the front of the queue of documents to be judged. Cormack et al. [1998] go on to show that combining these two insights enables more assessing effort to be spent on the most effective systems and on the best topics.

Alternatively, ranked retrieval results from multiple IR systems can be merged to derive a fused ranking of documents. Each IR system produces a list of top-Q documents ranked by preference. Fusion schemes score each retrieved document according to a specific criteria. The documents are then ranked in descending order by the fusion score. Ties are broken randomly. Documents in the new ranking are assessed until a fixed judging capacity of the pool is reached. Popular fusion schemes are: *Borda Count* (BC) [Aslam and Montague, 2001], *CombSum*, *CombMNZ* (CMNZ), *CombANZ* [Fox and Shaw, 1993], and *Static Judgement Orderings* [Moffat et al., 2007]. The criteria for scoring documents with each fusion method is presented next.

Borda Count scores documents using a simple voting method. For each list the highest ranked document receives Q votes, the second highest document receives $Q - 1$ votes, and so forth. For fused ranking each document is scored with the total number of votes received from all systems. CombSum is also a voting method. Each list votes for each document retrieved in the list with the normalized relevance score given by the IR system for the document. Again, the total votes received from all systems for each document defines the score for ranking. CombMNZ defines the fusion score as the CombSum score multiplied by the number of systems that retrieved the document. The fusion scores as defined by the CombSum are averaged over the number of systems that retrieved the document for CombANZ. SJO computes the fusion score as the sum of $(1 - \rho) \cdot \rho^{rank(d)-1}$ from all systems, where $rank(d)$ is the rank of document d in a ranked list. An appropriate value for ρ is empirically determined to maximise effectiveness of the fused ranking. On distinct datasets different fusion approaches have shown to be effective [Aslam and Montague, 2001]. However, CMNZ is widely used.

An online learning algorithm for selecting the next document to judge is proposed by Aslam et al. [2003]. Here the next document to be judged is chosen as a weighted expert opinion with each IR system viewed as an expert. Each system's expertness score increases with each predicted relevant document and decrease when the predicted document is non-relevant. Each system recommends documents based on the rank of the document weighted by its expertness. The next document to judge is the one with the strongest aggregated recommendation from all systems. The method found relevant documents at an approximate rate of 50% higher than conventional polling.

The methods discussed so far make no distinction between automatic and manual runs. The methods may potentially aggravate the pooling bias by favouring clusters of similar systems [Webber, 2011]. Hence, no guarantee on confidence in the reliability of the test collection in the absence of manual runs is offered for traditional evaluation metrics. An alternative method which used no pooling of runs, *Interactive Search and Judging* (ISJ) requires assessors to manually find relevant

documents by searching while judging at the same time [Cormack et al., 1998]. The method identified a similar number of relevant documents to the traditional pooling approach with 75% less work than standard TREC pooling.

However, the approach was not adopted by TREC as the assessors were judged to be better at assessing documents than as query reformulators [Soboroff and Robertson, 2003]. Soboroff and Robertson [2003] went on to argue that relevance feedback can be used to replace manual reformulation of queries in ISJ. After assessing the initial pool of documents produced with an automatic run, relevance feedback is applied using seven ranking strategies including a few machine learning rankers in order to obtain a newer set of retrieval results. The retrieval results are fused using CombMNZ, and the top ranked results in the fused list are assessed. The relevance feedback is applied iteratively to find new relevant documents. For the machine learning rankers a SVM and a Naïve Bayes classifiers are used on a BOW feature space. This is the first effort to construct test collections with only automatic runs. Next the focus is turned to research question 1-(a).

3.2 Evaluating IR Systems with Predicted Relevance Judgements

Recall that a class of methods for evaluating IR systems with incomplete relevance assessments rely on predicting the relevance of retrieved unjudged documents. Each prediction is the probability of an unjudged document being relevant to a topic. Effectiveness computed for a topic with probabilistic relevance judgements cause the effectiveness score to vary for each repeated assessment. In prior research, the variance in average effectiveness caused by varying effectiveness for each topic and sampling of topics is captured in a confidence interval using mathematically proven formulae specific to an evaluation metric [Carterette et al., 2010a]. The analysis is limited to binary relevance assessments [Büttcher et al., 2007; Carterette et al., 2010a]. In this section, a bootstrap method independent of the evaluation metric is proposed for computing confidence intervals for mean effectiveness when judgements are probabilistic. Using the proposed evaluation framework, the effectiveness of Predictive Method 1 and Predictive Method 2 (defined in Section 3.1.1) are examined in the context of graded relevance judgements.

3.2.1 Methodology

For each topic, to predict the probable relevance of unjudged documents, Predictive Method 1 and Predictive Method 2 require training a linear SVM classifier and a logistic regression classifier re-

ALGORITHM 3 Bootstrap algorithm for computing a confidence interval for mean effectiveness of IR system $\mathcal{A}$, when relevance of an unjudged document to a topic is a probability function.

$BS \leftarrow$ Number of bootstrap samples;
$N \leftarrow$ Number of topics;
$[a_{ni}] \leftarrow$ A matrix of $i{\cdot}th$ repeated measurement for $n{\cdot}th$ topic on IR system $\mathcal{A}$;

// Construct bootstrap samples using resampling with replacement.
$bootstrap\_sample\_means \leftarrow []$;
for $j = 1$ *to* BS **do**
    $a^*_j \leftarrow$ Resample N rows with replacement from $[a_{ni}]$, for each selected row each time randomly selecting a measurement.
    $bootstrap\_sample\_means.append(mean(a^*_j))$;
end
$sort(bootstrap\_sample\_means)$;
Confidence interval $\leftarrow \alpha/2$ and $1 - \alpha/2$ percentiles of $bootstrap\_sample\_means$;

spectively. In contrast to prior experiments where binary relevance assessments are considered, here Predictive Method 1 and Predictive Method 2 are extended to predict graded relevance assessments for unjudged documents. The graded relevance judgements pose a multi-class classification problem that can be resolved using a one versus rest strategy, where the probability of relevance to each grade is a binary classification problem. These can be ensembled to compute the final probability of relevance [Fan et al., 2008; Keerthi et al., 2008]. With Predictive Method 1 judgements are definitive, and the predicted grade has a probability of 1. With Predictive Method 2 judgements are probabilistic, and therefore each document has some chance of being relevant at each graded level. Judged documents for each topic form the training dataset for the topic. The grade of the relevance judgement for each judged document is the label for the document. The trained machine learning concepts are used to predict relevance of unjudged documents in retrieval results. More predictive methods can be derived by using a probabilistic SVM classifier, or by considering definitive predictions from the logistic regression classifier, or by using alternative data mining algorithms. However, this study is limited to investigating existing predictive methods that have been successfully used to predict binary relevance judgements for predicting graded relevance judgements. The probabilistic relevance assessments derived using Predictive Method 2 cause the effectiveness for each topic to vary on each repeated assessment. Therefore, a novel bootstrap methodology for computing the confidence interval for mean effectiveness of an IR system when judgements are probabilistic is discussed next.

The pseudocode for the bootstrap evaluation methodology is presented in Algorithm 3. Probabilistic relevance judgements cause the effectiveness for a topic to vary on each repeated assessment.

The matrix $[a_{ni}]$ holds the repeated assessments of effectiveness for N topics on IR system $\mathcal{A}$. Recall the bootstrap method uses resampling of topics in the original sample with replacement. Therefore, BS bootstrap samples are formed for which each time a topic n is resampled, an effectiveness score from the repeated measurements for the topic is randomly picked. The $\alpha/2 \times 100$ and $1 - \alpha/2 \times 100$ percentiles of the sorted bootstrap sample means are the bounds for the $(1 - \alpha) \times 100\%$ confidence interval.

3.2.2 Datasets

Experiments are performed on the TREC GOV2 dataset. Runs submitted for 2004.adhoc, 2005.adhoc, and 2006.adhoc tasks and the relevance assessments for TREC topics 701–750, 751–800, and 801–850 are separately used respectively.

3.2.3 Experimental Setup

For all experiments, documents are Krovetz stemmed [Krovetz, 1993], and the leave one group out methodology is used. For each topic and left out group judged documents are split into two folds. The first fold holds documents exclusively pooled by the left out group, while the second fold holds the rest. The second fold of judged documents and the data mining software LIBLINEAR [Fan et al., 2008] are used to train a multi-class linear SVM classifier and a multi-class regularised logistic regression regression classifier for Predictive Method 1 and Predictive Method 2 respectively. The classifiers are then used to obtain probabilistic relevance assessments for documents in the first fold. Effectiveness for each topic, and run in the left out group are repeatedly assessed 100 times by computing an evaluation metric using the relevance judgements from the second fold and predicted relevance judgements from the first fold. Here relevance judgements for the first fold are sampled from the corresponding predicted probability distributions. The bootstrap evaluation methodology presented in Algorithm 3 is used to compute the confidence interval for the mean effectiveness of each run in the left out group.

3.2.4 Evaluation

The Kendall's τ correlation and AP correlation (τ_{AP}) are used to compute pairwise inversions between two mean effectiveness rankings of runs, the first using full relevance assessments and the second using incomplete relevance assessments with a predictive approach in the leave one group out setting. In contrast to Kendall's τ correlation, the AP correlation is more sensitive to inversions at the

Kendall's τ correlation

Metric	Method	2004.adhoc	2005.adhoc	2006.adhoc
NDCG	Incomplete Judgements	0.9793	0.9879	0.9777
	Predictive Method 1	**0.9940**	**0.9976**	**0.9867**
	Predictive Method 2	0.9793	0.9900	0.9525
NDCG@10	Incomplete Judgements	0.9548	0.9852	0.9568
	Predictive Method 1	**0.9822**	**0.9885**	**0.9805**
	Predictive Method 2	0.9698	0.9837	0.9699
NDCG@30	Incomplete Judgements	0.9319	0.9749	0.9438
	Predictive Method 1	**0.9869**	0.9776	**0.9851**
	Predictive Method 2	0.9691	0.9824	0.9528

AP correlation

Metric	Method	2004.adhoc	2005.adhoc	2006.adhoc
NDCG	Incomplete Judgements	0.9425	0.9867	0.9787
	Predictive Method 1	0.9632	**0.9983**	**0.9851**
	Predictive Method 2	0.9793	0.9855	0.9479
NDCG@10	Incomplete Judgements	0.9357	0.9795	0.9516
	Predictive Method 1	**0.9845**	**0.9900**	**0.9776**
	Predictive Method 2	0.9665	0.9861	0.9557
NDCG@30	Incomplete Judgements	0.8827	0.9630	0.9340
	Predictive Method 1	**0.9821**	0.9630	**0.9772**
	Predictive Method 2	0.9535	0.9738	0.9369

Table 3.1: Kendall's τ and AP correlation between system rankings computed with incomplete relevance assessments in the leave one group out setting and full relevance assessments. Instances with a higher correlation score for Predictive Method 1 *than the other methods are bold-faced.*

top of the ranking than the bottom. Hence, the AP correlation is considered more appropriate for IR evaluation. However, Kendall's τ is widely used. Therefore, results are presented for both correlation measures. Here the mean effectiveness for a topic on a run evaluated with probabilistic relevance judgements is the mean effectiveness over 100 repeated observations. The results are presented in Table 3.1.

An approach which constantly under or over estimates effectiveness may produce a similar sys-

Metric	Method	2004.adhoc	2005.adhoc	2006.adhoc
NDCG	Incomplete Judgements	0.9144	0.9706	0.9749
	Predictive Method 1	$0.9842^{\ddagger}$	$0.9854^{\ddagger}$	$0.983^{\curlywedge}$
	Predictive Method 2	0.8805	0.9675	0.9182
NDCG@10	Incomplete Judgements	0.9344	0.9775	0.9738
	Predictive Method 1	$0.9678^{\ddagger}$	0.9833	$0.9865^{\ddagger}$
	Predictive Method 2	0.9392	0.9728	0.9145
NDCG@30	Incomplete Judgements	0.8904	0.9651	0.9599
	Predictive Method 1	$0.9656^{\ddagger}$	$0.9833^{\dagger}$	$0.9832^{\ddagger}$
	Predictive Method 2	0.914	0.9631	0.8952

Table 3.2: The similarity between 95% confidence intervals for mean effectiveness estimated with incomplete judgements in leave one group out setting and full relevance assessments. The confidence intervals with Predictive Method 1 *is compared with just* Incomplete Judgements *for significance. The symbols* $^{\curlywedge}$, †, *and* ‡ *indicates a significant difference at* 0.1, 0.05 *and* 0.01 *respectively.*

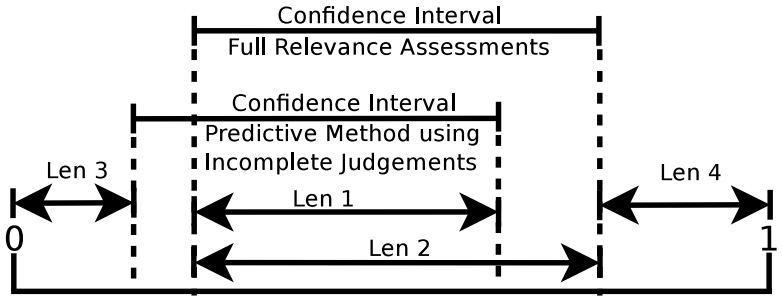


Figure 3.1: Assessing similarity between two confidence intervals produced with full and incomplete relevance assessments. Here $x_1 = Len\ 1/Len\ 2$, *and* $x_2 = Len\ 1 + Len\ 3 + Len\ 4$. *Similarity* $= (2 \cdot x_1 \cdot x_2)/(x_1 + x_2)$.

tem ranking to the one produced using full relevance assessments. However, an approach producing a similar ranking and with a true sense of effectiveness is preferred. A predictive approach with no uncertainty in prediction will produce the same relevance judgements as a manual assessment (assuming no uncertainty in manual assessment). Hence, one may compare the similarity between confidence intervals produced with the full relevance assessments, and the predictive approaches using incomplete relevance assessments. The similarity between two confidence intervals is computed as follows. The confidence intervals for mean effectiveness of IR systems lie between 0 and 1. Let x_1 be the overlap length of two confidence intervals divided by the length of the confidence interval produced with full relevance assessments, and x_2 be the length between 0 and 1 agreed by both confidence intervals. The similarity between two confidence intervals is defined as the harmonic mean for x_1

and x_2. An example of the computation is shown in Figure 3.1. The mean similarity of confidence intervals for submitted runs in the leave one group out setting are analysed in Table 3.2.

Predictive Method 1 achieved a better Kendall's τ correlation and a better AP correlation than Predictive Method 2 on a majority of datasets. The similarity between confidence intervals produced by a method in the leave one group out setting and with full relevance assessments is better for Predictive Method 1 than for just incomplete judgements on all datasets, and significant on all but one datasets. However, the same for Predictive Method 2 on a majority of datasets are worse than that of for just incomplete judgements. This indicates a constant over or under estimation of effectiveness with Predictive Method 2. Therefore, Predictive Method 1 is a better choice than Predictive Method 2 for predicting relevance of unjudged documents in the context of graded relevance assessments on the TREC GOV2 dataset. The low effectiveness of Predictive Method 2 could be a consequence of lack of information in the summarised feature space, compared to the larger feature space used by Predictive Method 1 for machine learning.

3.2.5 Summary

Here, existing predictive methods for evaluating IR systems when judgements are incomplete are compared in the context of graded relevance judgements using a novel bootstrap approach. The bootstrap method is generic (independent of evaluation metric) and accounts for uncertainty due to selection of topics and probabilistic relevance judgements. Predictive Method 1 resulted in a better evaluation compared to the traditional approach of assuming unjudged documents are non-relevant and Predictive Method 2 in the context of graded relevance assessments on the TREC GOV2 dataset. The research question 1-(b) is analysed next.

3.3 Extending Test Collection Pools without Manual Runs

None of the solutions reviewed in Section 3.1 for evaluating IR systems with incomplete judgements is entirely satisfying. Effective IR systems may emerge that retrieve new relevant documents [Carterette et al., 2010b]. To minimise the chance of unfairly judging such systems, proper judgement coverage must be maintained. The traditional approach for pooling relies on manual runs for diversity in coverage [Soboroff and Robertson, 2003]. In the next section, methods for future proofing test collections in the absence of manual runs are proposed.

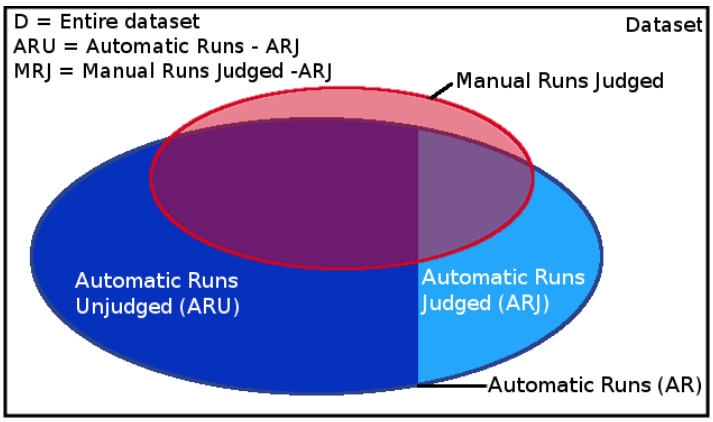


Figure 3.2: The subsets of the document corpus corresponding to formation of a pool for judging documents.

3.3.1 Potential Methods

This section commences by defining subsets of a document corpus pertaining to a pool for judging documents. The following subsets illustrated in Figure 3.2 are of note. *Automatic runs judged (ARJ)* is the pool of top-q documents from the automatic runs. *Automatic runs unjudged (ARU)* is the pool of top-Q documents minus ARJ from the automatic runs. Similarly *Manual runs judged (MRJ)* is the top-q documents of the manual runs less documents in ARJ. Pooling strategies to locate relevant documents that are in ARU are investigated.

The approaches work in two steps: a *first pool* is drawn with documents from set ARJ and evaluated. Then a set of documents that are not in the first pool are drawn from the collection. The documents are ranked by one of the following approaches and the top-κ are drawn into a *second pool* which is also evaluated.

In this section the following approaches are analysed for ranking documents to form the second pool: simple fusion methods – Borda Count (BC) and CombMNZ (CMNZ) – naïve baselines; a machine learning classifier (ML) – similar in spirit to the approach proposed by Soboroff and Robertson [2003] – another baseline; and a combination of fusion methods and machine learning.

Fusion Methods

Recall that fusion methods can be used to fuse multiple ranked lists to a consolidated ranked list. Here a consolidated ranked list is derived by fusing the automatic runs. Remember only ARU documents qualify for inclusion in the second pool. Therefore, ARJ documents are removed from the fused

ranking. The top-κ from the final ranking are drawn into the second pool.

Machine Learning Approach (ML)

A linear SVM classifier per topic is trained on the documents in the first pool. Here the choice of classification algorithm is due to proven effectiveness and efficiency of linear SVM in high dimensional text classification problems [Drucker et al., 1999; Joachims, 1998]. Documents are represented in a vector space using Krovetz stemming and a TF $\cdot$ IDF weighting. A similar feature space was used by Büttcher et al. [2007] to predict the relevance of an unjudged document for a given topic (Predictive Method 1). After training the linear SVM classifier for each topic, documents are scored and ranked by the classifier.

The linear SVM classifier computes a weight vector ω corresponding to the maximum-margin hyperplane that maximally separates relevant (labeled 1) and non-relevant (labeled 0) documents of the set ARJ. The decision function $h(X)$ for the classifier is given by $sign(\omega^{\mathcal{T}} \cdot X)$, where X contains document feature vectors. The relevance score for the $i{\cdot}th$ document X_i is taken from $\omega^{\mathcal{T}} \cdot X_i$. The documents not in ARJ are sorted by their score and the top-κ are drawn into the second pool.

This method is similar in spirit to the use of relevance feedback in the approach proposed by Soboroff and Robertson [2003] but differs in the following aspects. In Soboroff and Robertson [2003] top ranked documents from an automatic run are judged, and used as relevance feedback to rank documents using seven ranking strategies, including a few machine learning approaches. The ranked results are fused, top ranked documents are assessed, and relevance feedback is iteratively applied. Here, relevance feedback is applied to relevant documents found in many automatic runs, and uses only one machine learning ranker. Multiple machine learning rankers are not used to avoid the high computational overhead which is substantial for large datasets. For the same reason, repeated relevance feedback is not used.

Combined Methods

The third category of approaches combine the first two methods. Here, a fusion method is used as a filter to generate a subset of the most promising documents in the collection for a given topic. The subset is then reranked using the machine learning classifier, and the top-κ form the second pool. In contrast to the method proposed by Soboroff and Robertson [2003], here fusion is used to select a subset of documents to rank using the machine learning approach, rather than to form a ranking for

manual assessment. The combined approaches using Borda Count and CombMNZ are respectively referred to as *Combined - Borda Count* (CBC) and *Combined - CombMNZ* (CCMNZ) henceforth.

3.3.2 Datasets

The TREC 8 and the TREC GOV2 datasets are used with TREC topics 401–450 [Voorhees and Harman, 2000] and 801–850 [Büttcher et al., 2006] respectively. Although 129 and 80 retrieval runs were submitted, the same number of runs but not all from each group were pooled for evaluation. Pooled retrieval runs were scanned to a pool depth of 100 for TREC 8 and 50 for TREC GOV2 datasets to generate a set of documents for assessment for the corresponding topics.

3.3.3 Experimental Setup

For recent IR datasets pool depths lower than 100 have been used [Büttcher et al., 2006; Clarke et al., 2011; 2012; Collins-Thompson et al., 2013]. Therefore, for both datasets a pool depth of 50 is considered in the experiments ($q = 50$). Only retrieval runs with complete relevance assessments above the pool depth are considered. Only relevance assessments pooled by the selected retrieval runs are used. These filtered datasets are referred to by its original name henceforth. It was found that 53.5% and 48.8% of the runs fulfilled the above criteria with a $7.6 : 1$ and $2.5 : 1$ ratio of automatic to manual runs for the respective datasets. Out of the total relevance judgements made, 27.0% and 11.2% of the documents were uniquely pooled by manual runs of which 19.6% and 17.6% were relevant respectively.

The aim here is to locate relevant documents that would normally be found only by including manual runs in the pooling process. Documents in MRJ are used as surrogates of relevance assessments for the documents placed in the second pool by the approaches. Because the documents in the second pool are ranked by each of the approaches, the quality of the pool can be measured using a retrieval effectiveness metric such as MAP.

The Kendall's τ correlation and the AP correlation are used to compute the agreement between the two mean rankings of runs evaluated with full relevance assessments and the relevance assessments generated from the union of the first and the second pools formed by the approaches. Using a convention from Voorhees [Voorhees, 2000], if the Kendall's τ correlation is greater than 0.9, the rankings are considered equivalent. The same convention is later followed by Carterette et al. [2008], and Carterette and Soboroff [2010].

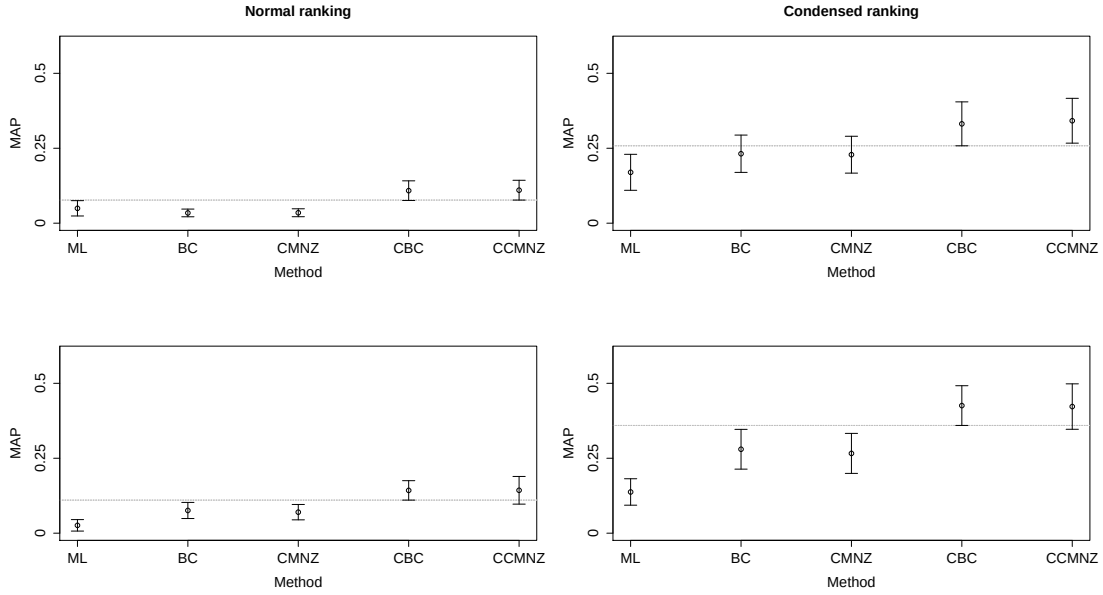


Figure 3.3: Retrieval effectiveness and 95% *confidence interval on finding relevant documents in MRJ with traditional evaluation (left) and using a condensed ranking (right) for TREC topics* 401-450 *on the TREC 8 dataset (top) and TREC topics* 801–850 *on the TREC GOV2 dataset (bottom).*

3.3.4 Results

An evaluation with MAP using the original as well as a condensed ranking (unjudged documents removed) for both datasets are presented in Figure 3.3. The combined approaches perform better than all other approaches for traditional evaluation. For instance, CBC is significantly better than ML, its counterpart fusion method BC, and CMNZ. Note that the relatively low reported effectiveness in Figure 3.3 for traditional evaluation is largely a byproduct of evaluating only unique relevant documents pooled by the manual runs and not the entire pool.

However, no claims can be made about the real effectiveness of these approaches since a large portion of retrieved documents using these methods remain unjudged. This is illustrated in Figure 3.4. The ML method retrieves a much larger portion of unjudged documents compared to other two approaches. In fact, 97% and 98% of the top-200 documents returned across all 50 topics using only machine learning for TREC 8 and TREC GOV2 datasets are currently unjudged. Therefore, each of these approaches are evaluated using a condensed ranking, for which effectiveness is shown in Figure 3.3 (right). Effectiveness is overestimated with condensed rankings when a large portion of documents in the ranked result lists are unjudged [Sakai, 2008a;b; 2009]. Combined approaches

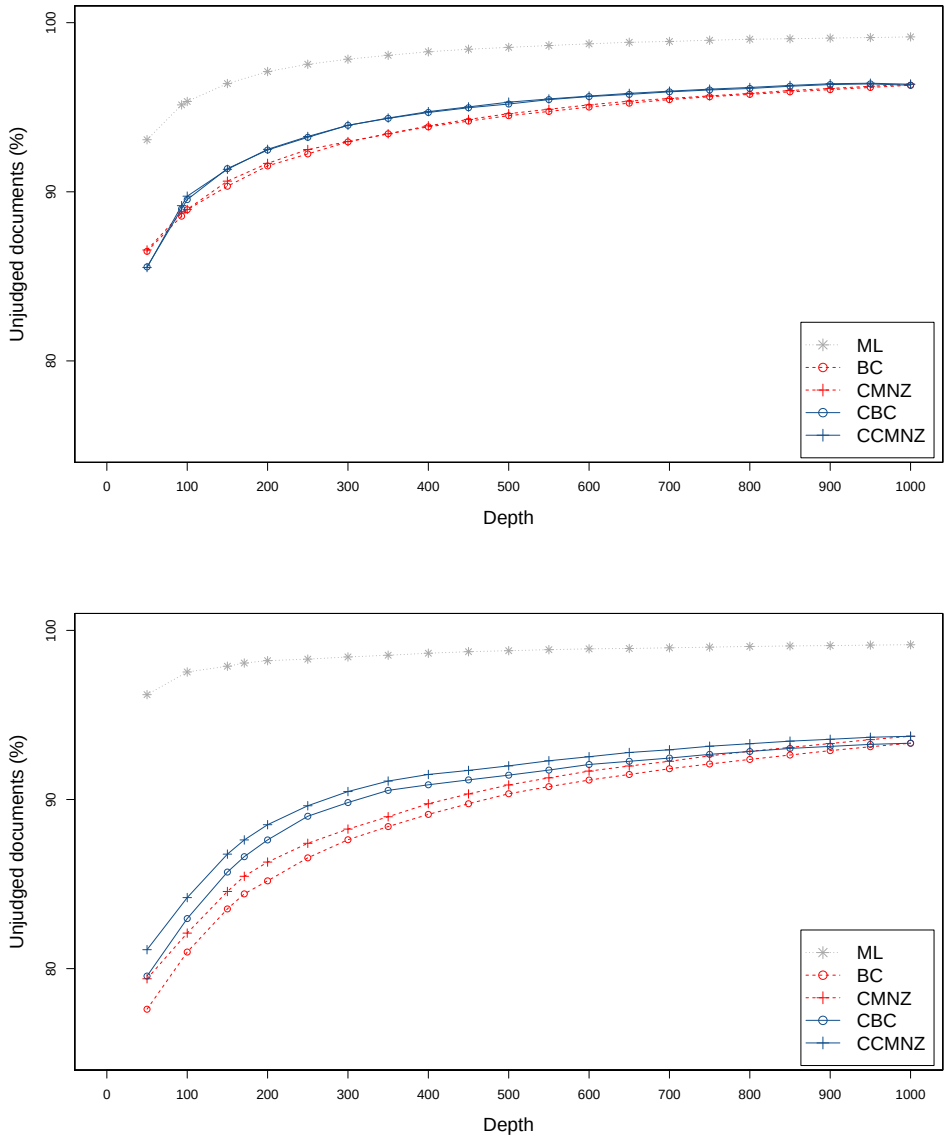


Figure 3.4: Percentage of unjudged documents found in the top-(κ) of the proposed rankings for TREC topics 401-450 *on the TREC 8 dataset (top) and TREC topics* 801–850 *on the TREC GOV2 dataset (bottom).*

Depth (κ)	ML	BC	CMNZ	CBC	CCMNZ
50	10.40	13.02	12.63	$23.61^{\bullet\dagger}$	$24.90^{\bullet\ddagger}$
93 ✠	12.64	20.68	20.12	$30.26^{\bullet}$	$30.73^{\bullet\dagger}$
100	12.74	21.24	20.61	$31.11^{\bullet}$	$31.13^{\bullet\dagger}$
150	13.87	27.10	26.83	$35.61^{\bullet}$	$36.65^{\bullet}$
200	15.04	31.05	30.77	$38.86^{\bullet}$	$39.52^{\bullet}$

Depth (κ)	ML	BC	CMNZ	CBC	CCMNZ
50	5.41	15.04	13.17	$29.34^{\bullet\ddagger}$	$28.18^{\bullet\ddagger}$
100	6.45	27.49	24.14	$41.97^{\bullet\ddagger}$	$38.96^{\bullet\ddagger}$
150	7.58	32.71	31.13	$49.29^{\bullet\ddagger}$	$44.89^{\bullet\dagger}$
171 ✠	7.83	34.86	32.67	$50.14^{\bullet\ddagger}$	$46.01^{\bullet\dagger}$
200	8.48	37.28	34.66	$51.35^{\bullet\dagger}$	$47.02^{\bullet\dagger}$

Table 3.3: Percentage of MRJ documents found per topic in the top-(κ) of the proposed rankings for TREC topics 401-450 *on the TREC 8 dataset (top) and TREC topics* 801–850 *on the TREC GOV2 dataset (bottom).* ✠ *implies a similar assessment effort to a traditional pooling method. Combined approaches are tested for significance. A* $^{\bullet}$ *implies a significant improvement at* $p < 0.01$ *compared to ML. Similarly, a* † *and* ‡ *implies a significant improvement at* $p < 0.05$ *and* $p < 0.01$ *compared to the base fusion method.*

reported a higher effectiveness than other approaches on a condensed ranking. Surprisingly, effectiveness for the ML method on a condensed ranking is worse than the other approaches, and somewhat similar to the effectiveness of combined approaches with a normal ranking. This suggests that using only the machine learning approach for locating relevant documents from the dataset is not effective. However, a definitive assessment for ML method can only be made by judging the ranked result list.

Recall that the traditional evaluation underestimates effectiveness when retrieved documents are unjudged. Therefore, the higher effectiveness for combined approaches with a condensed ranking compared to a traditional evaluation indicate that there could be more relevant documents that are not found by the existing approaches to pooling.

In Table 3.3 the proportion of MRJ documents per topic that were found to be relevant in the second pool are analysed. Again a similar trend of differences are seen, but with significant improvements up to a depth of $\kappa = 200$ for combined methods.

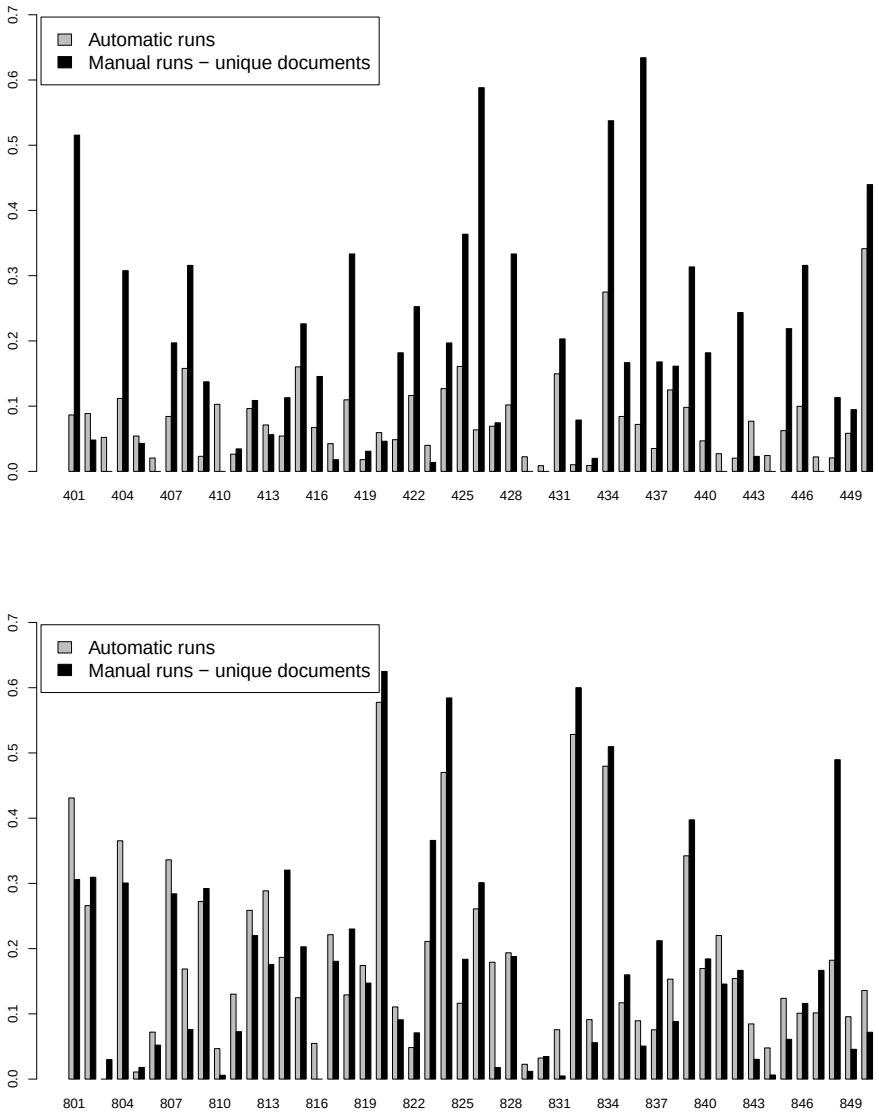


Figure 3.5: Proportion of relevant documents added by automatic runs and exclusively pooled by manual runs out of total documents pooled by the corresponding type of runs for each topic in topics 401-450 *on the TREC 8 dataset (top) and TREC topics* 801–850 *on the TREC GOV2 dataset (bottom).*

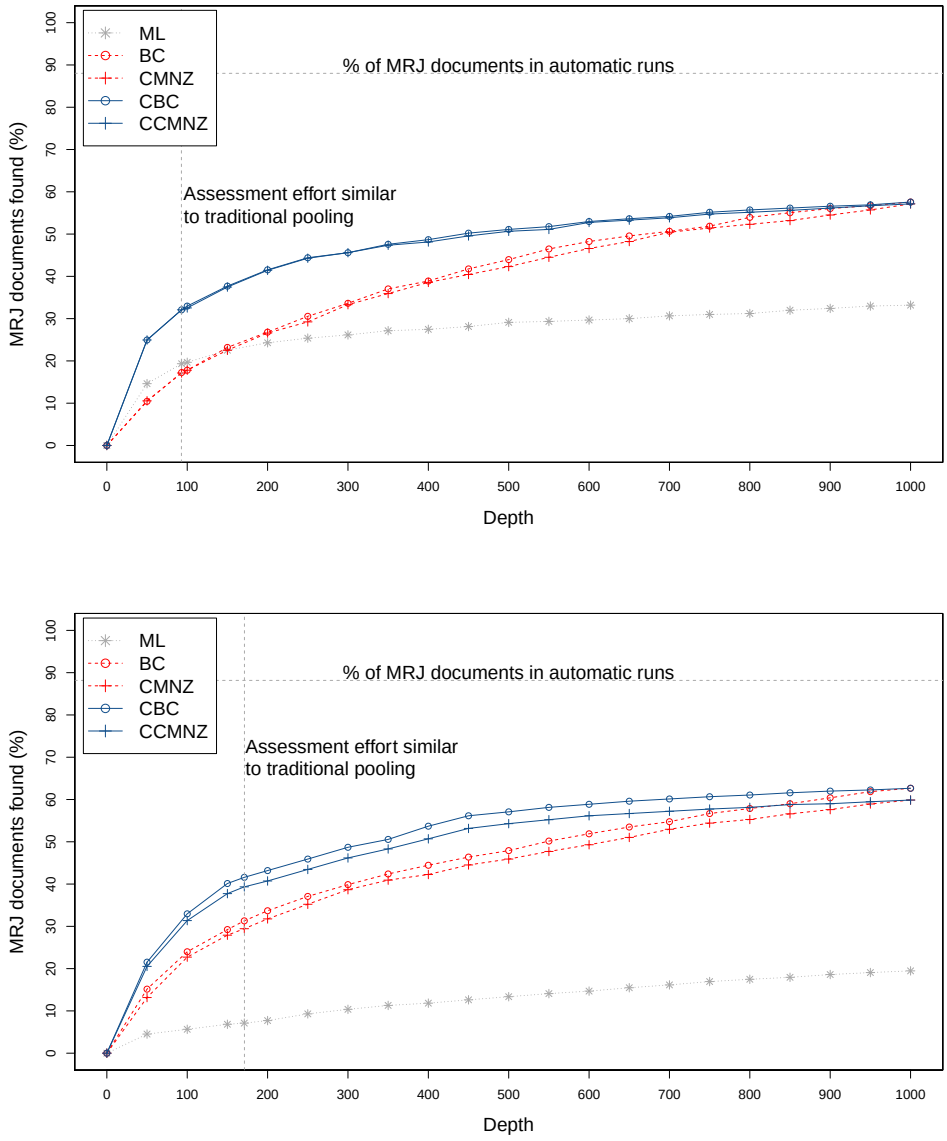


Figure 3.6: Percentage of MRJ documents found in the top-(κ) of the proposed rankings for TREC topics 401-450 *on the TREC 8 dataset (top) and TREC topics* 801–850 *on the TREC GOV2 dataset (bottom).*

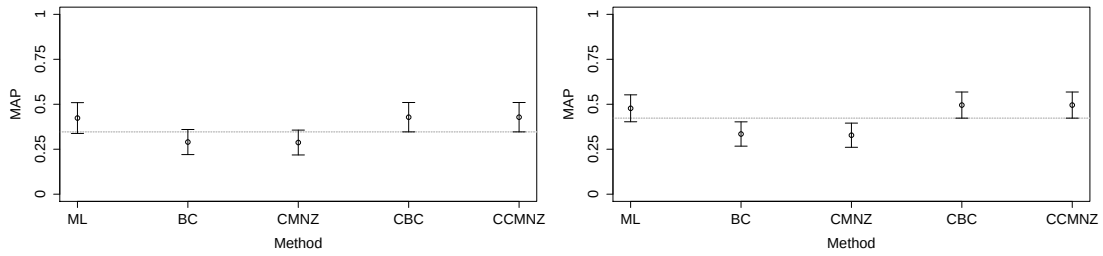


*Figure 3.7: Just considering the documents in MRJ, how effective are ranking algorithms (*MAP*) on retrieving relevant documents for TREC topics* 401-450 *from the TREC 8 dataset (left) and for TREC topics* 801–850 *from the TREC GOV2 dataset (right)?*

Discussion

The proportion of relevant documents that are pooled by automatic and manual runs out of the total pooled by each type of runs per topic is shown in Figure 3.5. As shown in the plot, manual runs provide a rich source of relevant documents for judging. If documents exclusively pooled by manual runs were not judged (i.e. no MRJ), the effectiveness of IR systems producing results similar to manual runs would be underestimated and the improvements would go unnoticed.

However, this still provides no guarantee that manual runs alone are a sufficient surrogate for all future IR systems. In fact, increased effectiveness for fusion and combined approaches on a condensed ranking rather than using a traditional ranking suggests the possibility of finding more relevant document not found by either automatic or manual runs using current pooling strategies. Nonetheless, manual runs still play a critical role in improving the reusability of test collections.

In Figure 3.6, the total proportion of relevant MRJ documents found by each of these approaches are analysed. The majority of documents uniquely pooled with manual runs (MRJ) are also retrieved by automatic runs. However, they do not appear in the first pool as they are not ranked highly enough by automatic runs alone. In fact, 88.02% and 88.17% of the documents judged as relevant that are uniquely pooled by manual runs on TREC 8 and TREC GOV2 datasets could be found in the first pool if a pool depth of 1000 was used. This upper threshold is the maximum proportion of relevant MRJ documents that can be found using the proposed fusion and combined methods and represented in the plots as a dashed horizontal line. The upper threshold for the maximum proportion of relevant MRJ documents that can be found by each of the combined methods is the maximum proportion of relevant MRJ documents found by the base fusion approach. The combined approaches effectively rerank the results produced by the fusion approaches.

Metric	ML	BC	CMNZ	CBC	CCMNZ
P@10	0.0500	0.3375$^\bullet$	0.3250$^\bullet$	0.4187$^\bullet$	0.4000$^\bullet$
P@20	0.0406	0.3094$^\bullet$	0.3156$^\bullet$	0.4000$^\bullet$	0.3875$^\bullet$

Table 3.4: Effectiveness (P@10 and P@20) for each ranking approach when complete judgements are manually assessed up to a depth of 20 *for the first* 16 *topics in the TREC GOV2 dataset. A* $^\bullet$ *implies a significant difference at* $p < 0.01$ *compared to ML.*

Missing judgements for a large portion of the ranked lists from the proposed methods is one potential reason for the low retrieval effectiveness of the proposed approaches. The effectiveness for all approaches are higher when using a condensed ranking rather than a traditional evaluation, but rankings are of varying length. Therefore, retrieval effectiveness on retrieving documents from MRJ is computed in Figure 3.7. Note that the first pool and the ranking functions remains the same. The ML method now reranks the top-q unique documents ranked by manual runs. The ranking produced by ML shows a considerable improvement. The combined method is more effective than fusion methods for MAP on both datasets, and the improvement is significant on the TREC GOV2 dataset. Recall that more unjudged relevant documents exist in larger test collections. Hence there is more room for improvement in the TREC GOV2 collection compared to the TREC-8 collection. Reranking a carefully retrieved subset of documents for topics with ML is an effective approach to locate new documents to be pooled and judged.

The true effectiveness of the above methods cannot be estimated without judging all of the unjudged documents for each method. Therefore, the top 20 documents produced by each method for the first 16 topics in the TREC GOV2 dataset were manually judged, and results are shown in Table 3.4. The ML method is significantly worse than other methods. The ML method is trained using ARJ documents. However, using only the ML method also ranks documents that contain similar terms to relevant documents in ARJ but not the terms related to the topic highly, and therefore the ML method alone (without query dependent features) is not effective. The above weakness is overcome when ranking is limited to documents that are already ranked by IR systems. This supports prior observed results. That is, effectiveness with ML method alone is low as in Figure 3.3 when entire document corpus is ranked. However, as observed in Figure 3.7 just ML method is highly effective when documents retrieved by manual runs (MRJ) are ranked. Therefore, the combined method is more effective than any of the other methods analysed.

For the rest of the discussion, the Combined - Borda Count method is used. Whenever a new approach for pool composition is proposed, it is vital to quantify how well the approach ranks IR

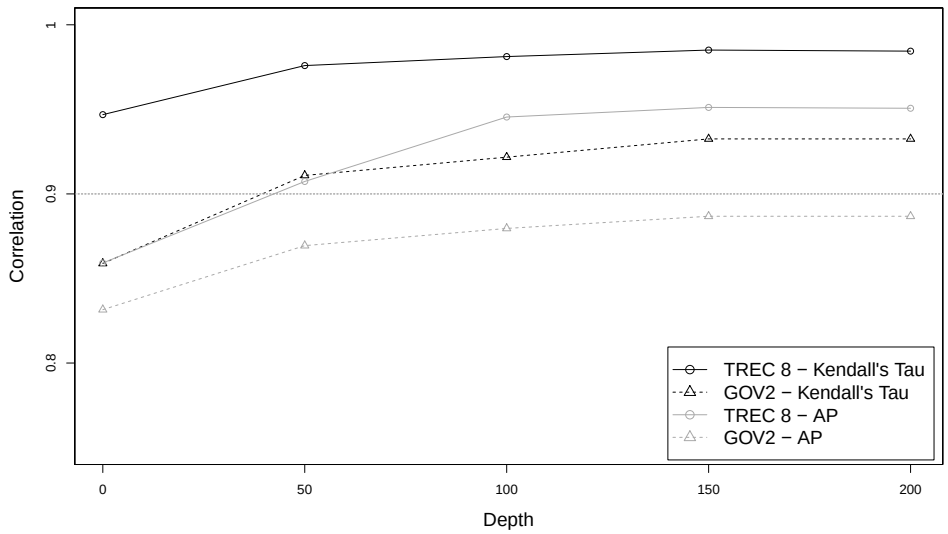


*Figure 3.8: Kendall's τ and AP correlation of IR system rankings for varying depths of assessing documents with combined method (*cbc*) on the TREC 8 dataset with TREC topics* 401–450 *and the TREC GOV2 dataset with TREC topics* 801–850*.*

systems compared to the original method. A Kendall's τ and AP ranking correlation for varying depths of assessing documents with the Combined - Borda Count approach are shown in Figure 3.8. Manual runs are viewed as novel approaches to retrieval. The Kendall's τ correlation for MAP is above 0.9 beyond a depth of 100. A budget similar to original assessment permits processing up to a depth of 93 and 171 documents for TREC 8 and TREC GOV2 datasets respectively. However, the AP correlation is lower than the Kendall's τ correlation. The lower AP correlation compared to Kendall's τ correlation indicates top results are affected more. Yet the combined approach provides a valid method for improving reusability of test collections in the absence of manual runs.

The data available to train a machine learning ranker is less with lower pool depths. As a consequence, the classifiers can also be less effective. In Figure 3.9 we investigate how shallow the pool depth can be before the combined method is equally or less effective than the simple fusion methods. As shown in the graphs, the method is more effective than the fusion only method (BC) when the pool depth is above 10 or 20 for the TREC-8 and the TREC GOV2 datasets respectively.

3.3.5 Summary

In this chapter, a methodology for building reusable evaluation pools in the absence of manual

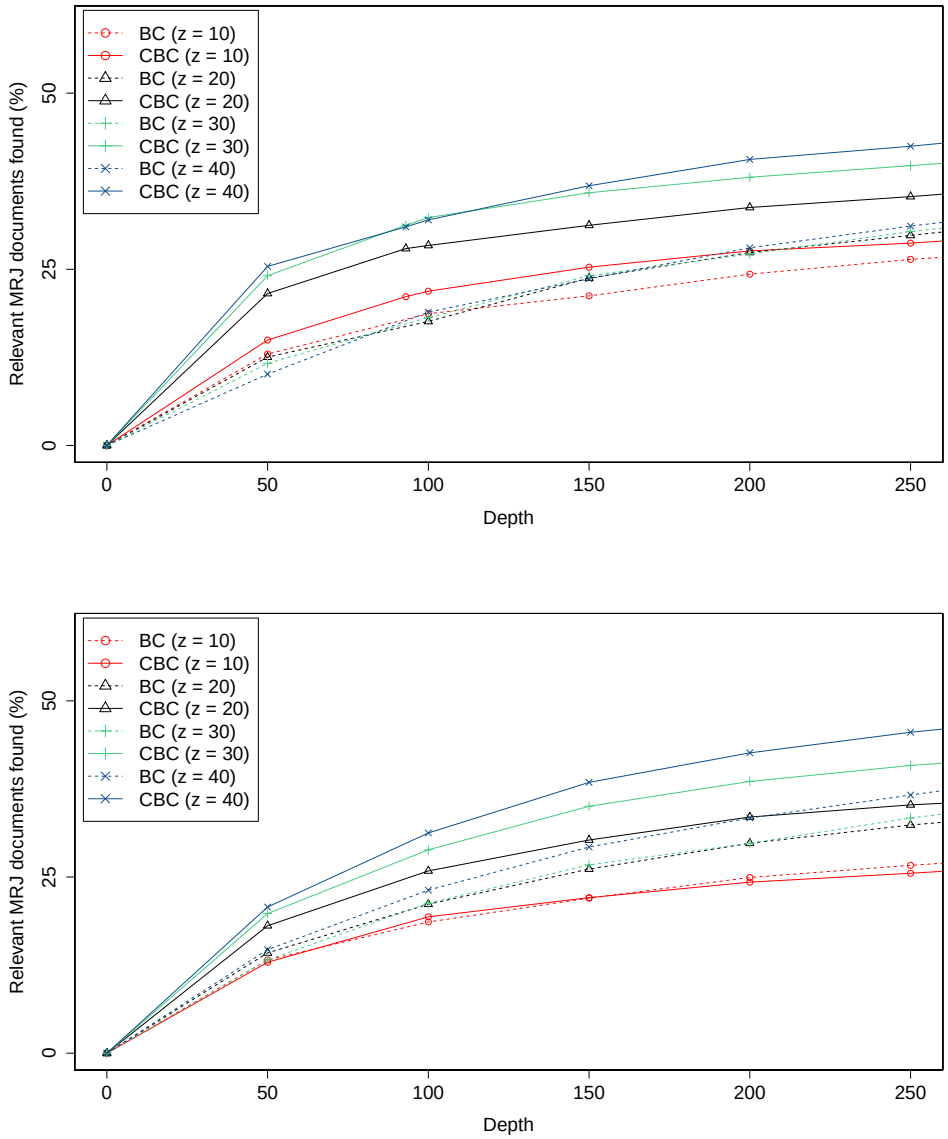


Figure 3.9: Percentage of relevant MRJ documents found out of total relevant MRJ documents found with a pool depth of 50 *with varying pool depths for TREC topics* 401-450 *on the TREC 8 dataset (left) and TREC topics* 801-850 *on the TREC GOV2 dataset (right).*

runs is presented. A large portion of relevant documents that are uniquely added by manual runs are also retrieved but not pooled by automatic runs. Taking advantage of the above fact, the approach discovers a considerable proportion of relevant documents that were previously only found by manual runs. The approach demonstrates the potential of finding relevant documents that are not possible using the current pooling approaches. However, the true efficacy of the approach cannot be properly assessed until all of the newly retrieved documents are judged. By judging top ranked documents for few topics, we demonstrate using ML method (without query-dependent features) is not effective and combined methods are the most effective. Topics containing more relevant documents could also be judged to different depths to maximise the effectiveness of the proposed approaches, but was not explored here. The initial results are promising as the method is already able to achieve a closer IR system ranking to previous approaches which depended heavily on manual runs to add the necessary diversity to the assessment pool. The combined method is more effective than simple fusion methods on finding relevant MRJ documents even when the pool depth is shallow as 20.

Abbreviations and Symbols used in Chapter 4

Notation	Description
c	A cluster centroid.
CSI	Central sample index.
G	Average of k centroid models.
k	Number of clusters.
l	An index into IR systems.
M	Number of sampled IR system instances.
m	An index into IR system instances.
S	An IR system.
s	An IR system instance produced using a non-deterministic IR system.

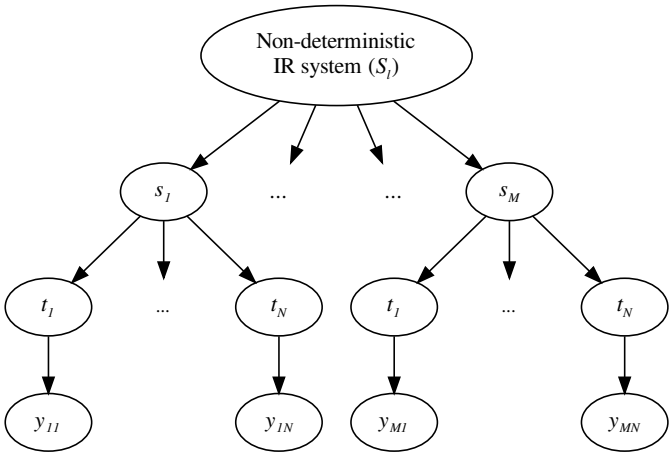
Chapter 4

Evaluating Non-deterministic Retrieval Systems

The output of IR systems can be non-deterministic. This can be due to use of sampling in randomized algorithms [Aly et al., 2013; Callan et al., 1999; Callan and Connell, 2001; Kulkarni and Callan, 2010; Si and Callan, 2003; 2004; 2005; Shokouhi, 2007; Thomas and Shokouhi, 2009], or training based on the unpredictable inputs of users [Finkelstein et al., 2001; Lawrence, 2000; Liu et al., 2004]. Evaluating the effectiveness of such IR systems can be difficult because each run of the IR system will produce a different IR system instance, hence different effectiveness scores. Figure 4.1 illustrates this situation. How can the effectiveness of an IR system be measured if each run might produce a different output?

An evaluation based on a single instance of an IR system may produce results which might change an experimental conclusion of whether one IR system is better than another. The obvious solution is to generate several IR system instances and make statistically grounded inferences about the overall average effectiveness.

Experiments in IR add another layer of complexity to the problem as retrieval effectiveness varies by topic [Voorhees and Buckley, 2002]. Recall that the effectiveness of an IR system instance is characterized and compared using average effectiveness under whatever evaluation metric is employed across a set of topics. Differences are tested for statistical significance across a hypothetical population of topics using a significance test. However, standard significance tests only support one source of variability, in this instance the choice of topics. The use of non-deterministic IR systems introduces an additional dimension of variability. Namely, is one IR system significantly better than



S_l Non-deterministic IR system l,

s_m IR system instance m produced using non-deterministic IR system l, where $m = 1, ..., M$

t_n $n{\cdot}th$ topic for evaluation, where $n = 1, ..., N$.

y_{mn} Effectiveness for topic n on IR system instance m.

Figure 4.1: Variation in effectiveness for a non-deterministic IR system along two dimensions (IR system instances and topics), with effectiveness for topics grouped within IR system instances.

another, averaged across all topics as well as all possible IR system instances?

In order to understand the uncertainty due to variability in more than one dimension, consider the following simulation. The effectiveness of each topic on each non-deterministic IR system instance is considered to be composed of two effects. First, the "topical effect" between 0 and 1 is randomly picked from a uniform distribution for each topic. Next, the effect due to non-determinism in the system is realised from a normal distribution $\mathcal{N}(\mu, \sigma^2)$, and capped between 0 and 1. Now assume that the simulated effectiveness for each topic on each system instance is the *Euclidean distance (L_2 norm)*[1] between the two effects, normalized to a value between 0 and 1. Consider another synthetic system with a deterministic output, whose effectiveness for each topic between 0 and 1 is randomly picked from a uniform distribution. For the simulation, two sets of 50 topical effects, one for each system being compared and 100 system instance effects for a given μ and σ are sampled from the respective distributions. The effectiveness for each system instance is then compared with the deterministic system using a standard t-test, and the mean of the significant differences over 100

[1]The Euclidean distance (L_2 norm) is the square root of the sum of the squares of each effect.

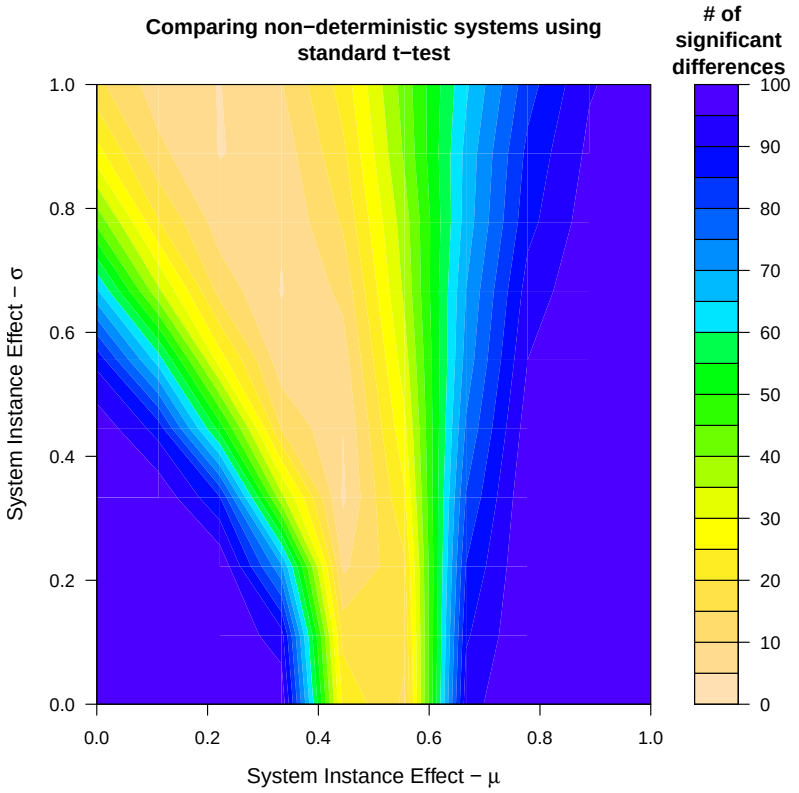


Figure 4.2: Results for comparing simulated non-deterministic IR system instances with a deterministic IR system. The system instance effect is randomly sampled from a normal distribution for each IR system instance. The topic effect is randomly sampled from a uniform distribution between 0 *and* 1*. The effectiveness for a topic-system instance pair is the Euclidean distance of respective component effects normalized to a value between* 0 *and* 1*. Each system instance is compared with the deterministic system just having the topic effect using a standard* t*-test, and the significant differences are noted at a* $p < 0.05$*.*

repetitions of the above process are noted as μ and σ are varied. This is equivalent to repeatedly comparing an instance of a non-deterministic IR system having two dimensional variance with a deterministic IR system. The results are illustrated in Figure 4.2. The expected effectiveness for the deterministic system is 0.5. Similarly, the expected effectiveness for the non-deterministic IR system is 0.5 when μ is 0.5. Therefore, the majority of comparisons show no significant difference when μ is 0.5 with a small σ. When the standard deviation is increased, the majority of the comparisons with a μ less than 0.5 are not significantly different. However, a comprehensive analysis of the impact on a standard significance test due to variation in system instance effect for a non-deterministic IR system is not possible with the above simple simulation. The take home message from the simulation

is that all comparisons derive the same conclusion when number of significant differences are 0 or 100. An intermediate shade of color implies that conclusions derived using some instances contradict the conclusions from the rest of the instances when using the same non-deterministic IR system. For example, 50% of the comparisons show no significant difference at the center of the color scale, while the rest imply the opposite. So the gradual hill climbing pattern of the graph implies that experimental conclusions derived with certain system instances can be contradicted by the others and exemplifies the pitfalls in using standard significance tests to compare non-deterministic IR systems.

In this chapter, possible solutions are investigated in the context of distributed information retrieval where sampling based algorithms are widely used to improve efficiency, and make the following contributions:

1. A pair of methodologies, one using bootstrapping (Section 4.1.1) and the other based on multivariate linear modelling (Section 4.1.2), are proposed to solve the two-dimensional significance testing problem for comparing a non-deterministic IR system with a deterministic IR system. Both methods result in equivalent inferences (Section 4.4.2).

2. The multivariate linear modelling approach is extended to compare two non-deterministic IR systems (Section 4.2).

3. The properties of the proposed solutions are explored on a case study of common sampling based algorithms – shard construction and centralized resource allocation in distributed IR [Si and Callan, 2003; Kulkarni and Callan, 2010]. The variability that can occur in this environment is examined while observing that an apparently significant result on one instance of a sample based algorithm can be contradicted by another. The use of the proposed two-dimensional significance testing methods to handle the variability are demonstrated while providing sound statistical inferences (Section 4.4).

4. When a new non-deterministic IR system is evaluated for efficiency, a common question of interest is: Does the new method achieve a greater or equivalent effectiveness to the old one with less effort (that is, greater efficiency)? Using a two dimensional evaluation, a methodology for comparing a non-deterministic IR system with either a deterministic or another non-deterministic IR system for greater or equivalent effectiveness is shown in Section 4.4.2.

In Chapter 2, previous work on evaluating IR systems with repeated observations for each system-topic pair were discussed. Both Carterette et al. [2011] and Robertson and Kanoulas [2012] used a multivariate linear model to compare IR systems. Effectiveness for each system-topic pair were

repeatedly observed with different users in the former, and over different document sets in the latter. Subsequently, how the bootstrap method can be used to construct confidence intervals, when repeated observations are encountered due to incomplete judgements is shown in Chapter 3. The focus here in this chapter is on the added dimension of variability introduced by using a sampling based algorithm instead of repeated observations on a topic-deterministic IR system pair. Hence, variability exists in two dimensions (IR system instances and topics) with one (topical variation) grouped within the other (IR system instances). This question is investigated next.

4.1 Deterministic:Non-deterministic Comparison

Two novel significance tests to compare a non-deterministic IR system with a deterministic IR system are proposed next. One solution is an extension of the standard bootstrap test, while the other is based on multivariate linear modelling.

4.1.1 Bootstrap Test

Algorithm 4 presents the bootstrap based approach for comparing a non-deterministic IR system with a deterministic IR system. The principle is the same in spirit as in the bootstrap Algorithm 1 defined in Section 2.4.3, except the additional complexity of comparing over M IR system instances of the non-deterministic IR system. Effectiveness scores on each IR system instance are paired up by topic with effectiveness scores on the deterministic IR system. As effectiveness scores for the same set of topics are now measured on many IR system instances, the t-statistic for the original effectiveness score deltas $t(z)$ is computed to the mean score delta for each topic across M IR system instances. BS bootstrap observations are drawn from the topic score deltas observed for each IR system instance, and recentered by deducting the mean over the BS bootstrap observations. Evidence in favour of the null hypothesis is found each time the absolute value of the t-statistic for a recentered bootstrap resample $t(z_j^{*k})$ is greater than or equal to the absolute value of $t(z)$. The probability of such evidence is the fraction of evidence gathered over $BS \times M$ comparisons and provides the p value for the test.

ALGORITHM 4 Bootstrap algorithm for comparing a non-deterministic IR system with a deterministic IR system.

$BS \leftarrow$ Number of bootstrap samples;
$N \leftarrow$ Number of topics;
$M \leftarrow$ Number of IR system instances;
$a_n^m \leftarrow$ Matrix of scores for $m{\cdot}th$ IR system instance of non-deterministic IR system a on $n{\cdot}th$ topic, where $m = 1, ..., M$ and $n = 1, ..., N$;
$b_n \leftarrow$ Vector of scores of deterministic IR system b on $n{\cdot}th$ topic, where $n = 1, ..., N$;

// Compute $t(z)$ using mean effectiveness for each topic.
$z \leftarrow []$;
for $n = 1$ *to* N **do**
    $mean_{a_n} \leftarrow \frac{\sum_{m=1}^{M} a_n^m}{M}$;
    $z[n] \leftarrow mean_{a_n} - b_n$;
end
$t(z) \leftarrow \overline{z}/\sqrt{\hat{\sigma}_z^2/N}$;

// Compute p value.
$count \leftarrow 0$;
for $m = 1$ *to* M **do**
    $z \leftarrow [a_1^m - b_1, ..., a_N^m - b_N]$;
    $bootstrap\_samples \leftarrow []$;
    $total\_shift \leftarrow 0$;
    for $i = 1$ *to* BS **do**
        $z_i^* \leftarrow$ Resample N items with replacement from z;
        $bootstrap\_samples.append(z_i^*)$;
        $total\_shift \leftarrow total\_shift + mean(z_i^*)$;
    end
    $shift \leftarrow total\_shift/BS$;
    for $i = 1$ *to* BS **do**
        $z_i^* \leftarrow bootstrap\_samples[i]$;
        for $n = 1$ *to* N **do**
            $z_i^*[n] \leftarrow z_i^*[n] - shift$;
        end
        $t(z_i^*) \leftarrow \overline{z_i^*}/\sqrt{\hat{\sigma}_{z_i^*}^2/N}$;
        if $abs(t(z_i^*)) \geq abs(t(z))$ **then**
            $count \leftarrow count + 1$;
        end
    end
end
$p\_value \leftarrow count/(BS \times M)$;

4.1.2 Multivariate Linear Model Test

The second approach for comparing a non-deterministic IR system with a deterministic IR system uses the following LME model:

$$y_{lmn} = \gamma + S_l + s_m + t_n + St_{ln} + \varepsilon_{lmn}. \tag{4.1}$$

Here y_{lmn} is the effectiveness score observed for $n{\cdot}th$ topic on the $m{\cdot}th$ IR system instance produced with the $l{\cdot}th$ non-deterministic IR system, and γ represents the model intercept. The factors S_l, s_m, and t_n represent effects due to $l{\cdot}th$ IR system, $m{\cdot}th$ IR system instance, and $n{\cdot}th$ topic respectively. The system-topic interaction effect is captured by St_{ln}. The unallocated portion of effectiveness y_{lmn} is what resides in ε_{lmn}. The IR system effect (S_l) is fixed in the above model where topics (t_n), IR system instances (s_m) and IR system-topic interaction (St_{ln}) provide the non-deterministic effects.

The data for the above model consists of effectiveness (y), and three other factor variables for IR system (S), IR system instance (s) and topic (t). Two factors are *crossed* when each level of one factor occurs in every level of another factor, and *nested* if the levels of one factor occurring within the levels of another factor are different. Crossed factors produce an interactive effect in the presence of repeated observations. The effectiveness scores for the same set of topics is measured on each non-deterministic IR system instance and the deterministic IR system. Hence, a crossed design can be used whereby each level of the IR system instance factor for the deterministic IR system is a replicate. In such a model, the IR system and IR system instance factors are crossed with the factor topics which results in a system-topic interaction effect, but not a system instance-topic interaction effect as repeated observations are not available. The p value for the test is obtained using the t-statistic for the IR system factor. The degrees of freedom for the test are the number of observations less one.

4.2 Non-deterministic:Non-deterministic Comparison

Now the multivariate linear model test is extended to compare two non-deterministic IR systems. The equation for the model remains the same. The comparison is based on M IR system instances sampled from each non-deterministic IR system. These system instances are different from each other. Therefore, the IR system instance factor (s) naturally nests within the IR system factor (S), as opposed to the crossed design used when comparing with a deterministic IR system. Similar to

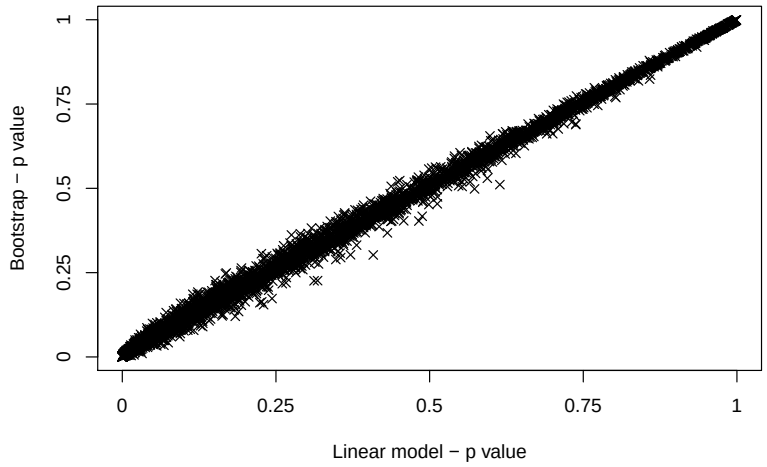


Figure 4.3: Correlation between bootstrap test and multivariate linear model test when comparing 5000 *simulated instance groups with simulated deterministic systems.*

before the IR system factor is crossed with topics, and causes a IR system-topic interaction effect. IR system instance-topic interaction effect does not occur as the factors are not crossed and have no repeated observations. The p value for the test is computed in the same fashion.

4.3 Simulation

Do the two proposed methods for the deterministic:non-deterministic IR system comparisons agree? Data is simulated for the study. For the deterministic IR system, the effectiveness score between 0 and 1 is randomly picked from a uniform distribution for each topic. For the non-deterministic IR system, data is generated in a manner similar to the previous simulation. That is, the topic effect between 0 and 1 is sampled from a uniform distribution while the system instance effects capped between 0 and 1 and sampled from a normal distribution $\mathcal{N}(\mu, \sigma^2)$. The mean μ and the variance σ^2 between 0 and 1 for a non-deterministic IR system is randomly generated. The effectiveness for a topic-system instance pair is the normalized *Euclidean distance* between two effects. Here, to generate effectiveness scores for many system instances for a non-deterministic IR system, the topic effect is held fixed, and repeatedly sample for the system instance effect. Effectiveness scores are generated for a set of 50 topics on 100 IR system instances for the non-deterministic IR system, and compared with the effectiveness scores for the deterministic IR system using the two proposed methods. Figure 4.3 illustrates the agreement

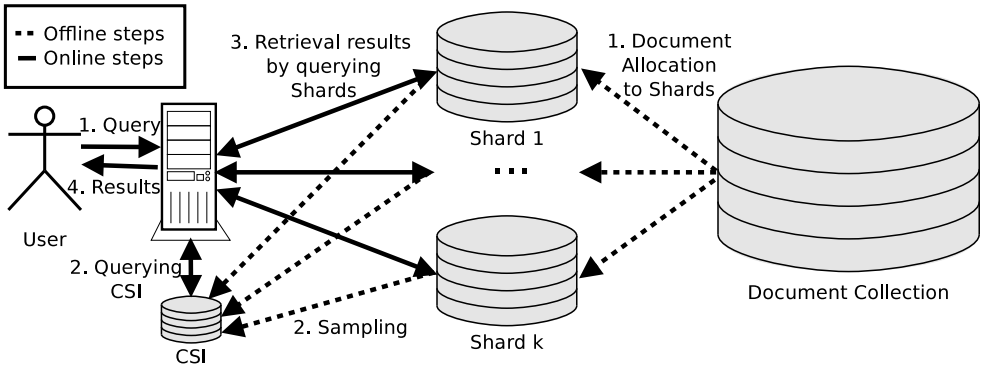


Figure 4.4: Offline formation and online search of shards. Each offline and online step is numbered in the order of occurrence.

between the two proposed methods for 5000 such comparisons. Both methods yield similar results, providing credence to the validity of the approaches.

4.4 Case Study

Now attention is turned to a concrete example. *Sharding* and *tiering* are well-known techniques to divide very large document collections. The technique is used in distributed IR to allocate shards of the collection to different nodes of a cluster [Callan, 2002]. In such a configuration, document search is performed across multiple indexes, one per node.

Efficiency can be improved in the overall system if the query is only sent to a subset of the indexes. However, there is a risk that relevant documents in unsearched indexes will be missed. The question becomes how many and which indexes should be queried without causing a measurable loss in retrieval effectiveness? In order to efficiently and effectively select the best subset of shards to visit for each query, a widely-used approach, for example ReDDE algorithm is to sample documents from each shard, and use this sampled surrogate collection (the central sample index or CSI) to represent the true collection statistics [Si and Callan, 2003]. Such an offline split of a document collection to shards, and realtime parallel search of shards is illustrated in Figure 4.4.

The simplest form of sharding splits the document collection between shards randomly. However, many shards must be searched to achieve a comparable effectiveness to a search over the entire collection, which has a deterministic effectiveness score because it has no sampling.

Recent research has focused on reducing the search cost per query by reordering the documents for each shard by topic or similarity [Kulkarni and Callan, 2010; Xu and Croft, 1999]. *Topical*

partitioning gathers similar documents into a single shard. A search over shards formed with such a policy achieves early precision (P@D, and NDCG@D) closer to exhaustive search with fewer shards searched.

The k-means clustering algorithm is one approach to form topically partitioned shards [Kulkarni and Callan, 2010; Xu and Croft, 1999]. Each document is represented in a sparse vector space as a *bag-of-words*, where each unique term is a dimension. The k-means clustering algorithm is not able to scale to the typical IR collection size and therefore sampling is necessary. Randomly selected seeds from the sampled documents form the initial cluster centroids. Documents are assigned to the closet cluster centroid, based on a *similarity metric*. After all documents are assigned, the center of the assigned documents for each cluster form the new cluster centroids. New random seeds from the sample of documents are used to replace cluster centroids without documents. The process is repeated for several iterations, each time improving the quality of clusters. Finally, all documents in the collection are assigned to the closest cluster centroid, thus forming k topically partitioned shards. Each time the algorithm is ran, a distinct set of shards are formed.

Use of different clustering algorithms or varying parameters of the above clustering process results in alternative topical partitioning schemes. Two such schemes used in experiments are outlined below. Viewing each document (d) and each cluster centroid (c) as distinct probability distributions, Kulkarni and Callan [2010] used *Kullback-Liebler divergence* for the similarity metric in the k-means clustering algorithm, as follows:

$$sim(d,c) = \sum_{w \in d \bigcap c} p_c(w) \log \frac{p_d(w)}{\lambda p_G(w)} + \sum_{w \in d \bigcap c} p_d(w) \log \frac{p_c(w)}{\lambda p_G(w)}, \tag{4.2}$$

where

$$p_d(w) = (1-\lambda) \times f(w,d) / \sum_{\acute{w}} f(\acute{w},d) + \lambda p_G(w), \tag{4.3}$$

$$p_c(w) = \frac{f(w,c)}{\sum_{\acute{w}} f(\acute{w},c)}. \tag{4.4}$$

Here, $p_G(w)$ is the probability of the term w appearing in the average of the k centroids, $f(w,v)$ is the frequency of term w appearing in term vector v, and λ is a smoothing parameter. The above approach is referred to as Scheme 1.

Moderately common terms are more important for clustering documents [Dhillon et al., 2001]. Therefore, the second approach uses an information theoretic criteria – *information gain (IG)* for

term selection.

$$\text{IG}(w) = \begin{cases} 0 & \text{if } p_w = 0 \\ -p_w \ \log_2 \ (p_w) \ - (1 - p_w) \ \log_2 \ (1 - p_w) & \text{otherwise.} \end{cases} \tag{4.5}$$

Here p_w is the chance of a randomly picked document containing the term w. The $\text{IG}(w)$ is small for terms with extreme (high or low) occurrence rates. Terms appearing in small number of documents are important for ranked document retrieval [Manning et al., 2008], and such terms can be identified with the inverse document frequency (IDF):

$$\text{IDF}(w) = log\left(\frac{|D|}{|d : w \in d|}\right), \tag{4.6}$$

where d is a document and $|D|$ is the number of documents in the collection. Top ranked terms based on a score given by:

$$\text{SC}(w) = \text{IG}(w) \times \text{IDF}(w) \tag{4.7}$$

are used to represent documents in a reduced vector space for clustering. To assess similarity between a document and a cluster centroid, the *cosine* similarity is used:

$$sim(d, c) = \frac{\sum_w f(w, d) \times f(w, c)}{\sqrt{\sum_w f(w, d)^2} \times \sqrt{\sum_w f(c, w)^2}} \tag{4.8}$$

This approach is referred to as Scheme 2, henceforth.

For the purpose of this study, the problem of determining the optimal CSI sample rate is reexamined, originally presented by Si and Callan [2003]. To select a subset of shards for a given query, the CSI is searched first. The proportion of documents from each shard in the CSI search results are used to rank shards for a given query. Therefore, the time spent on searching the CSI is a key factor determining query response time. The CSI search time is correlated to the sample rate used to construct the CSI and the query difficulty. The sample rate used for constructing the CSI must be sufficiently high to represent the shard in order to avoid poor retrieval effectiveness. A high sample rate can also minimize the likelihood of encountering out-of-vocabulary (OOV) terms in the mapping of the CSI to the shards. This is a classic effectiveness and efficiency tradeoff problem, whereby the best query response time is achieved when the sample rate is set to the smallest level that still achieves similar

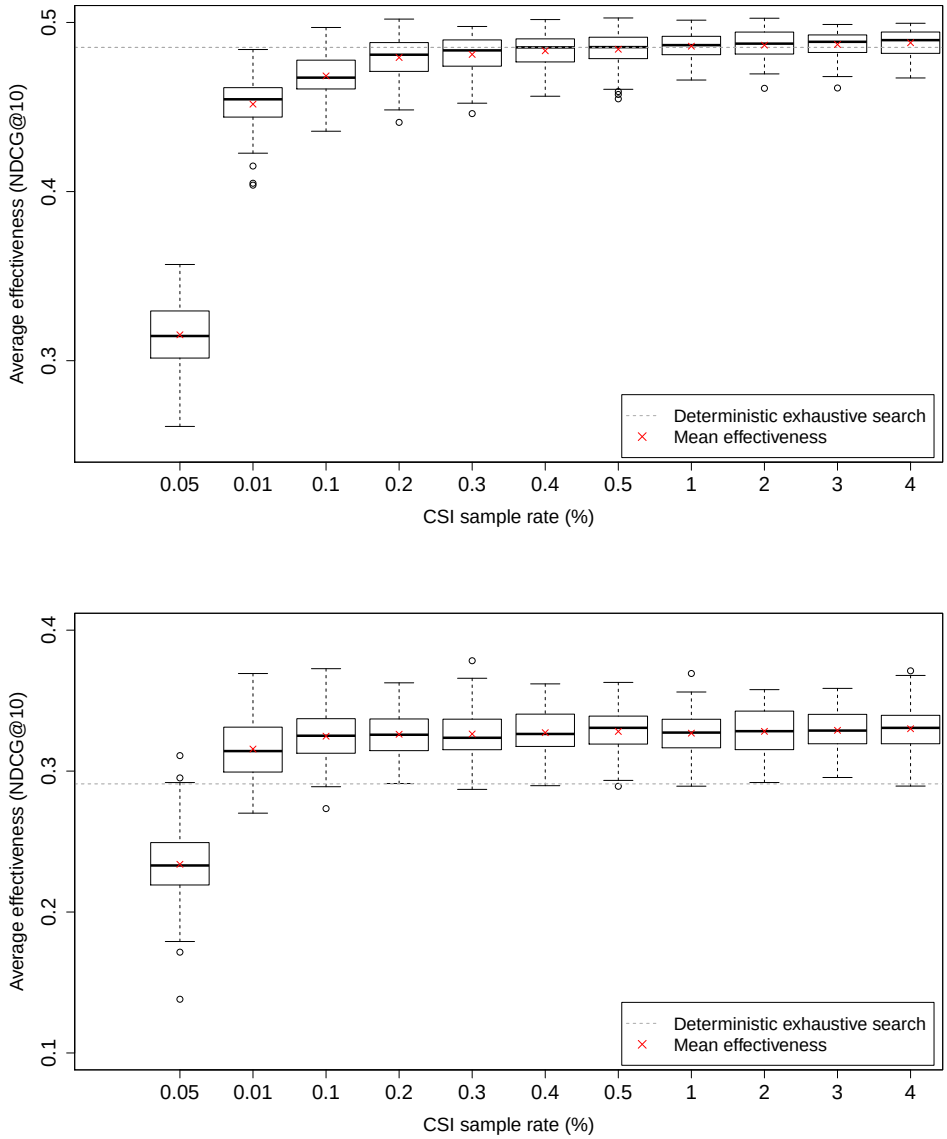


Figure 4.5: Variation in mean system instance effectiveness observed for the topical sharding Scheme 1 *on the TREC GOV2 dataset with TREC topics* 701–850 *(top) and on the ClueWeb'09B dataset with TREC'09 topics* 1–50 *(bottom). Each box in the box-and-whisker plot represents 25'th* (Q_1)*, 50'th* (Q_2) *and 75'th* (Q_3) *percentiles of the average system instance effectiveness, while the whiskers span* $1.5 \times (Q_3 - Q_1)$ *from the box. Any point outside interquartile range is considered an outlier.*

effectiveness to that resulting from exhaustive search.

4.4.1 Experimental Testbed

Experiments are separately performed on the TREC GOV2 dataset with TREC topics 701–850 and the ClueWeb'09B dataset using TREC'09 topics 1–50. MAP, NDCG@10, and P@10 are used to evaluate experiments. When not explicitly specified NDCG@10 is used.

On two independent 1% samples of the document collection, 5 topical partitions are formed for each sample and for each topical partitioning scheme. Here, a 1% sample is used following prior research by Kulkarni and Callan [2010]. Thus, 10 different topical partitions for each topical partitioning scheme are used. Further, 5 distinct random partitions are produced by non-deterministically allocating documents to shards. As with the original experiments [Kulkarni and Callan, 2010], 50 shards per instance for the TREC GOV2 dataset and 100 shards per instance for the ClueWeb'09B dataset were formed, and the full dependency model (FDM) is used to rank the queries [Metzler and Croft, 2005]. For topically partitioned shards, searching up to 5 shards produced early precision results equivalent to exhaustive search [Kulkarni and Callan, 2010]. Therefore, on topically partitioned shards up to the 5 ranked shards are searched for each query, except for MAP where a maximum of 10 and 15 shards are searched respectively for TREC GOV2 and ClueWeb'09B datasets. For each of the sharded version, 10 CSI instances are formed for each sample rate, giving 100 instances in total for each sample rate for each topical partitioning scheme and 50 instances in total for random shards. ReDDE algorithm is used to query shards.

4.4.2 Results

First the necessity for a two dimensional significance test is investigated. Different instances of a non-deterministic IR system may have varying effectiveness levels, and therefore evaluating effectiveness for just a single instance can be misleading. This is illustrated in Figure 4.5. The figure shows the variation in mean effectiveness for the 100 IR system instances constructed with the topical partitioning Scheme 1 for each CSI sample rate on the two datasets. Ignoring variation due to system instances by using a single instance for evaluation may therefore lead to inconsistent conclusions.

So, what is the impact of using conventional significance tests to evaluate non-deterministic IR systems? Each individual IR system instance for the topical partitioning Scheme 1 is compared with the exhaustive search using a paired t-test, and the results are illustrated in Figure 4.6. As can be seen on both datasets, the number of significant differences with a mean deterioration compared to

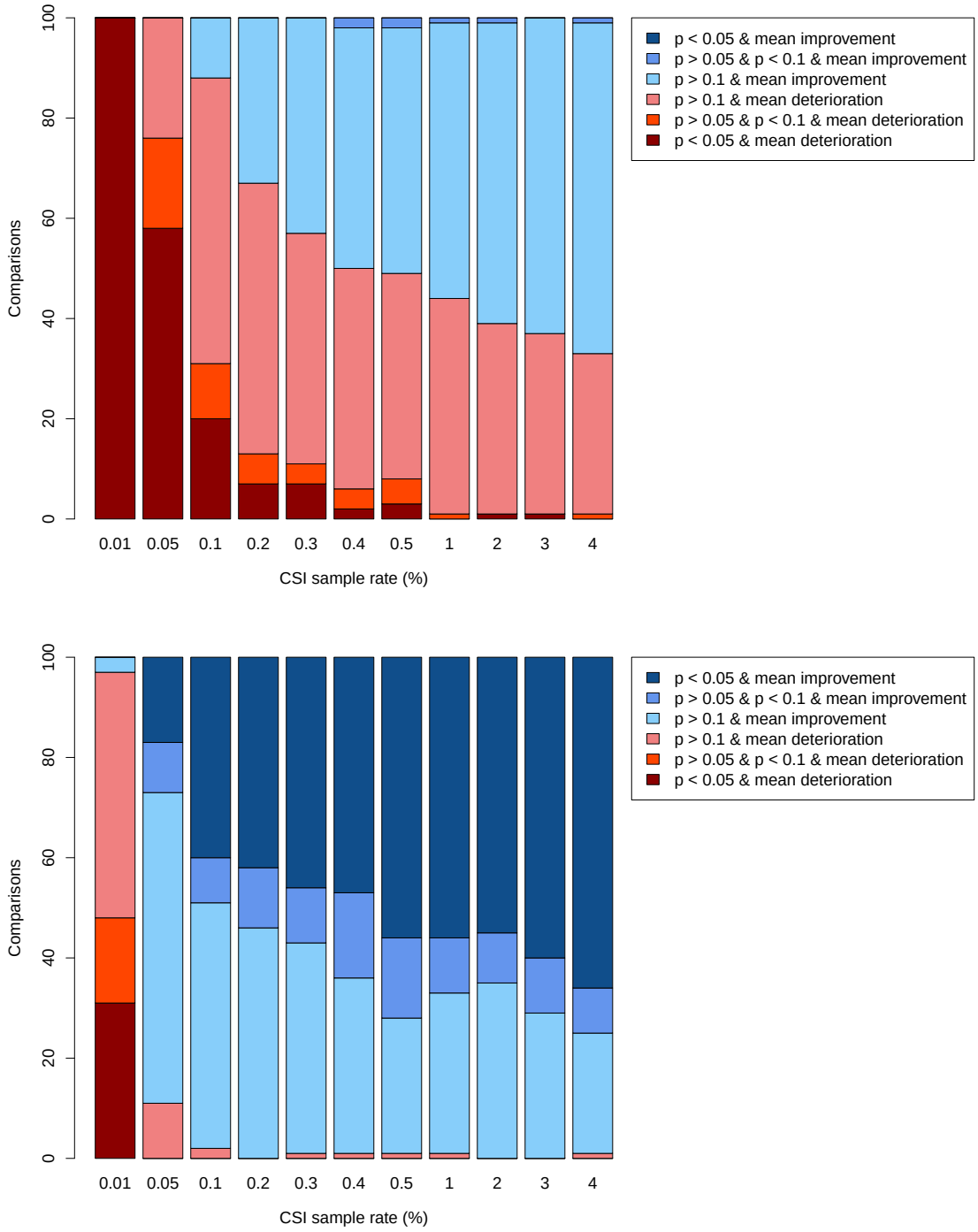


Figure 4.6: The distribution of p values for multiple paired t-tests, where each significance test compares effectiveness of an IR system instance derived using the sampling based topical partitioning Scheme 1 *for distributed IR system instance setup with exhaustive search on the TREC GOV2 dataset with TREC topics* 701–850 *(top) and on the ClueWeb'09B dataset with TREC'09 topics* 1–50 *(bottom).*

exhaustive search increases as the CSI sample rates are reduced. Similarly, the number of significant differences with a mean improvement compared to exhaustive search also increases along with increasing CSI sample rates. On both datasets some individual comparisons show a significant difference while the rest agree on no such difference for a given CSI sample rate. For example, on the TREC GOV2 dataset using a CSI sample rate of 0.5%, 3% of the comparisons show a mean deterioration and a significant difference with a p value less than 0.05, and 8% with a p value less than 0.1. For the same CSI sample rate, 2% show a mean improvement and significant difference over deterministic exhaustive search with a p value less than 0.1, and 90% show no significant difference. This exemplifies the potential of drawing an inaccurate conclusion, and the difficulty of confidently comparing a non-deterministic IR system using a single instance.

The mean effectiveness for each topic across a large number of non-deterministic IR system instances can be reported to reduce the likelihood of producing conflicting results. However such a comparison may not be fair, as variance due to non-determinism is not explicitly captured in such an evaluation framework. The variance in effectiveness for TREC topics 801 – 850 on the TREC GOV2 dataset and TREC'09 topics 1 – 50 on the ClueWeb'09B dataset across 100 topically partitioned distributed IR system instances for topical partitioning Scheme 1 are illustrated in Figure 4.7. While effectiveness for some topics is consistent, others are not. For example, on the TREC GOV2 dataset the TREC topic 826 "Florida Seminole Indians" demonstrate a high variation in effectiveness across system instances, while the topic 836 "illegal immigrant wages" in the same environment displays consistent effectiveness. Similarly, on the ClueWeb'09B dataset TREC'09 topic 1 "obama family tree" shows a high variation in effectiveness, while TREC'09 topic 11 "gmat prep classes" shows no variation in effectiveness. Many reasons can cause such variation in effectiveness. One is the representation of potentially relevant documents in the CSI. The effectiveness is normally high when possibly relevant documents are well represented in the CSI. Similarly, the effectiveness tends to be low when the CSI is under represented with such documents. Another is the split of probably relevant documents between shards. The effectiveness is high when they are concentrated to few shards, and low when they are split between many shards. However, in the presence of such variation, the proposed approaches outlined next can be used to minimise ambiguity for significance testing and help researchers draw more accurate conclusions.

Deterministic:Non-deterministic Comparison

Measuring the effectiveness of a non-deterministic search algorithm using a single instance can never be recommended due to the variance in p values. The variation in p value can somewhat be

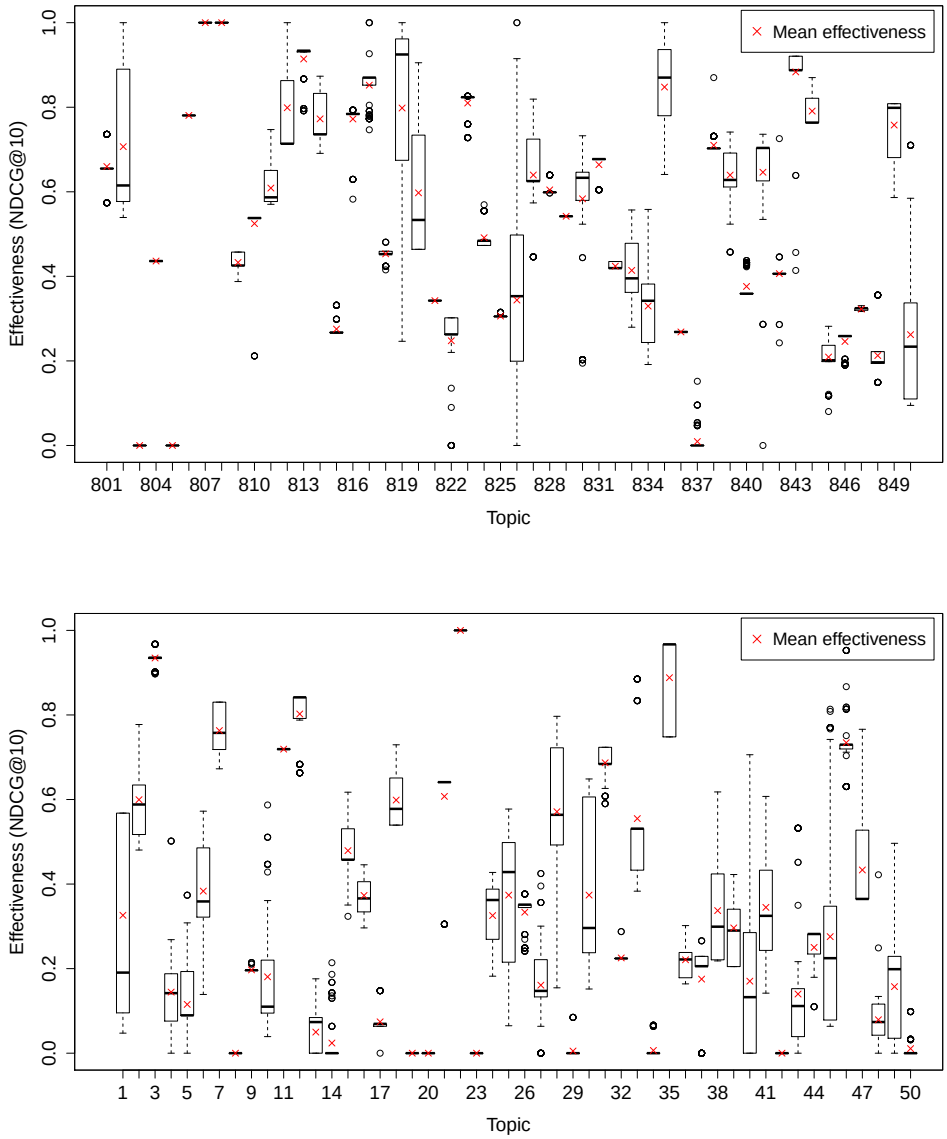


Figure 4.7: Topical variance for TREC topics 801 – 850 *on the TREC GOV2 dataset (top) and TREC'09 topics* 1 – 50 *on the ClueWeb'09B dataset (bottom) observed with the* 100 *IR system instances produced using the topical sharding* Scheme 1 *with a CSI sample rate of* 4%.

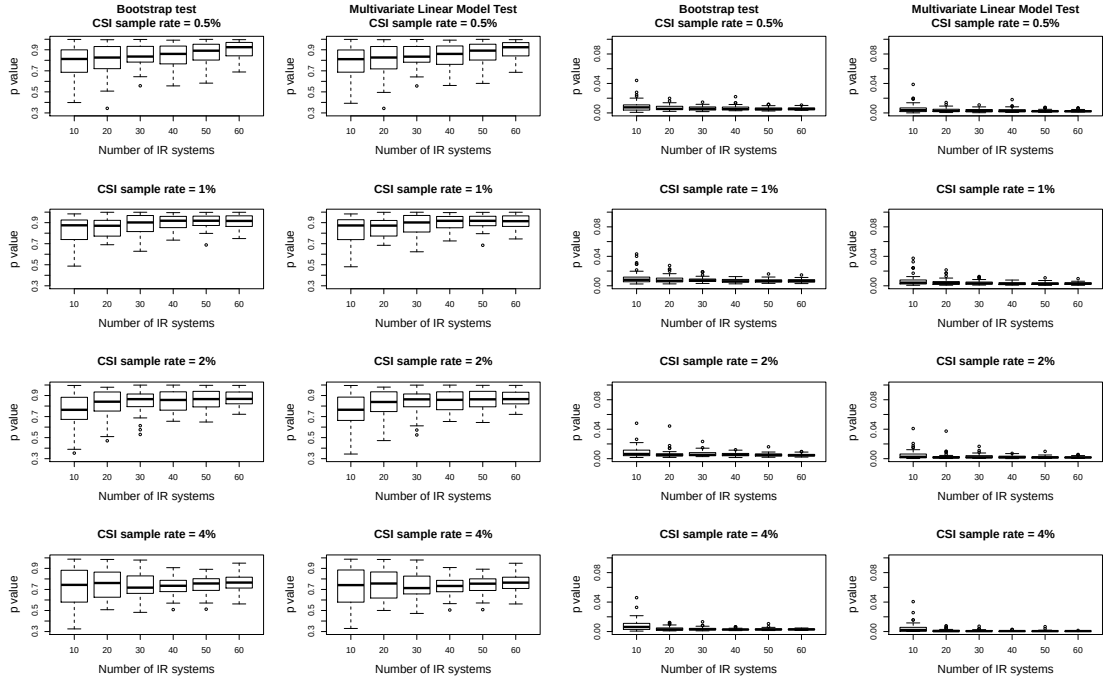


Figure 4.8: Variation in p values with proposed tests for varying sample sizes of IR system instances for topical partitioning Scheme 1 *at different sample rates for CSI with the* ReDDE *algorithm. The pools of* Scheme 1 *instances are compared with exhaustive search. Experiments are ran on the TREC GOV2 dataset with TREC topics* 701–850 *(left) and on the ClueWeb'09B dataset with TREC'09 topics* 1–50 *(right).*

offset by increasing the number of IR system instances in the sample. This scenario for the topical partitioning Scheme 1 on the TREC GOV2 dataset with TREC topics 701–850 and ClueWeb'09B dataset with TREC'09 topics 1–50 is illustrated in Figure 4.8. Using sampling with replacement (bootstrapping), 50 samples of IR system instance pools are generated for each varying pool size on each dataset. For the experiment, 100 IR system instances formed for each CSI sample rate are assumed as produce the ground truth for the population. Each sample is compared with exhaustive search using the proposed methods for comparing a non-deterministic IR system with a deterministic one. As the graphs for varying CSI sample rates illustrate, having more IR system instances leads to more accurate evaluation.

Recall that the probability of rejecting the null hypothesis when in fact the alternative hypothesis is true is "the power" of the significance test. Plotting the p values for the number of tests in descending order for each comparison has been a standard practice to illustrate the potential power of significance tests [Sakai, 2006; Robertson and Kanoulas, 2012]. The p values for all comparisons

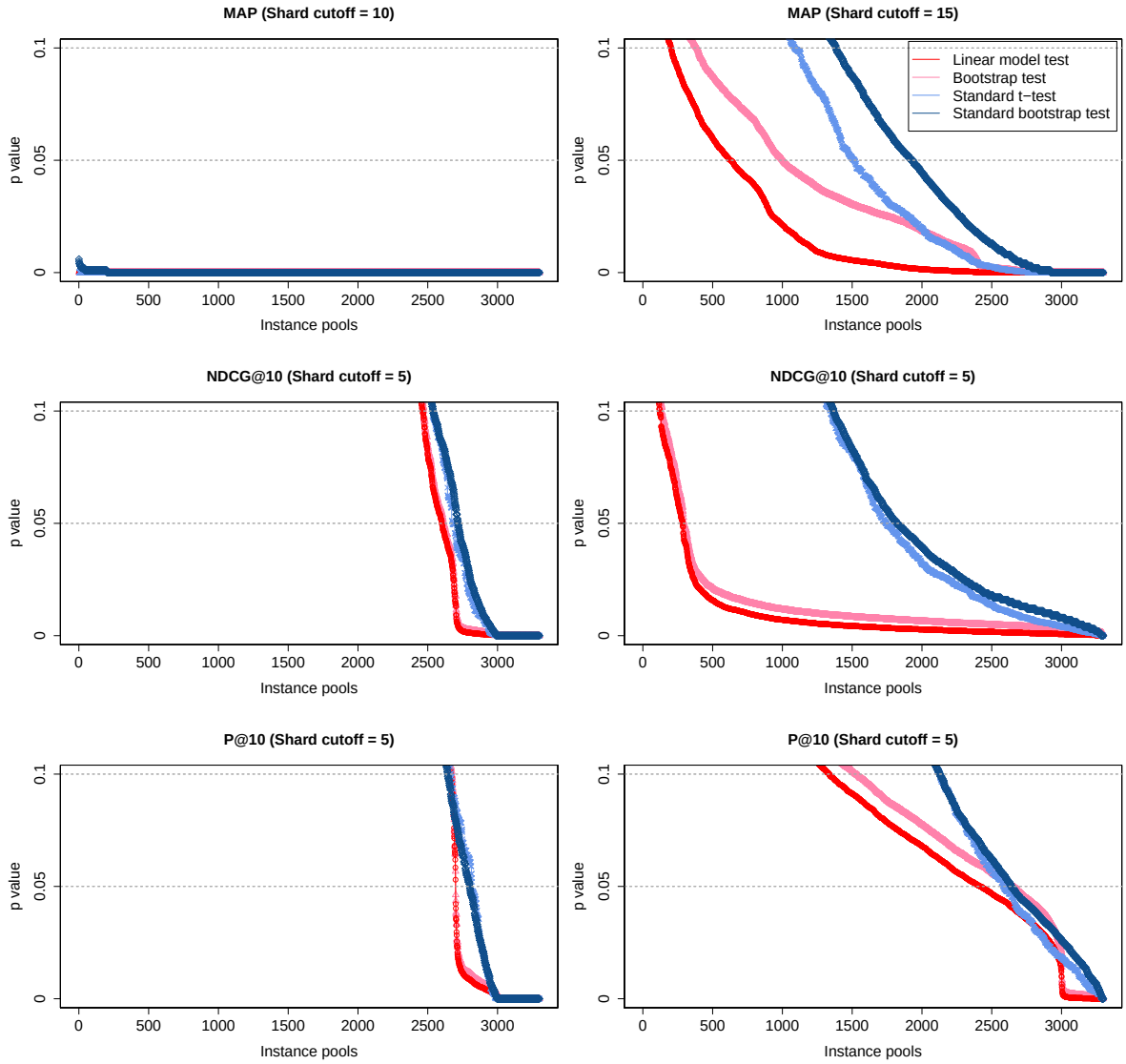


Figure 4.9: The p values for different significance tests when comparing 3300 *IR system instance pools of* Scheme 1 *with exhaustive search on the TREC GOV2 dataset using TREC topics* 701–850 *(left) and on the ClueWeb'09B dataset using TREC'09 topics* 1–50 *(right). Instance pools are sorted in descending order of p value for each test. For standard significance tests an instance is randomly picked from the pool of instances for each comparison.*

performed in the previous experiment on the TREC GOV2 and ClueWeb'09B datasets arranged in the above manner are shown in Figure 4.9. Results for each evaluation metric are shown separately, as each graph illustrates a unique set of comparisons. To construct the plot for the standard t-test and the standard bootstrap test, an IR system instance is randomly selected from the pool of instances

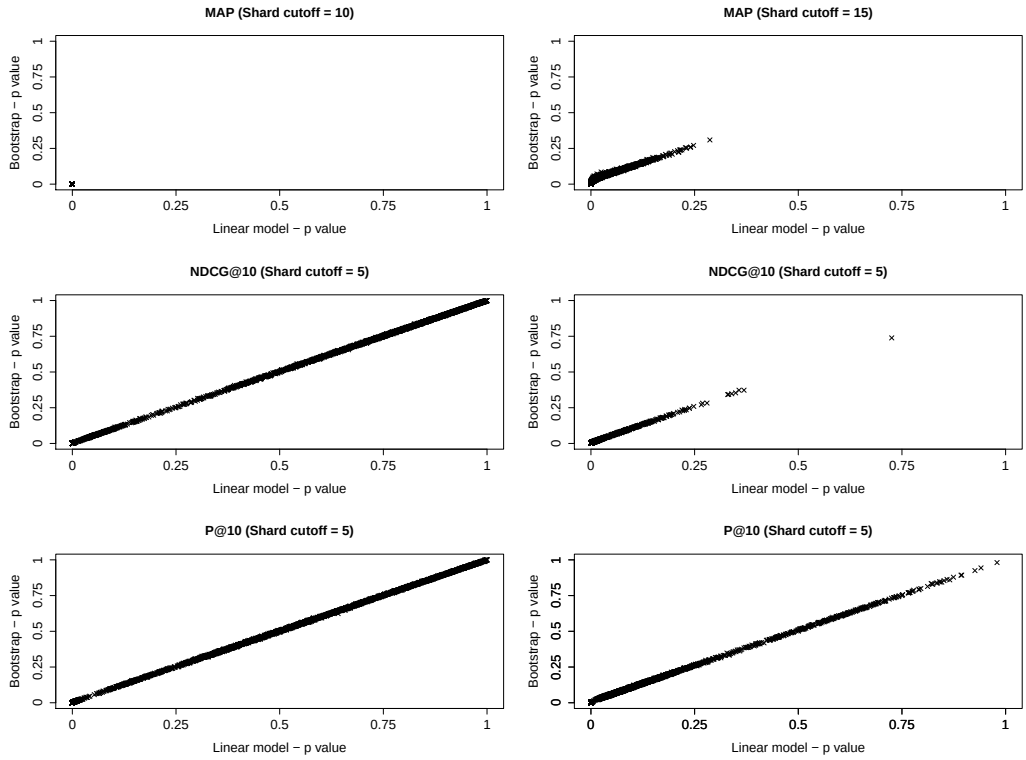


Figure 4.10: Correlation between a bootstrap test and a multivariate linear model test when comparing 3300 *instance pools from the sample based topical sharding* Scheme 1 *with deterministic exhaustive search on the TREC GOV2 dataset with TREC topics* 701–850 *(left) and on the ClueWeb'09B dataset with TREC'09 topics* 1–50 *(right).*

for each comparison. The power of a multivariate linear model test is higher than the bootstrap test for the examples considered here. The difference in the two tests are attributed to fewer assumptions being made about the sampling distribution by the bootstrap test. Both proposed approaches for comparing a non-deterministic IR system with a deterministic system demonstrate a much higher power than either of the standard tests. This indicates the standard tests lack discriminative power, due to less observations (data) seen by standard tests than the proposed two dimensional tests. Further, more comparisons are found to be significantly different with NDCG@10 than with P@10 for the ClueWeb'09B dataset. This is due to the higher prominence given to highly ranked relevant documents and use of graded relevance in NDCG@10. A similar observation has previously been made by Sakai [2007].

When a new evaluation methodology is proposed, verifying the validity of the proposed approach should be done with care. In this chapter, two standard hypothesis testing methodologies are

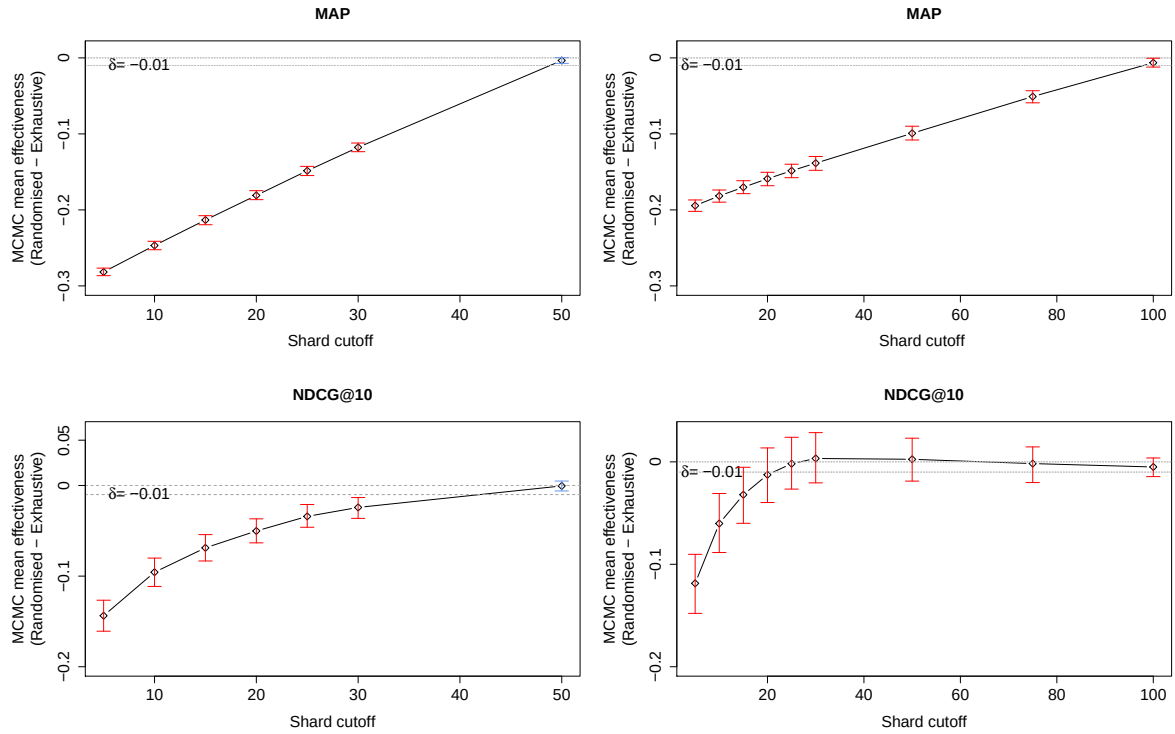


Figure 4.11: The MCMC *mean and the* 95% HPD *interval when comparing random sharding with exhaustive search at various shard cutoffs on the TREC GOV2 dataset with TREC topics* 701–850 *(left) and on the ClueWeb'09B dataset with TREC'09 topics* 1–50 *(right). A CSI sample rate of* 4% *is used.*

extended from first principles to compare a non-deterministic IR system with a deterministic system. The two approaches use different techniques to compute the p value. The agreement between the two approaches on the comparisons conducted for previous experiments on TREC GOV2 dataset, and ClueWeb'09B dataset with each evaluation metric are analysed in Figure 4.10. As shown in the figure, both methods yield consistently correlated results, reinforcing the viability of the two approaches.

Comparing for Equivalent or Greater Effectiveness

When a new sample based IR system is introduced for efficiency purposes, a common question is: What is the minimum sample rate required to achieve an equivalent or greater effectiveness than a deterministic baseline? Recall the method used to test for statistically significant equivalence in Chapter 2. Now a multivariate linear model method to test for most efficient equivalent or greater effectiveness when at least one IR system is non-deterministic is presented.

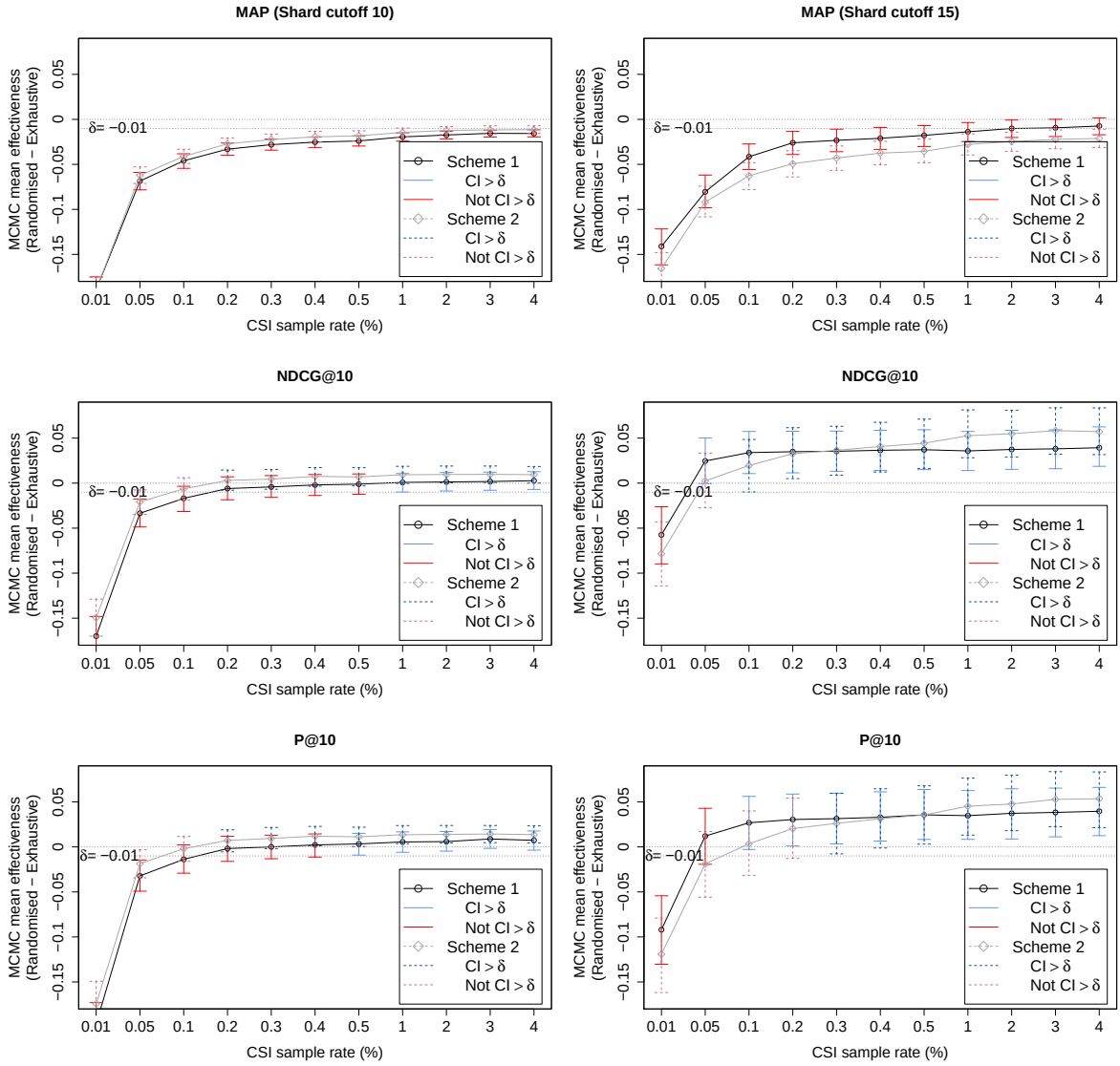


Figure 4.12: The MCMC *mean and the* 95% HPD *interval when comparing the non-deterministic* Scheme 1 *and* Scheme 2 *with exhaustive search on the TREC GOV2 dataset with TREC topics* 701–850 *(left) and on the ClueWeb'09B dataset with TREC'09 topics* 1–50 *(right) with varying CSI sample rates.*

The case study is used as an example. Two potential problems are: find the minimum number of shards to search; or find the minimum sample rate for the CSI when shards are clustered topically. In both examples, an efficient setting that is still able to achieve equivalent effectiveness to the exhaustive solution is needed. Hence, only a minimally consequential degradation is considered, so the null hypothesis is $\mathcal{M}_{\mathcal{A}} - \mathcal{M}_{\mathcal{B}} \leq \delta$, instead of $|\mathcal{M}_{\mathcal{A}} - \mathcal{M}_{\mathcal{B}}| \geq \delta$. So $\mathcal{M}_{\mathcal{A}} - \mathcal{M}_{\mathcal{B}} > \delta$ becomes the alternative hypothesis. Here, $\mathcal{A}$ is the non-deterministic approach and $\mathcal{B}$ is the deterministic baseline. The δ is set to -0.01. That is, a degradation of -0.01 is viewed as acceptable. The null hypothesis is tested, and the smallest setting for which this null hypothesis is rejected is selected. So, effectiveness of non-deterministic IR system is significantly better than the minimal consequential degradation.

The null hypothesis can be indirectly tested using the 95% highest posterior density (HPD) confidence interval computed using a posterior distribution for the system factor of the multivariate linear model. If the confidence interval is above δ, one can reject the null hypothesis, and conclude with 95% confidence that the effectiveness of non-deterministic IR system is above δ of the effectiveness of exhaustive search. However, if δ is above or within the HPD confidence interval, then the null hypothesis cannot be rejected.

Effectiveness against the maximum number of shards searched (shard cutoff) for random shards is analysed in Figure 4.11. Note that the ReDDE algorithm may process a fewer number of shards than the shard cutoff, if the documents retrieved from the CSI for the query correspond to fewer shards than the shard cutoff. As such, even when the shard cutoff is equal to the total number of shards, the effectiveness may not be equivalent to exhaustive search. For random shards, a large number of shards must be searched to achieve a similar effectiveness to exhaustive search. Although the confidence interval when the shard cutoff is equal to the total number of shards is above $\delta = -0.01$ for the TREC GOV2 dataset, it is not for the ClueWeb'09B dataset. One can conclude with confidence that the effectiveness for TREC GOV2 dataset is not less than δ compared to the full index search when the shard cutoff is the same as the total number of shards. However, for the ClueWeb'09B dataset effectiveness may be equivalent to the full index search, but one cannot conclude that with any certainty. The confidence interval can be narrowed only by increasing the sample data (topics and/or IR system instances). This shows how the test incentivizes a researcher to increase the sample data in order to maximise the chance of seeing a statistically equivalent result.

Now the problem of determining the most efficient CSI sample rate for topical sharding schemes that achieve a similar effectiveness to exhaustive search is analysed. The 95% HPD interval for Scheme 1 and Scheme 2 at different CSI sample rates on the TREC GOV2 and the ClueWeb'09B datasets for different evaluation metrics are shown in Figure 4.12. Keeping the focus on Scheme 1

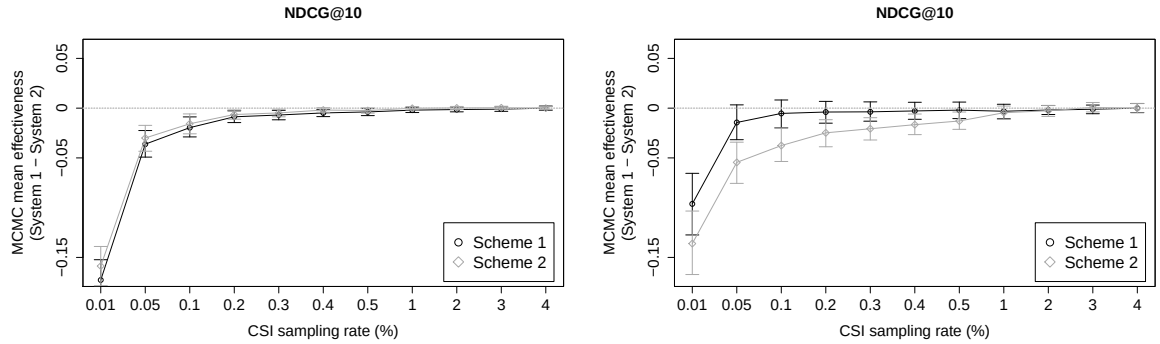


Figure 4.13: The MCMC mean and the 95% *HPD interval when comparing the non-deterministic* Scheme 1 *and* Scheme 2 *with the same algorithm. The CSI sample rate is fixed for System* 1 *at* 4%*, and varies for System* 2*. Presented are results for the TREC GOV2 dataset with TREC topics* 701–850 *(left) and for the ClueWeb'09B dataset with TREC'09 topics* 1–50 *(right).*

with NDCG@10, the results are analysed below. The effectiveness is clearly worse than exhaustive search with low sample rates, such that the maximum consequential degradation δ is above the confidence interval for sample rates of 0.01% and 0.05% on the TREC GOV2 dataset and 0.01% on the ClueWeb'09B dataset. The δ is within the confidence interval for sample rates 0.1% and above on the TREC GOV2 dataset. So, the non-deterministic approach might not be consequently worse than exhaustive search, but one cannot conclude that with any certainty. For sample rates on and beyond 1% for the TREC GOV2 dataset and 0.05% for the ClueWeb'09B dataset, the confidence intervals are above the δ degradation level, and therefore one can conclude that the effectiveness for the non-deterministic method is equivalent or greater than that of the exhaustive method. The confidence intervals are above zero for a sample rate $\geq 0.1\%$ on the ClueWeb'09B dataset. A confidence interval lying above the $randomised - exhaustive = 0$ line implies that the approach performs significantly better than a full index search. A pattern similar to NDCG@10 is observed for P@10 on both datasets. The Scheme 1 proposed by Kulkarni and Callan [2010] is more effective than the exhaustive approach at a 95% confidence level. Shard cutoffs of 10 and 15 are considered for MAP on the TREC GOV2 and ClueWeb'09B datasets respectively. However, for MAP one cannot conclude with confidence that the effectiveness is not worse than δ for CSI sample rates up to 4%. Both sharding schemes demonstrated a similar pattern. Here, a few applications of the proposed evaluation methods for comparing a non-deterministic IR system with a deterministic IR system are illustrated.

Next, results for evaluating two non-deterministic IR systems are presented.

Non-deterministic:Non-deterministic Comparison

Separately comparing Scheme 1 and Scheme 2 at varying CSI sample rates with the same algorithm using a CSI sample rate of 4% (another non-deterministic IR system) is demonstrated in Figure 4.13. The two IR systems are equivalent when the CSI sample rate on the x-axis is 4%. The fact that the comparisons agree as the CSI sample rate approaches 4% for the one with varying CSI sample rate illustrates the validity of the proposed approach for comparing two non-deterministic systems.

4.5 Summary

Despite non-deterministic IR systems becoming increasingly common, little work has been conducted on how best to compare these systems using traditional evaluation frameworks. Exploring the pitfalls of depending on a single instance for evaluation, how an apparently significant result on a single instance can be contradicted by another instance of the same algorithm is shown in this chapter. The notion of two-dimensional significance is introduced. Two significance tests for non-deterministic:deterministic comparison, and a significance test for non-deterministic:non-deterministic comparison are proposed. The proposed methods are used to show that topical partitions significantly improve top 10 retrieval results compared to existing exhaustive search algorithms. Hence, there is scope for further improvements to exhaustive search. How a non-deterministic algorithm can be shown to have greater or equivalent effectiveness to a deterministic algorithm or another non-deterministic system is illustrated. How such a comparison can be used for an effectiveness-efficiency tradeoff is also shown. The proposed significance tests can be used to evaluate problems having variance along many dimensions and repeated observations, or in other domains.

Abbreviations and Symbols used in Chapter 5

Notation	Description
ℓ	Number of labeled examples.
Φ	Number of unique Opcode function calls.
c	A classifier.
h	A machine learning concept.
ht	A hash table.
m	Length of the Opcode function call trace.
N	Number of JavaScript instructions executed.
n	Length of the sliding window.
S	Worst case number of unique n-grams (features).
s	Non-zero features per example.
t	Opcode function call trace.
v	A stratified 10-fold cross validation partition of the dataset.

Chapter 5

Uncertainty in Evaluating Machine Learning Experiments

To automate certain tasks, people prefer machines to learn from previous examples. Lying hidden in these examples are data patterns that are yet to be discovered. Data mining is used to excavate these patterns from a sample of examples known as training data, and to embed them in a concept. The learned concept is used by machines to predict or take actions on new data [Hand et al., 2001; Witten and Frank, 2005; Marsland, 2009]. Hence, a learned concept is useful only when the concept can be generalised to new data.

Two caveats may prevent generalisability. First, the data mining algorithm may fail to sufficiently capture predictive patterns from training data, causing the concept to underfit (high bias). Underfitting can cause a lack of accuracy irrespective of the training-test data sample. Second, the concept may

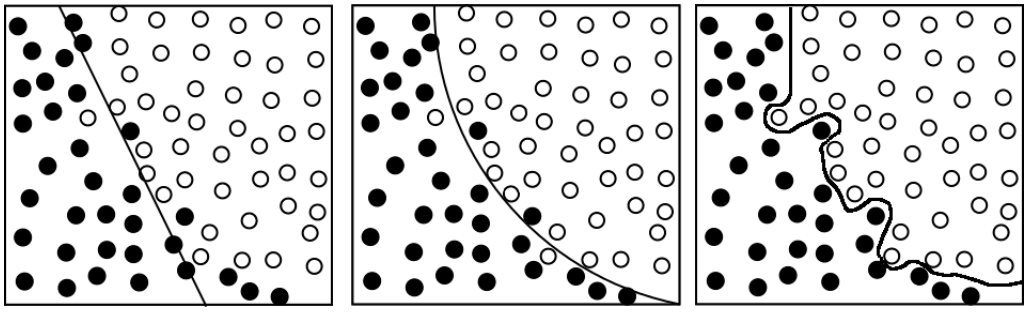


Figure 5.1: Decision function (concept) learned on a training dataset for a binary classification problem. The decision function moves from high bias (left) to high variance (right) as the complexity of the decision function increases. The decision function in the middle is most desirable.

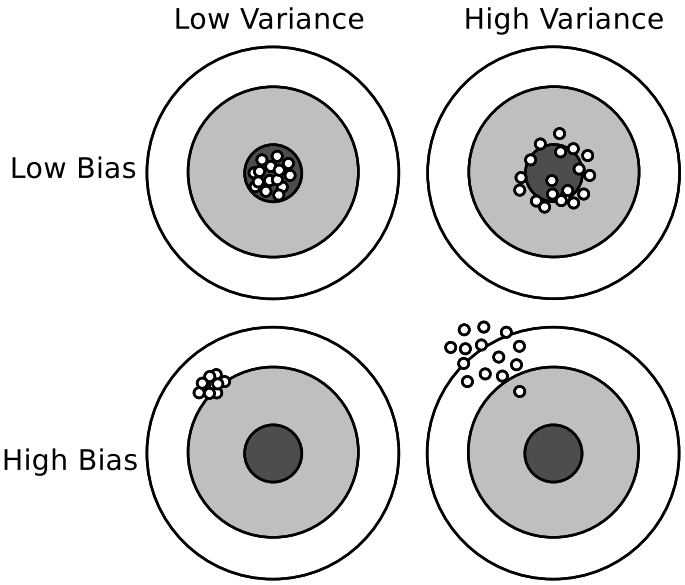


Figure 5.2: The impact of bias and variance on effectiveness for a machine learning concept. The target concept is the one producing the lowest error, or a point in the center of the circles. High bias causes a constant error, represented by equi-distance points from the center. High variance causes the points to disperse. Therefore, a low bias and a low variance concept is most desirable.

overfit and overly match patterns in training data even to include non-generalisable structures in examples (high variance). A high variance concept accurately predicts data very similar to the ones seen in training data, but non-generalisable patterns embedded in the concept cause errors on unseen data. Such a concept causes a high variation in the accuracy given different training-test data samples. Example decision functions for a high bias, an optimal, and a high variance classifier concepts on a hypothetical two dimensional training dataset is shown in Figure 5.1. As the complexity of the decision function increases, the classifier precisely captures patterns in training data to include non-generalisable structures in data, causing a shift from high bias to high variance. The impact of bias and variance on accuracy computed on various training-test datasets is graphically illustrated using a bull's eye diagram in Figure 5.2. A dot in the center represents the most optimal predictor. Dots are away from the center for a high bias classifier showing a constant lack of accuracy, and scattered around for a high variance classifier that cause variability in accuracy.

A concept $\hat{h}(x)$ learned on a sample L is an estimate for a hypothetical concept $h(x)$ learned on the population. Let MSE be the mean squared error of $\hat{h}(x)$, whose expectation is:

$$E\{\mathsf{MSE}\} = \frac{1}{|L|}\sum_{i=1}^{|L|} E\big\{[y - \hat{h}(x)]^2\big\} \tag{5.1}$$

Formally, $E\{[y - \hat{h}(x)]^2\}$ can be decomposed as:

$$\begin{aligned}
E\{[y - \hat{h}(x)]^2\} &= E\{[y - h(x) + h(x) - \hat{h}(x)]^2\} \\
&= E\{[y - h(x)]^2\} + E\{[h(x) - \hat{h}(x)]^2\} + 2 \cdot E\{[y - h(x)] \cdot [h(x) - \hat{h}(x)]\} \\
&= E\{[y - h(x)]^2\} + E\{[h(x) - \hat{h}(x)]^2\} + \\
&\quad 2 \cdot \Big\{E\{[y \cdot h(x)]\} - E\{[y \cdot \hat{h}(x)]\} - E\{[h(x)^2]\} + E\{[h(x) \cdot \hat{h}(x)]\}\Big\} \\
&= E\{[y - h(x)]^2\} + E\{[h(x) - \hat{h}(x)]^2\} + \\
&\quad 2 \cdot \Big\{E\{y\} \cdot E\{h(x)\} - E\{y\} \cdot E\{\hat{h}(x)\} - E\{h(x)\} \cdot E\{h(x)\} + \\
&\quad E\{h(x)\} \cdot E\{\hat{h}(x)\Big\}
\end{aligned}$$

Here $E\{y\} = E\{h(x)\}$. Hence,

$$E\{[y - \hat{h}(x)]^2\} = E\{[y - h(x)]^2\} + E\{[h(x) - \hat{h}(x)]^2\}$$

Similarly, one can decompose $E\{[h(x) - \hat{h}(x)]^2\}$.

$$E\{[h(x) - \hat{h}(x)]^2\} = E\{[h(x) - E\{\hat{h}(x)\}]^2\} + \{[E\{\hat{h}(x)\} - \hat{h}(x)]^2\}$$

Therefore,

$$\begin{aligned}
E\{[y - \hat{h}(x)]^2\} &= E\{[y - h(x)]^2\} + E\{[h(x) - E\{\hat{h}(x)\}]^2\} + \{[E\{\hat{h}(x)\} - \hat{h}(x)]^2\} \\
&= var(\varepsilon) + bias^2 + variance
\end{aligned}$$

The error in classification is composed of the irreducible error ε, bias, and variance [Hastie et al., 2009]. Hence, a lower bias and a lower variance is desirable. However, bias and variance are in tension. Reducing one often causes the other to increase. However, both can be lowered by improving the learning algorithm, and increasing the size of the training dataset [Banko and Brill, 2001].

High variance is problematic for evaluation. A concept evaluated on similar training and test datasets may show a high effectiveness, that may lead to wrong experimental conclusions. Therefore, high variance classifiers must be evaluated carefully.

In this chapter, a classification problem – dynamic detection of drive-by download attacks from the domain of information security is considered as a case study. The rapidly changing malicious behavioural patterns and the difficulty in manually locating new predictive features to detect new attacks have caused drive-by download detection techniques to use feature spaces automatically gen-

erated from a training dataset [Rieck et al., 2010; Curtsinger et al., 2011]. Such feature spaces have successfully been used in natural language processing and text classification problems [Jurafsky and Martin, 2009; Joachims, 1998]. In this chapter, demonstrating that unbounded feature spaces, and improperly curated small datasets can lead to a high classifier variance and unreliable predictors for detecting drive-by download attacks, a bounded feature space to improve the accuracy and reliability of dynamic detection of drive-by download attacks is proposed in Section 5.3. An example of how a high variance classifier can lead to wrong conclusions is shown in Section 5.4. The time efficiency of data stream classification algorithms can be improved by using non-deterministic procedures [Aggarwal, 2007]. However, evaluating the effectiveness of non-deterministic classification algorithms can be difficult as repeated assessments will produce different effectiveness scores for the same training-test data sample. How multivariate linear model tests for IR evaluation can be extended to evaluate non-deterministic classifiers with repeated observations and how effectiveness is traded off for efficiency is shown in Section 5.5. The case study is presented next.

5.1 Detecting Drive-by Download Attacks

Traditional intrusion prevention techniques, such as firewalls, provide protection against a wide variety of *push-based* security exploits. But, people's increased reliance on the web for routine activities such as communication and entertainment has created an opportunity for adversaries to infect many client computers simultaneously using *pull-based* methods. As a result, recent web-based exploits are predominantly deployed using pull-based mechanisms [Provos et al., 2007]. An increasingly common class of pull-based clientside security exploits delivered via the internet are drive-by downloads [Moshchuk et al., 2006; Provos et al., 2008]. These attacks usually leverage web browser vulnerabilities in order to hide malicious software downloads onto a computer or mobile device. Backdoors that provide unhindered remote access to the compromised client devices, trojan downloaders that fetch and install other programs from the web, and trojan proxies that re-route Internet traffic are some of the malware installed via drive-by downloads. Consequently, a user may loose sensitive data, or the client computer may become a part of a large *botnet* [Provos et al., 2007].

JavaScript is the primary programming language used to implement and deploy context rich web applications. However, JavaScript is not a security conscious language [Hoffman and Sullivan, 2008; Johns, 2008]. The ability to run arbitrary code on client devices, and the tightly integrated interaction between browser and supporting technologies makes JavaScript an obvious candidate for targeted drive-by download attacks [Egele et al., 2009a; Sophos, 2010]. In a recent study of

common attack vectors using drive-by download attacks, Provos et al. [2008] discovered more than 3 million malicious URLs. Adversaries use a variety of tactics to lure users to malicious webpages. Compromised websites, user contributed content, third-party widgets and advertising are commonly used to insert malicious content into benign sites [Provos et al., 2007]. In fact, finding malicious web content injected into otherwise trusted websites is now more common than stumbling upon rogue websites which are explicitly designed to distribute malware [Sophos, 2010]. Poisoning the search results of popular search engines using search engine optimization is another common strategy used to draw users to malicious websites [John et al., 2011; Sophos, 2010].

As a consequence, detection and prevention of drive-by downloads has been an important topic to security researchers. Typical solutions may run *offline* on server infrastructure, producing *malware signature*s and *blacklist*s [Wang et al., 2006], while others provide *realtime* protection on client devices. Analyzing the properties of webpage URLs [Ma et al., 2009; Stokes et al., 2010; Zhang et al., 2011], and investigating the embedded webpage content [Cova et al., 2010; Curtsinger et al., 2011; Dewald et al., 2010; Rieck et al., 2010] are two defensive strategies commonly employed to detect malicious webpages. URL based strategies operate on the serverside as an offline solution to generate blacklists [Stokes et al., 2010; Zhang et al., 2011]. In contrast, webpage content analysis attempts to detect webpages performing drive-by download attacks on-the-fly. If the content analysis is suitably efficient, the approach may be used for realtime detection on the clientside. *Static* and *dynamic* (emulation based) analysis are two possible approaches to webpage content analysis.

Static analysis in the form of *misuse detection*, which uses a set of pre-defined rules and heuristic signatures to detect malicious patterns, is often used on the clientside in *antivirus* solutions. Static analysis integrated with machine learning to detect drive-by downloads was subsequently proposed by Rieck et al. [2010]. Static analysis can reveal malicious features in JavaScript code, but dynamic features offered by JavaScript language [Richards et al., 2011] and obfuscated or encrypted code appearing in both benign and malicious pages [Kaplan et al., 2011] can marginalize the effectiveness of approaches depending solely on static analysis. To overcome evasion by code obfuscation, a *semi-dynamic* approach that detect malware using static analysis of deobfuscated code [Curtsinger et al., 2011] has also been explored. Dynamic analysis techniques in the form of misuse detection [Dewald et al., 2010; Nazario, 2009; Seifert et al., 2007], and anomaly detection [Cova et al., 2010; Rieck et al., 2010] has also been extensively studied. Compared to static analysis, dynamic analysis has the potential to detect malicious webpages more accurately as the attack can be exposed by monitoring the execution sequence.

Anomaly detection techniques commonly use machine learning and classification to identify

drive-by download attacks, and differ mainly in feature extraction techniques. The features for anomaly detection can be manually crafted [Canali et al., 2011; Cova et al., 2010], automatically extracted from the training dataset [Curtsinger et al., 2011], or a combination of both [Ma et al., 2009; Rieck et al., 2010]. Automatic feature extraction has the advantage of not requiring domain specific knowledge to come up with a set of features. Existing automatic feature extraction methods for dynamic analysis include sample specific values such as user defined JavaScript identifiers (variable and function names) and values (strings, integers, etc.) to construct the feature space for machine learning [Curtsinger et al., 2011; Rieck et al., 2010]. A large portion of today's JavaScript attacks either copy and paste, or borrow code from previous attacks [Curtsinger et al., 2011]. Capturing these unique identifiers in the feature space can lead to high detection rates. However, inclusion of variable assignments in the automatically generated feature space can also result in a feature space of unbounded size; an *overfitting* (*high variance*) classifier [Hawkins, 2004; Marsland, 2009]; and/or a resource intensive classification process [Hawkins, 2004]. The problem is much more severe when automatic feature extraction is employed with dynamic analysis. As a result, dynamic analysis solutions with automatic feature extraction are often computationally expensive, and have not gained widespread usage in resource constrained devices.

A proxy server is a reasonable compromise between signature-based and fully dynamic anomaly detection on the clientside as the approach can protect multiple users simultaneously on a private network, and localize computational costs to a single machine [Li et al., 2011; Moshchuk et al., 2007; Rieck et al., 2010]. The proxy-based approach can also be augmented with an array of solutions to detect drive-by download attacks. However, proxy server based solutions are limited to private networks.

None of these solutions is entirely satisfying. While blacklists and signature-based solutions somewhat mitigate the spread of malware, empirical studies show that new outbreaks tend to outpace updates to commercial databases [Cova et al., 2010; Rieck et al., 2010]. Static and semi-dynamic anomaly detection approaches can also be evaded using code obfuscation or by rearranging the code [Curtsinger et al., 2011]. So, developing novel clientside dynamic anomaly detection approaches capable of discovering previously unrecognized exploits without bombarding the user with a disproportionate number of false alarms, and that minimize resource usage remain an attractive research goal. The background on detecting drive-by download attacks is reviewed next.

5.2 Background

In this section, current JavaScript drive-by download attacks are compared and contrasted using a simple example. Finally, the prior work on detecting drive-by download attacks are reviewed.

5.2.1 JavaScript Drive-by Download Attacks

Line	JavaScript Code
1	`// (a) Shell code and construction of the NOP sled`
2	`shellcode = unescape("%u9090%u9090...");`
3	`heap_block_size = 0x400000;`
4	`shellcode_size = shellcode.length * 2;`
5	`spray_slide_size = heap_block_size - (shellcode_size + 0x38);`
6	`spray_slide = unescape("%u0505%u0505");`
7	`while (spray_slide.length * 2 < spray_slide_size) {`
8	`  spray_slide += spray_slide;`
9	`}`
10	`spray_slide = spray_slide.substring(0, spray_slide_size / 2);`
11	
12	`// (b) Heap spray`
13	`heap_spray_to_address = 0x05050505;`
14	`heap_blocks = (heap_spray_to_address - 0x400000) / heap_block_size;`
15	`memory = new Array();`
16	`for (i=0; i < heap_blocks; i++) {`
17	`  memory[i] = spray_slide + shellcode;`
18	`}`
19	
20	`// (c) Exploitation of the browser vulnerability`
21	`...`

Figure 5.3: JavaScript code segment from a heap spraying attack.

Figure 5.3 is a sample snippet of JavaScript code used to perform a drive-by download attack. The code contains a *shellcode* (line 2), a routine for constructing a NOP sled (lines 3–10), and a heap spray (lines 12-18). The shellcode is a binary payload injected and executed on a user device. The NOP sled positions the shellcode in a protected memory region so that it can overwrite another running process, and execute on the target system. The final location of the shellcode is determined by a *heap spray*, which positions the NOP sled/shellcode in a specific heap location. In JavaScript this can be achieved by populating an array with fixed length copies of the NOP sled and shellcode. The heap spray is necessary to overcome address space layout randomization (ASLR) [Shacham et al., 2004] protection mechanisms provided in many modern operating systems. In the final step, the attack triggers a vulnerability in the browser, or a browser plug-in causing random execution of the shellcode. A drive-by download attack is usually preceded by pre-attack JavaScript routines, which are used to deliver a suitable attack, or to conceal the attack. These pre-attack JavaScript routines include, but are not limited to: browser sniffing to identify a vulnerability in the browser or a browser plugin; dynamic redirections to guide execution to an exploit; deobfuscation and DOM manipulation to dynamically embed scripts and vulnerable objects; or invisible *iFrames* to extract a hidden attack.

5.2.2 Related Work

Since Provos et al. [2008; 2007] presented studies on the widespread abuse of drive-by downloads in web browsers, a series of research studies have been undertaken to mitigate this problem. The two general approaches employed to solve the problem are URL or webpage content based analysis. Webpage content analysis can be static, dynamic (emulation based), or a combination of both. These methods can be incorporated into an offline service in a client-server environment, or on the clientside as a *semi-realtime* solution embedded in a web proxy, or as a realtime solution embedded in the web browser.

Offline Solutions

Offline solutions include blacklists of malicious URLs or signature databases [Wang et al., 2006], which can be used to protect client devices. Webpage content analysis is one strategy employed by offline solutions to detect malicious webpages. These solutions can use a variety of tools to analyze webpage content, and are not restricted by execution time or computing resources. *High interaction client honeypots* monitor the entire operating system, including a web browser, using a virtual

machine. When a potentially malicious URL is loaded in the sandboxed web browser, unexpected changes to the virtual machine environment can be flagged as malicious. These solutions are effective at identifying new attacks (*zero day exploits*) and have negligible false positive rates [Seifert et al., 2009]. However, high interaction client honeypots can only detect malware that targets the part of client system implemented in the honeypot. Furthermore, the detection mechanism is vulnerable to various evasion techniques employed by adversaries, such as time bombs (delayed exploits triggered long after the initial visit to a web page) or logic bombs (attack is triggered after a user satisfies a set of conditions or actions).

A less resource intensive alternative is a *low interaction client honeypot*, which simulates a network protocol stack or a subset of targeted browser features in order to identify specific attack vectors. Examples include PhoneyC [Nazario, 2009] and HoneyC [Seifert et al., 2007]. These solutions examine the response from a server against pre-defined rules and heuristics, or an anomaly detection classifier, in order to assess the maliciousness of a webpage. Low interaction client honeypots must be instrumented to detect specific attack vectors, and therefore are not a robust solution for detecting new exploits [Seifert et al., 2007].

Executing code in a controlled environment and monitoring the resulting activity was proposed as lightweight alternative to client honeypots [Cova et al., 2010]. This solution uses a manually crafted set of domain-specific features, such as website behaviour for different browser personalities, number of redirections, number of components loaded, or frequent use of string operations. Each feature is then evaluated against a pre-defined prediction model that assigns a score to each feature. If the weighted sum of the scores for a page exceeds a predefined threshold, then the page is marked malicious.

Analyzing the properties associated with webpage URLs is another strategy used in offline solutions. A complementary approach, which derives a combination of automatically extracted, and manually created domain specific features from URL and host-based properties for supervised machine learning was proposed by Ma et al. [2009].

Discovering entire malware neighbourhoods or malware distribution networks is also a viable approach in offline detection. By identifying not only the malicious webpages, but also the landing pages linking the malicious sites, more malicious pages can be identified and blacklisted. In this approach, the data is first collected by a search engine crawler. The solution then produces a web graph of hyperlinks. While traversing the graph, the algorithm uses telemetry reports produced by anti-malware products running on client devices to discover potential malware networks [Stokes et al., 2010].

Another solution uses secondary URLs, redirect chains, and malware information logged by a cluster of high interaction client honeypots to identify networks where the members are involved in distributing malware executables (malware distribution networks). For each malware distribution network, the method then discovers the servers shared by the drive-by download attacks (central servers). The URLs of these central servers are then converted into regular expression signatures. These signatures are then used by the web crawler or the browser to blacklist sites launching drive-by download attacks [Zhang et al., 2011].

Despite many advances, existing offline solutions are resource intensive and/or not transparent. Furthermore, they are vulnerable to transience and cloaking techniques [Curtsinger et al., 2011]. The approaches may also require additional hardware to be maintained, and can introduce long latency periods into a normal browser session, frustrating users. As a result, these solutions are not suitable for realtime detection of drive-by download attacks on client devices.

Realtime Solutions

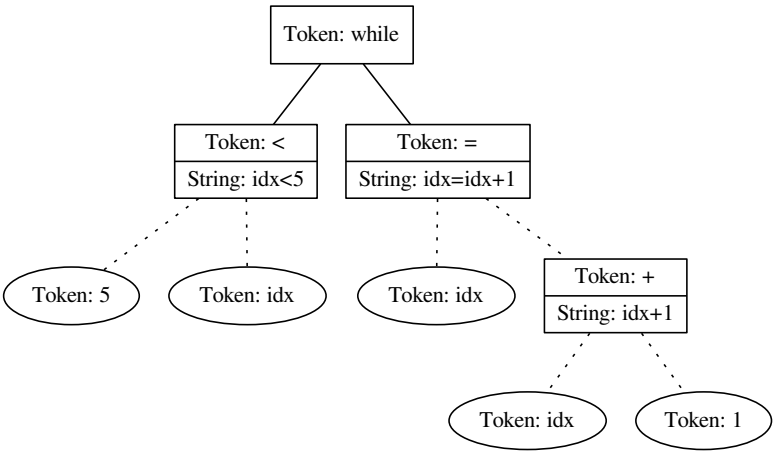
Realtime approaches analyze each page on demand. Realtime solutions must be efficient and transparent to the user. These methods also tend to have strict resource restrictions. Since JavaScript is an interpreted language, static and dynamic analysis can be performed in realtime. These solutions are effective against transient and cloaking techniques that are often employed by adversaries, as specific JavaScript code paths are executed and analyzed. More fine-grained control for dynamic in-browser methods is possible by executing JavaScript in a sandbox. This can be done with the help of readily available JavaScript interpreters, allowing malicious pages to be blocked before rendering. Dewald et al. [2010] proposed one such tool, called ADSandbox, that uses a modified Mozilla SpiderMonkey JavaScript engine to log all accesses to JavaScript objects, data type conversions, and function calls that occur during interpretation of embedded JavaScript code. The traces are then matched against previously constructed regular expressions of malicious behaviour.

Curtsinger et al. [2011] presented a similar approach in the ZOZZLE tool. ZOZZLE performs static analysis of deobfuscated JavaScript code in the browser. Each code segment sent to the JavaScript engine for compilation is transformed into a JavaScript abstract syntax tree (AST). Features for classification are extracted from AST nodes corresponding to expressions and variable declarations. Each feature consists of the context in which the AST node appears and the string of the AST node. As a result, ZOZZLE features capture the hierarchical structure of the code, as opposed to the flat feature space of prior methods. ZOZZLE's process of feature extraction is shown in Figure 5.4. To select fea-

(a) JavaScript code segment

```
while (idx<5) {
 idx=idx+1;
}
```

(b) AST



(c) ZOZZLE features

```
while: idx<5
while: idx=idx+1
```

Figure 5.4: ZOZZLE *extracting features for classification from abstract syntax tree (AST) corresponding to JavaScript code segments.*

tures that indicate malware presence, a χ^2 test (for one degree of freedom) with a 99.9% confidence interval is used. Finally, a Bayesian classification over the filtered hierarchical features in the AST is performed to predict maliciousness.

Lu et al. [2010] take a different approach to dynamic detection of drive-by downloads. Since this class of attacks must inject a payload, Lu et al. focus analysis on only the downloaded content. Browsers use MIME types to receive data content divided into two classes: supported and unsupported. File types such as JPEG, PDF, and HTML are supported and executed transparently, while other file types such as EXE or ZIP require *user consent* before download or execution. The method creates a non-executable sandbox for all supported data types, thereby preventing any binary file to execute without explicit user intervention.

Other methods target specific types of drive-by download attacks in order to balance efficiency and effectiveness. For example, approaches to detect heap spraying attacks in the browser runtime were proposed by Egele et al. [2009b] and Ratanaworabhan et al. [2009]. Both methods detect a heap spraying attack by identifying shell code in the memory heap. Egele et al. emulate x86 assembly instructions to evaluate heap buffers (all string allocations) directly in the browser. Ratanaworabhan et al. identify possible shell codes using NOP sled detection using a tool called NOZZLE. NOZZLE samples objects allocated on the heap, and analyzes the contents concurrently to detect the NOP sled event leading up to the final shell code execution. By detecting the NOP sled, NOZZLE can abort the process before shell code execution.

Realtime detection methods must be resource conscious. Static or semi-dynamic analysis is one way to increase efficiency, but often at the cost of effectiveness. Other methods target specific attacks to improve response time. While effective for a subset of attack vectors, these approaches leave the system vulnerable to other classes of attack. Fully dynamic anomaly detection has not yet achieved widespread acceptance as a realtime solution on client devices as existing dynamic analysis techniques are only targeted at a small subset of attach vectors, or are too resource intensive.

Semi-realtime Solutions

Recent advancements in hardware have enabled the use of a subset of offline solutions directly on the clientside [Moshchuk et al., 2007; Li et al., 2011]. These methods run a server infrastructure, often embedded in a web proxy, and prevent users from loading malicious webpages during a browser session.

One such proxy resident technique that combines both static and dynamic analysis of the em-

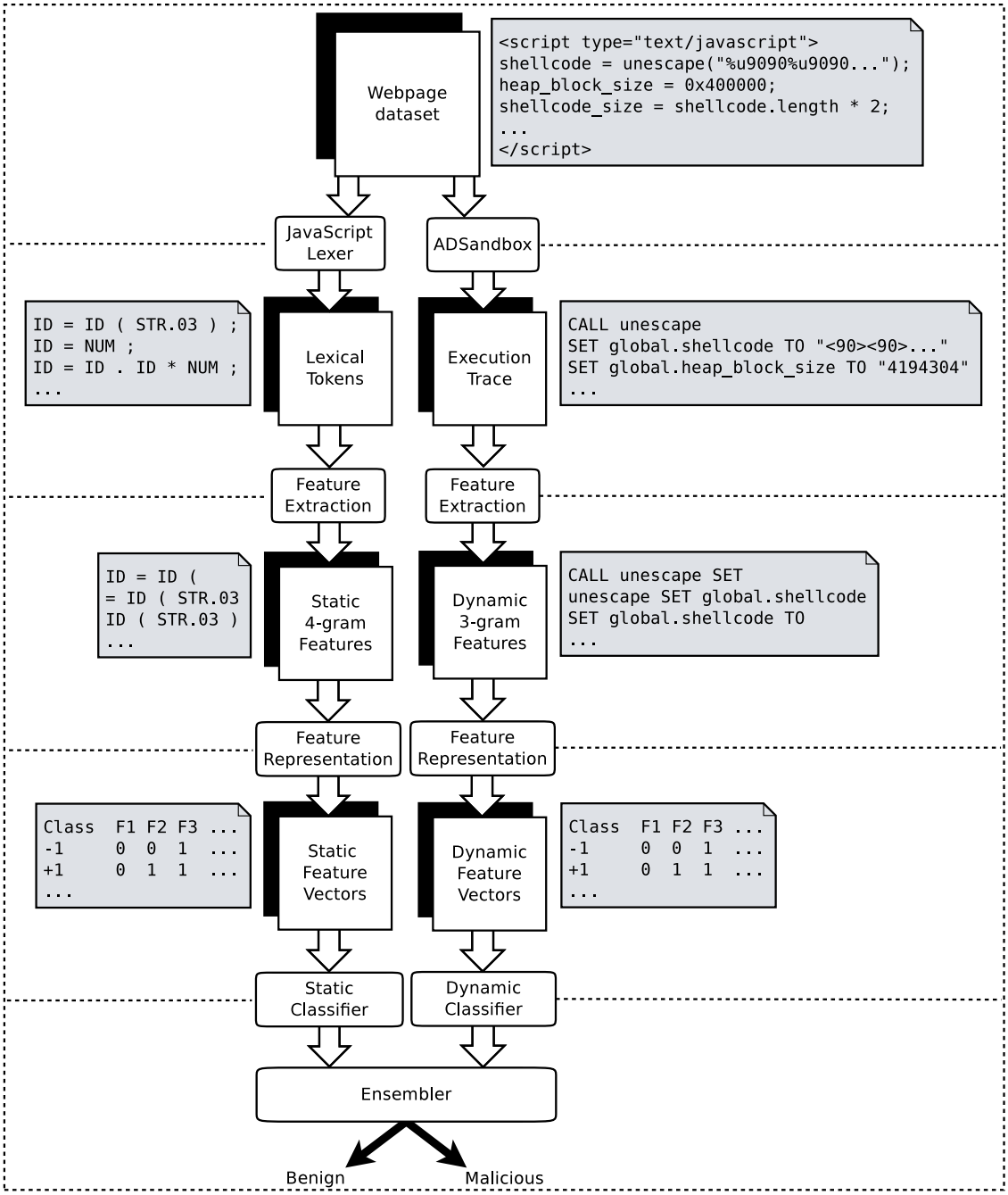


Figure 5.5: CUJO *using ensemble classification to detect malicious webpages.* CUJO *decomposes embedded JavaScript in a webpage using a lexical parse of the script, as well as generating an execution trace during rendering. Both components are then processed separately, and a final, ensembled classification is made using the two models.*

```
CALL unescape
SET global.shellcode TO "<90><90>..."
SET global.heap_block_size TO "4194304"
SET global.shellcode_size TO "138"
SET global.spray_slide_size TO "4194110"
CALL unescape
SET global.spray_slide TO "^E^E"
SET global.spray_slide TO "^E^E^E^E"
SET global.spray_slide TO "^E^E^E^E^E^E^E^E"
...
SET global.spray_slide TO "^E^E^E^E^E^E^E^E..."
...
```

Figure 5.6: CUJO *dynamic analysis trace.*

bedded JavaScript code to detect drive-by download attacks was proposed by Rieck et al. [2010]. As illustrated in the Figure 5.5, the tool, called CUJO, extracts lexical tokens from the embedded JavaScript code for static analysis. For dynamic analysis, the Mozilla SpiderMonkey JavaScript engine is used with a virtual browser environment to produce a log of JavaScript function calls and the state changes of JavaScript objects. Figure 5.6 depicts the dynamic trace produced by CUJO for the JavaScript code snippet shown in Figure 5.3 (a). The n-gram output from static and dynamic analysis is represented in two separate vector spaces and classified independently using a Support Vector Machine classifier. If either of the classifiers return a positive result, the web page is classified as malicious.

Theses techniques permit resource intensive dynamic detection methods on the clientside. However, these solutions require special hardware and are often restricted to private networks. In the following section, an efficient and effective dynamic solution for detecting drive-by download attacks in realtime is proposed.

5.3 Proposed Approach

Similar to other successful dynamic approaches to drive-by download detection, the execution of JavaScript is analysed to identify malicious activity. The proposed approach dynamically evaluates browser Opcode call sequences and uses supervised machine learning for classification. Figure 5.7

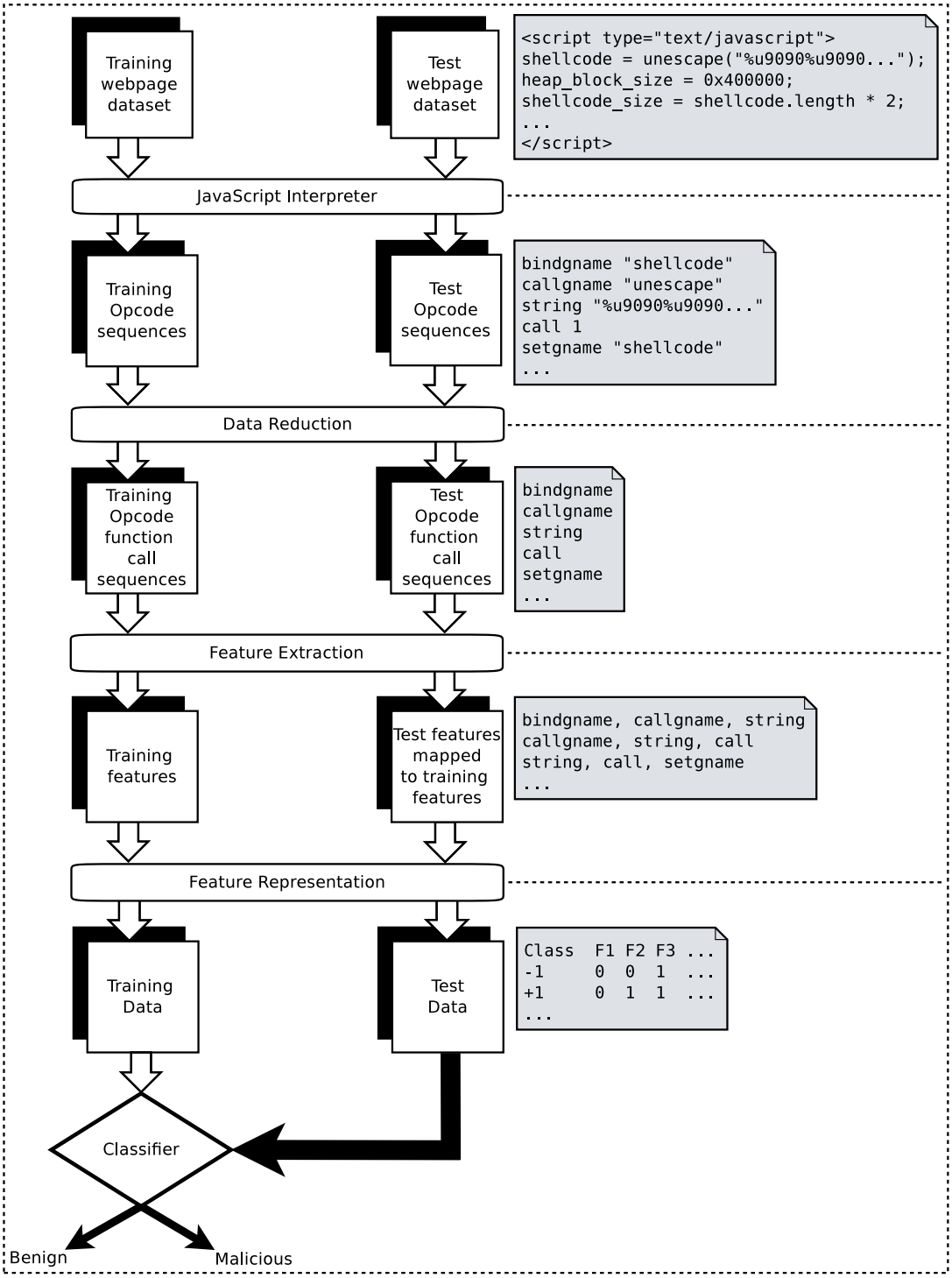


Figure 5.7: Opcode *data processing, reduction, transformation, and classification.*

illustrates the workflow for the proposed approach. There are five distinct phases in the overall process. As shown in the diagram, an Opcode sequence is extracted from the JavaScript engine, which generates Opcodes as a part of the rendering process for each webpage. The Opcodes are then converted to a format suitable for machine learning using data reduction, feature extraction, and feature representation. In the next phase, a classifier is trained using a seeded, labeled training dataset. Finally, new webpages are classified using the pre-trained classifier.

```
bindgname "shellcode"
callgname "unescape"
string "%u9090%u9090..."
call 1
setgname "shellcode"
bindgname "heap_block_size"
uint24 4194304
setgname "heap_block_size"
bindgname "shellcode_size"
getgname "shellcode"
length
int8 2
mul
setgname "shellcode_size"
bindgname "spray_slide_size"
getgname "heap_block_size"
getglobal "shellcode_size"
int8 56
add
sub
setgname "spray_slide_size"
bindgname "spray_slide"
callgname "unescape"
string "%u0505%u0505"
call 1
setgname "spray_slide"
goto 94 (20)
getgname "spray_slide"
length
int8 2
mul
getgname "spray_slide_size"
lt
trace 0
bindgname "spray_slide"
getgname "spray_slide"
getgname "spray_slide"
add
setgname "spray_slide"
getgname "spray_slide"
length
int8 2
mul
...
...
getgname "spray_slide_size"
lt
trace 0
bindgname "spray_slide"
getgname "spray_slide"
getgname "spray_slide"
add
setgname "spray_slide"
getgname "spray_slide"
length
int8 2
mul
...
```

Figure 5.8: Disassembled Opcode *generated using the* Firefox JaegerMonkey *JIT compiler for the JavaScript code segment shown in Figure 5.3 (a).*

Data extraction: For comprehensive dynamic analysis, an algorithm needs to capture an interme-

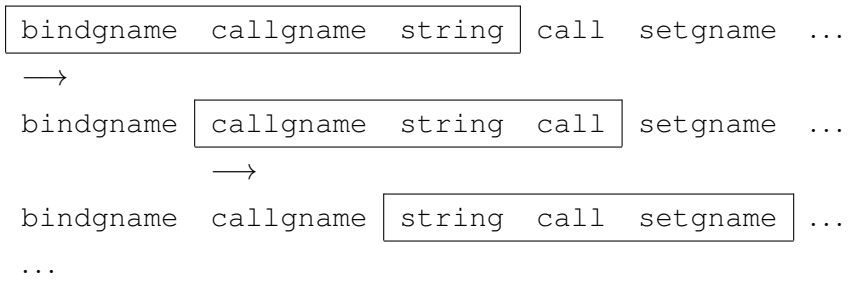


Figure 5.9: 3-grams extracted from the Opcode *function call sequence for the trace shown in Figure 5.8.*

diate representation of the webpage rendering process. JavaScript Opcode (bytecode) is one such intermediate produced by default by the JavaScript interpreter. Therefore the method requires no additional emulation. All JavaScript Opcode calls during the rendering process are logged using a modified version of the Firefox web browser [1]. Figure 5.8 illustrates the disassembled browser Opcode corresponding to the malicious JavaScript code segment shown in Figure 5.3 (a).

Data reduction: To reduce the intermediate space usage and to improve efficiency, the function call (first most element) of each Opcode is extracted to construct a sequence of Opcode keywords. For example, the disassembled Opcode calls in Figure 5.8 can be represented more succinctly as "`bindgname callgname string call setgname ...`". Each distinct Opcode function call is mapped to a unique integer within the JavaScript engine. Note that each Opcode keyword is shown in human readable format in Figure 5.8 for clarity. The Opcode function calls are captured as a sequence of integers and extracted with no additional processing, thereby reducing the log trace overhead. Evaluating only function keyword sequences has been successfully deployed at the kernel level for anomaly detection in operating systems [Forrest et al., 2008; 1996; Mutz et al., 2006; Warrender et al., 1999].

Feature extraction: Since a single browser session can be composed of many successive page loads, the intermediate call trace is treated as a data stream, and features are extracted for machine learning using a single pass sliding window [Datar et al., 2002]. As the window progresses through the stream n-grams of Opcode keywords are extracted to better capture order of execution within the training and prediction model. The use of n-grams in the sliding window model is a widely studied approach to improve overall effectiveness in machine learning and anomaly detection [Lee and Stolfo, 1998;

[1] http://www.mozilla.com/en-US/firefox

Perdisci et al., 2006; Rieck et al., 2010].

For example, a sliding window of length three over the Opcode keywords from Figure 5.8 produces the 3-grams shown in Figure 5.9. Each unique n-gram results in a distinct feature in the overall feature space used for the machine learning model. Since increasing n can also dramatically increase the size of the overall feature space, the most appropriate value of n is often determined empirically in order to find an acceptable effectiveness and efficiency tradeoff.

Feature representation: The n-grams for the features are represented as a vector space, where each unique n-gram constitutes a dimension. So, the maximum size of the feature space is bounded above by Φ^n, where Φ is the number of distinct Opcode tokens. Presence or absence of each n-gram in the trace is represented in the sparse vector space using a binary representation. However, the unique n-grams for a given webpage can never be greater than the length of the Opcode call trace in practice. Let m represent the length of an Opcode call trace for a given webpage. While the potential feature space is very large, the total number of distinct n-grams is linear with respect to the length of the JavaScript call trace for a rendered webpage since the trace can have at most $m - n + 1$ distinct n-grams. Therefore, the non-zero n-grams in the vector space is $s \in \mathcal{O}(m - n)$ in the worst case. As is always the case with n-gram sliding window methods, finding the best n that captures the essence of call ordering in the trace while not being unnecessarily sparse is an exercise in experimental tuning.

ALGORITHM 5 Transformation of an Opcode trace into a vector

```
// Input: t_i ← Opcode function call trace for the i·th webpage.
// Input: ht ← a hash table mapping corresponding n-grams to features.
x_i ← new Array[1 ... length(ht.keys())]
sliding_window ← new Queue()
for function_call in t_i do
    sliding_window.queue(function_call)
    if sliding_window.size() == n then
        x_i[ht.lookup(sliding_window)] = 1
        sliding_window.dequeue()
    end
end
return x_i
```

ALGORITHM 6 Classification

// Input: $x_i \leftarrow$ feature vector for the $i{\cdot}th$ webpage.
// Input: $model \leftarrow \omega$
$score \leftarrow 0$
for $feature \leftarrow 1$ ***to*** $length(model)$ **do**
    $score = score + model[feature] \times x_i[feature]$
end
return $sign(score)$

Data mining: Finally, a linear SVM algorithm (LIBLINEAR) is used to derive an appropriate hyperplane, which can then be used to make predictions on future data samples. Let $X_{\Phi^n \times \ell}$ be the feature vectors and Y_ℓ be the corresponding class labels (benign: -1 and malicious: $+1$) for ℓ examples. Given $X_{\Phi^n \times \ell}$ and Y_ℓ, the linear SVM algorithm constructs a weight vector ω representing the maximum-margin hyperplane, such that $h(X_{\Phi^n \times \ell}) = sign(\omega^{\mathcal{T}} \cdot X_{\Phi^n \times \ell})$. The Algorithm 5 and Algorithm 6 presents the pseudocode for Opcode analysis feature extraction and classification respectively.

The input to the data mining algorithm captures the execution sequence of the JavaScript Opcode function calls. The features for machine learning are derived from the function calls along with the context they occur using the n-gram model. As the features are derived from a fixed set of tokens, the overall feature coverage is high, and the probability of unseen features in the test dataset is low. Therefore, the machine learning algorithm has sufficient data to make accurate predictions on previously unseen traces since most of the feature space is represented in the training model.

Worst Case Analysis: The total length of an Opcode call trace (m) for a given webpage is linear with respect to the number of instructions executed (N) by the embedded JavaScript code. As such, the worst case efficiency of the Opcode analysis technique is dependent on the inherent efficiency of the JavaScript code. If the JavaScript code is efficient, the generated Opcode trace is very small, and subsequently the number of features extracted is small. However, if for example the code being executed has a worst case efficiency of $\mathcal{O}(N!)$, the number of Opcode tokens generated can also be $\mathcal{O}(N!)$ which is potentially catastrophic. While this seems to be significant limitation, this problem generally applies to any dynamic method that requires JavaScript code to be executed before processing.

Thankfully, just-in-time JavaScript compilers now have several optimizations such as maximum recursion depth, and timed execution truncation to prevent unreasonable response times of embedded code. See, for example Gal [2006]; Gal et al. [2009]; Sol et al. [2011] and the references therein for further details. The Opcode dynamic analysis implicitly benefits from all of these runtime opti-

mizations. Other pragmatic enhancements to the dynamic approach such as automatically enforcing a maximum length for any given Opcode call trace is also possible, but not explored further in this work.

As shown in Algorithm 5, feature extraction uses a hash table containing at most $S \in \mathcal{O}\big(\sum_{\ell}(m - n + 1)\big)$ elements, where ℓ is the number of training examples, and $S < \Phi^n$. Given a training dataset comprised of s non-zero features per example, and ℓ training examples, a linear SVM can be trained in $\mathcal{O}(s\ell)$ time [Joachims, 2006]. The linear SVM classifier performs at most S additions, and multiplications for each classification. Therefore, the worst case runtime and space efficiency for feature extraction and classification using linear SVM is $\mathcal{O}(S) + \mathcal{O}(m)$ for each Opcode call trace evaluated.

5.4 Evaluation

In this section, the effectiveness and efficiency of the proposed method is evaluated, and compared with existing state of the art drive-by download detection techniques. The dataset used in experiments, and the experimental methodology is presented first.

5.4.1 Datasets

To the best of our knowledge, no publicly available gold standard test collection for JavaScript malware detection exists. Therefore, a benign webpage dataset, and a malicious webpage dataset were constructed. The benign dataset contains landing pages for $10,620$ sites listed as most popular by *Alexa*[2], collected in June 2011. Therefore, the dataset is representative of many presently used JavaScript examples. Each webpage of the benign dataset was validated with Google Safe Browsing API and standard antivirus solutions. The malicious webpage dataset was created by downloading URLs and using samples published in different malware forums. See Appendix A for a list of malware forum sites referred. The downloaded webpages from these URLs may not contain drive-by download attacks, due to transient and cloaking techniques employed by malicious sites. Therefore, the Google Safe Browsing API and antivirus solutions were used to flag pages believed to be malicious. To avoid false positives, only the pages identified as malicious by at least 5 screening sources were marked as malicious. A total of 57 distinct attack vectors were identified in the collection.

Note that the use of antivirus solutions may remove the latest attacks from the dataset. But, this

[2]http://www.alexa.com/topsites

is not problematic as the objective is not to find the latest attacks, but to evaluate classification algorithms that do not contain domain specific intervention. Each data point is verified to contain a unique, complete exploit, or pre-attack JavaScript routines using attack categories given by the antivirus solutions and manual examination of the deobfuscated JavaScript code. If more than one page contained the same attack vector, it was removed from the collection. The goal of the pruning is to ensure that the classification makes accurate predictions on unseen exploits. If only one representative of each exploit exists, it cannot appear in both the training and test partitioning, and one can be confident that the predictive power of the algorithm is not biased. The data points containing pre-attack JavaScript routines are included in order to identify an attack as early as possible, making intervention at runtime more tractable.

5.4.2 Experimental Methodology

For evaluation, 10 successive stratified 10-fold cross validations are used. The dataset is randomly divided into 10 equal partitions, maintaining the same proportion of benign to malicious samples in each partition. Then, ten runs of 1 test : 9 training classifications are ran, effectively generating a test classification for each item in the dataset. The procedure is repeated 10 times. The same cross validation splits of the dataset are used to evaluate alternative drive-by download detection techniques.

5.4.3 Parameter Selection for Opcode Analysis

Opcode analysis depends on certain design parameters for good effectiveness and efficiency. Some of these design parameters require empirical analysis and are outlined here:

- Different data mining algorithms can be utilized for classification. Figure 5.10 presents the algorithmic effectiveness for Opcode analysis with a Naïve Bayes classifier, a J48 decision tree classifier and a linear SVM classifier for varying n between 1 and 4 determined experimentally using a single stratified 10-fold cross validation. Loops in JavaScript code cause certain set of n-grams to appear together in the vector space for Opcode analysis. The effectiveness for Naïve Bayes classifier is low due to these non-independent features. J48 performed well when the sliding window length is 2, and a similar effectiveness is observed with linear SVM for n between 2 and 4. The training time complexity of a balanced J48 classifier is $\mathcal{O}\big(S\ell^2 log\ \ell\big)$, where S is the number of features, and ℓ is the size of the training dataset [Marsland, 2009]. Therefore, training a J48 classifier is time consuming on large feature spaces, compared to a

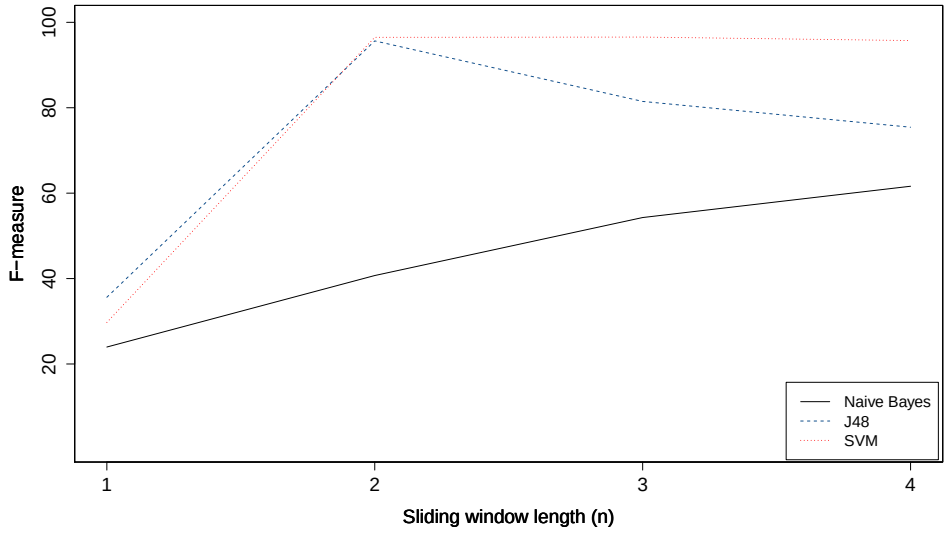


Figure 5.10: Sliding window length (n) and algorithmic effectiveness of Opcode *analysis for alternative classification algorithms using a stratified* 10*-fold cross validation on the dataset given in Section 5.4.1 for any n value between 1 and 4.*

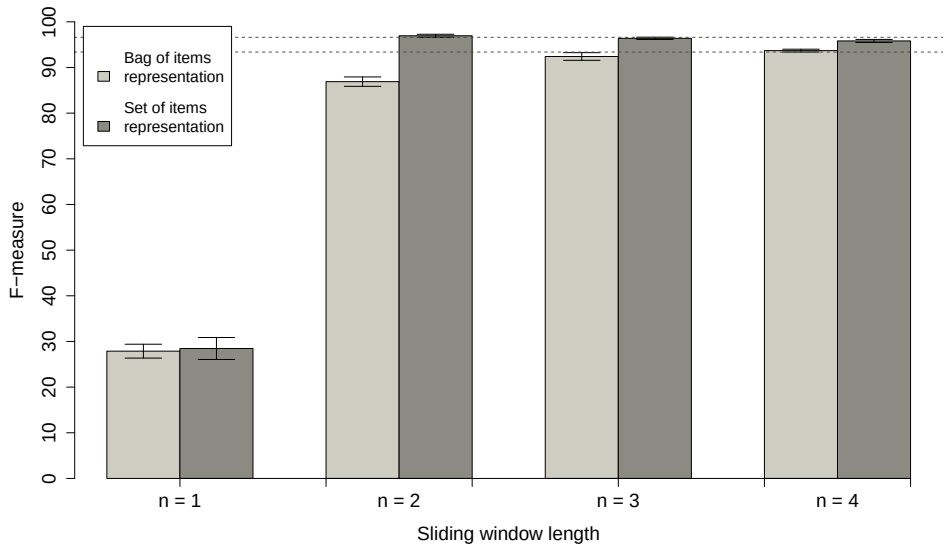


*Figure 5.11: Classification effectiveness (*F-measure*) and corresponding 95% confidence interval for "bag of items" and "set of items" feature representations and for any n value between 1 and 4 using a* linear SVM *classifier on the dataset described in Section 5.4.1.*

linear SVM classifier. Hence, a linear SVM classifier is used in rest of the experimentation.

- Recall that a classifier learned from data can move from high bias to high variance as the complexity of the classifier increases. Small values of n can result in high bias due to the lack of predictive signals, while large values of n can increase variance due to overfitting. Increasing n creates an exponentially larger feature space. Therefore, the smallest sliding window length with the desired accuracy can be determined empirically on the training dataset. In prior work, feature values were represented by the presence or absence of a feature in the trace (*set of items representation*) [Rieck et al., 2010], or as a frequency weighting scheme, where the number of appearances of a feature in the trace is used (*bag of items representation*) [Hu et al., 2003; Kang et al., 2005]. The bag of items representation can lead to increased variance, while the set of items representation can result in high bias. Figure 5.11 shows the classification effectiveness and 95% confidence interval in terms of F-measure for both representations with a linear SVM classifier on the dataset presented in the Section 5.4.1. The lower margin of the confidence interval is used for each setting to select the most appropriate feature representation and n. The dotted horizontal lines indicate the most effective lower margin of the 95% confidence interval for the two schemes of data representation. The results indicate that the set of items representation with a sliding window of length 2 produces good overall effectiveness.

5.4.4 Baselines

The JavaScript malware detection techniques ADSandbox [Dewald et al., 2010] and CUJO [Rieck et al., 2010] algorithms were re-implemented for the test environment as baselines. Furthermore, PhoneyC [Nazario, 2009] was installed using the latest code available at the time of the experiments. Since several of the baselines were implemented independently, the details of baselines are outlined.

- ADSandbox employs a misuse detection approach with regular expression signatures to find malicious webpages. Since the original signatures are not available, a sliding window of length n is parsed over the training data partition, and n-grams unique to malicious webpages are used as signatures to classify items in the test data partition.

- In addition to the JavaScript interpreter, CUJO and ADSandbox require a virtual browser environment to properly execute the embedded JavaScript. The completeness of virtual browser environment is a tradeoff between performance and effectiveness. In the re-implementation of the algorithms, a near complete virtual browser environment is used to reduce JavaScript errors

Algorithm	n	True positive rate (Recall) (%)	False positive rate (%)	Precision (%)	F-measure (%)
Opcode analysis	2	97.37	0.02	96.53	96.94‡
ADSandbox	1	80.70	2.09	17.52	28.78
CUJO static	4	69.65	0.07	85.45	76.72
CUJO dynamic	1	61.23	0.05	88.33	72.30
CUJO ensembled		80.88	0.10	80.21	80.52
PhoneyC		31.58	0.0	100.0	48.00

Table 5.1: Effectiveness of Opcode *analysis and existing state of the art detection methods on the dataset described in Section 5.4.1. The values of n presented achieve the highest accuracy for any n value between 1 and 4. ‡ represents a significant improvement ($p < 0.01$) in* F-measure *compared to all other methods.*

that might otherwise arise.

- Similar to Opcode analysis, the best sliding window length for CUJO static, CUJO dynamic, and ADSandbox algorithms were determined empirically in order to maximize the effectiveness.

5.4.5 Results

Table 5.1 presents the sliding window length (n) and the corresponding effectiveness for all four approaches on the dataset described in Section 5.4.1. The given sliding window lengths produce the best results under successive stratified 10-fold cross validations. The results demonstrate a significantly better effectiveness when evaluated with F-measure for Opcode analysis compared to all other methods on the dataset described in Section 5.4.1. The Opcode analysis has a high true positive rate and a very low false positive rate. For intrusion detection applications both of these qualities are important, as a low true positive rate implies a high probability of a compromise, where as a high false positive rate can result in loss of user confidence due to frequent false interruptions.

5.4.6 Discussion

ADSandbox in the test environment relies on dynamically generated signatures from the training dataset, while PhoneyC uses a set of predefined static heuristics to detect malicious shellcode and

Algorithm	n	True positive rate (Recall) (%)	False positive rate (%)	Precision (%)	F-measure (%)
Opcode analysis	2	100.00	0.01	98.28	99.13‡
CUJO static	4	78.42	0.03	93.19	85.15
CUJO dynamic	3	76.31	0.04	91.26	83.10

Table 5.2: Sliding window length (n) and algorithmic effectiveness with 10 $\times$ stratified 10*-fold cross validation for the* Opcode *and* CUJO *algorithms on the combined benign dataset described in Section 5.4.1 and the malicious dataset comprised of more than one sample of the same antivirus category.* ‡ *represents a significant improvement ($p < 0.01$) in* F-measure *compared to all other methods.*

heap spray attacks. As such, PhoneyC did not detect malicious activity in many of the test samples, especially if the sample contained only pre-attack JavaScript routines. The low detection rate of ADSandbox and PhoneyC highlight the difficulty of requiring manual generation of attack vector signatures, and the inability of these approaches to make accurate predictions on previously unseen attack types.

CUJO does not rely on signatures, but did not perform as well as anticipated in the test environment. A high statistical variance in accuracy between different datasets, and a lower accuracy for dynamic analysis than static analysis on most of the datasets was originally observed in the experiments presented by Rieck et al. [2010]. Since the original experiments of Rieck et al. [2010] included multiple instances of each attack category, the dataset is reconstructed to include multiple malicious sample pages from each attack category to ensure the consistency of the baseline implementation.

Table 5.2 shows the effectiveness of Opcode and CUJO algorithms on the combined benign dataset described in Section 5.4.1 and a dataset containing multiple malicious instances of each attack type. On this dataset the CUJO static and dynamic algorithms show a dramatic increase in true positive rate and a reduction in the false positive rate, and provide similar results to the original experiments performed by Rieck et al. [2010]. All machine learning algorithms show an improvement in effectiveness on this dataset. This exemplifies how measured effectiveness of data mining algorithms are augmented on unfiltered datasets consisting of more than an attack from the same category.

In Figure 5.12 is the effectiveness of CUJO dynamic algorithm on the dataset given in Section 5.4.1. The CUJO dynamic achieved the best effectiveness when the sliding window length is 1, and the detection rate decreased with increasing n. Having the best detection rate at a smaller sliding window length, compared to the original experiment [Rieck et al., 2010], and the deterioration of the effectiveness along with increasing sliding window lengths (i.e. increasing complexity of the

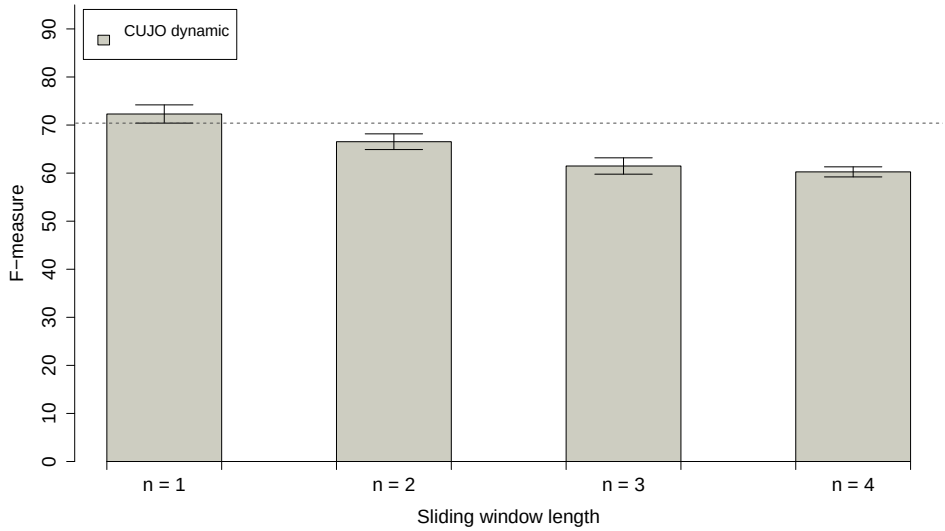


*Figure 5.12: Classification effectiveness (*F-measure*) for* CUJO *dynamic algorithm for varying sliding window lengths (n).*

machine learning algorithm), can be indicative of the CUJO dynamic algorithm's effort to reduce variance on a dataset with unique malicious samples.

An examination of a few malicious examples that were classified by the CUJO dynamic algorithm as benign revealed that the main reason for erroneous classification of those malicious items is the sample specific values (JavaScript variable names, and values) that are present in the malicious samples, but not in the training set. As a result, many of the 3-grams are unique, and therefore only appear in the training or the test partition, but not both. For example, consider the CUJO dynamic trace shown in the Figure 5.6. If the user defined JavaScript variable names and values are not seen in any of the examples in the training dataset, and if a sliding window of length 3 is selected, only "`CALL`, `unescape`, `SET`" 3-gram from the attack code segment would take part in the classification process. In fact, out of all 3-gram features in malicious instances, 40% contained a user defined variable name and 98% contained a user defined variable name or value. This results in a lack of predictive features that appear in both training and test partitions.

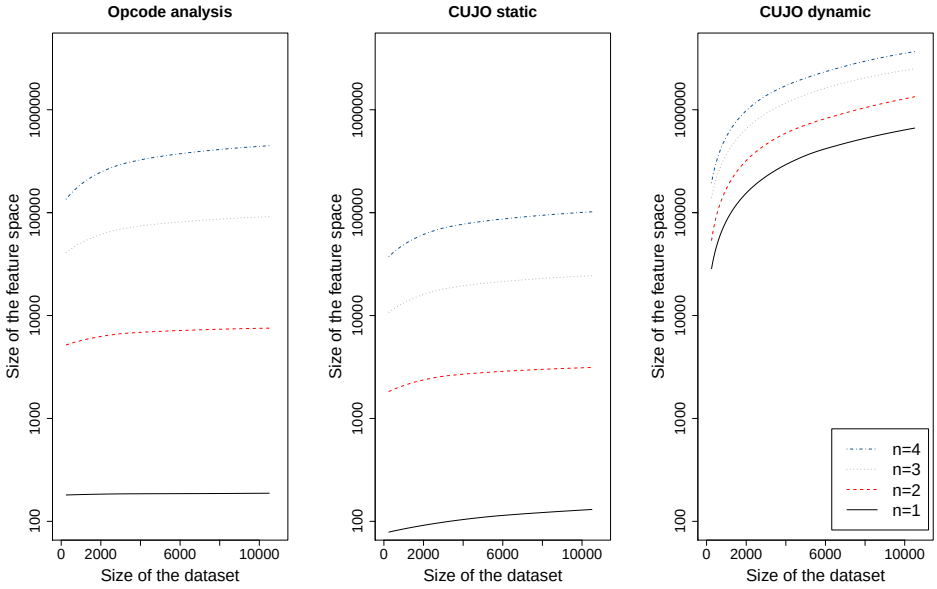


Figure 5.13: Growth in the feature space of Opcode *analysis and* CUJO *algorithms with increasing dataset size for different* n.

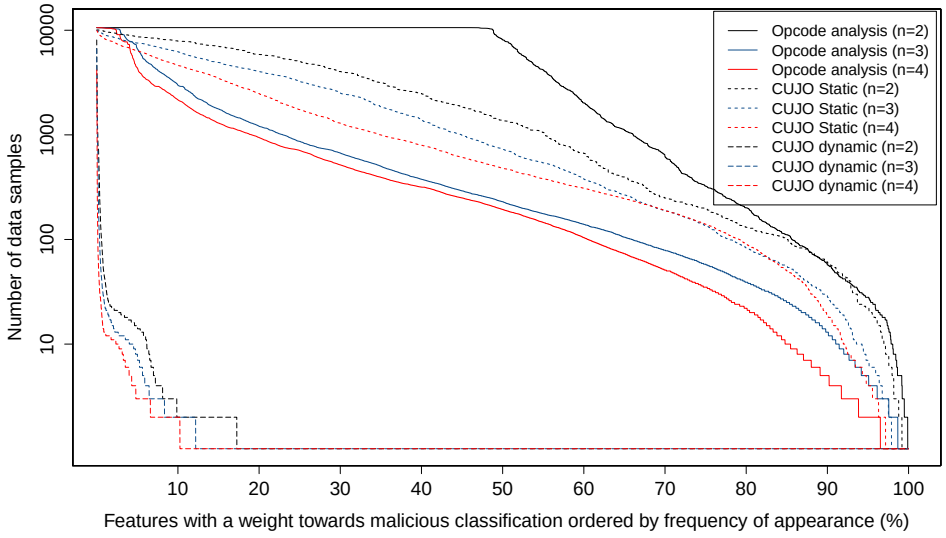


Figure 5.14: Number of training instances per distinct n*-gram.*

Feature Space

The growth of the feature space along with the increasing dataset sizes for the Opcode analysis, and the CUJO static and dynamic algorithms is shown in Figure 5.13. CUJO dynamic has a much larger feature space that grows along with the size of the dataset, which is caused by the inclusion of sample specific values. Figure 5.14 shows the number of data samples in which a feature is present, for features with a weight inclined towards a classification decision as malicious in the weight vector ω, ordered by the count of data samples containing the feature from most frequent to the least. CUJO dynamic demonstrates a high dimensional feature space with low coverage, which increases with n. The high dimensionality with low coverage is problematic for statistical learning methods. As such, the lower detection rate of CUJO algorithm in the current environment can be partially attributed to using a dataset with only unique malicious samples.

CUJO dynamic analysis is promising for predicting "maliciousness" of unseen attacks, but suffers from one major drawback. The elements of feature composition are not fixed. As a result, the CUJO dynamic algorithm generates a large feature space with low coverage. Variable feature composition does not pose a serious limitation if the token space adheres to Heaps law, which states that there will be diminishing returns in terms of new feature discovery along with growing training dataset sizes. But in malware detection, where adversaries actively search for ways to circumvent detection mechanisms, inclusion of sample specific values, especially user defined variable names can cause unexpected behaviour. In fact, in the case of the CUJO dynamic analysis, a considerable portion of the feature space that contribute towards a malicious classification can be avoided just by changing the JavaScript variable names used in the attack. The CUJO static algorithm feature space has higher coverage. But the predictability of the static algorithm is often low due to code obfuscation techniques employed by many adversaries.

The feature space of the Opcode method only contains Opcode function calls, where the number of Opcode functions (Φ) is fixed. As a result, the feature space automatically extracted from the training dataset is generic and can never grow larger than Φ^n. In contrast, the CUJO dynamic algorithm allows variable names and assignments to be valid tokens in the n-gram feature space [Rieck et al., 2010]. Moreover, CUJO augments the automatically generated feature space with a pre-matching filter of previously identified n-grams previously known to precede specific attacks. However, the prematching requires domain specific knowledge, and is equivalent to using signatures to identify well-known attack vectors. The Opcode method uses no domain specific pre-filtering, and automatically extracts the feature space from the training dataset, and requires no user intervention.

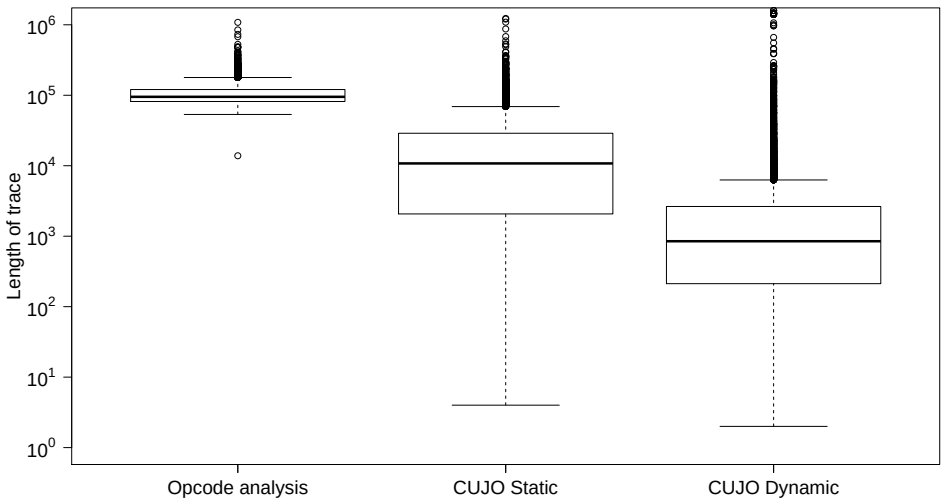


Figure 5.15: Lengths of trace for Opcode *analysis and* CUJO *static/dynamic algorithms for the dataset given in Section 5.4.1.*

Efficiency

For simplicity, and to illustrate benefits and tradeoffs of Opcode analysis, CUJO static and dynamic analysis logs are presumed to be generated from the same browser emulation process. This allows the comparison of additional time and space used by the core of the anomaly detection algorithm, which is the feature extraction and classification. Recall that the worst case runtime and space efficiency for feature extraction and classification using linear SVM is $\mathcal{O}(S) + \mathcal{O}(m)$. Due to sparseness of the vector space for a typical webpage, the average time spent on feature extraction and classification primarily depends on the length of the trace processed. The space usage is governed by the sizes of the token index, the hash table mapping tokens to n-gram features, and the trace processed. The length of the traces processed by Opcode analysis and CUJO static and dynamic methods for the dataset described in Section 5.4.1 are illustrated in Figure 5.15. The Opcode analysis processes a larger trace compared to the other two approaches. Figure 5.16 shows the effectiveness versus mean space usage on disk incurred on feature extraction and classification for Opcode analysis, CUJO static and CUJO dynamic analysis algorithms on the dataset given in the Section 5.4.1. Please note that the effectiveness is decreasing along the dependent variable in the Figure 5.16. So a point towards bottom left corner of the graph is more desirable. The Opcode analysis has a very low space utilization

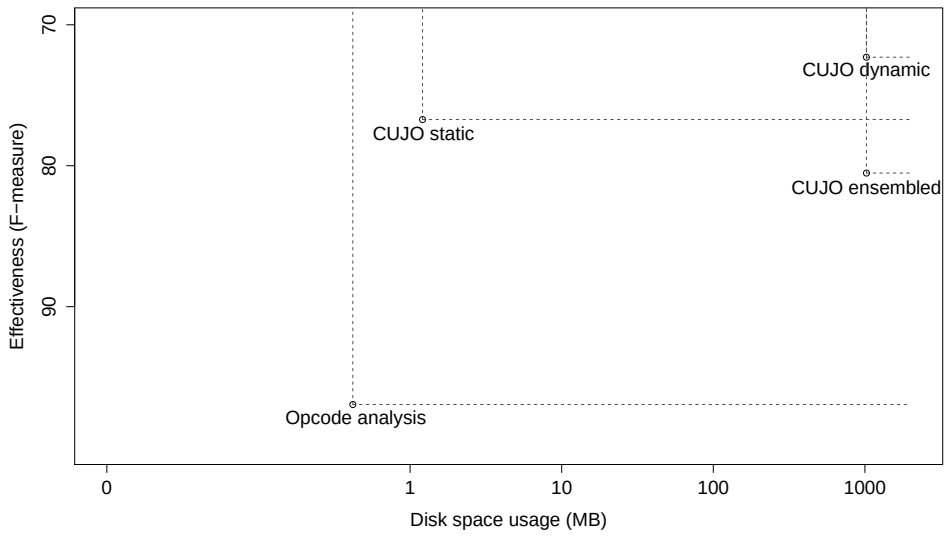


Figure 5.16: Effectiveness vs space usage for feature extraction and classification using Opcode*,* CUJO *static and* CUJO *dynamic algorithms for the dataset given in Section 5.4.1.*

compared to the CUJO algorithm, mainly because the derived features in an Opcode trace have a fixed vocabulary, but the feature space of CUJO dynamic grows with the number of training instances. Use of an integer token space in Opcode analysis instead of a text token space lowers the space usage relative to the CUJO static algorithm. As the graphs reveal, Opcode analysis trades time efficiency for better effectiveness and lower space utilization.

Training Data Requirements

Figure 5.17 show the impact of increasing training instances on effectiveness for the CUJO and Opcode algorithms. In Figure 5.17 (top), $10\times$stratified 10-fold cross validation is used, and therefore a minimum of 10 malicious instances are required – one for each fold. The Opcode method has a remarkably stable learning curve, while CUJO algorithms benefit from increasing the number of training instances. Therefore, in Figure 5.17 (bottom) a straight 1:2 test and train partitioning of malicious dataset is evaluated to determine exactly how sensitive the Opcode method is to malicious training instances. The entire benign dataset is included in the training partition and malicious instances in the training partition is incremented by 2 samples at a time. This procedure is repeated with random partitioning of the malicious dataset for 10 times and the results are averaged. The Opcode

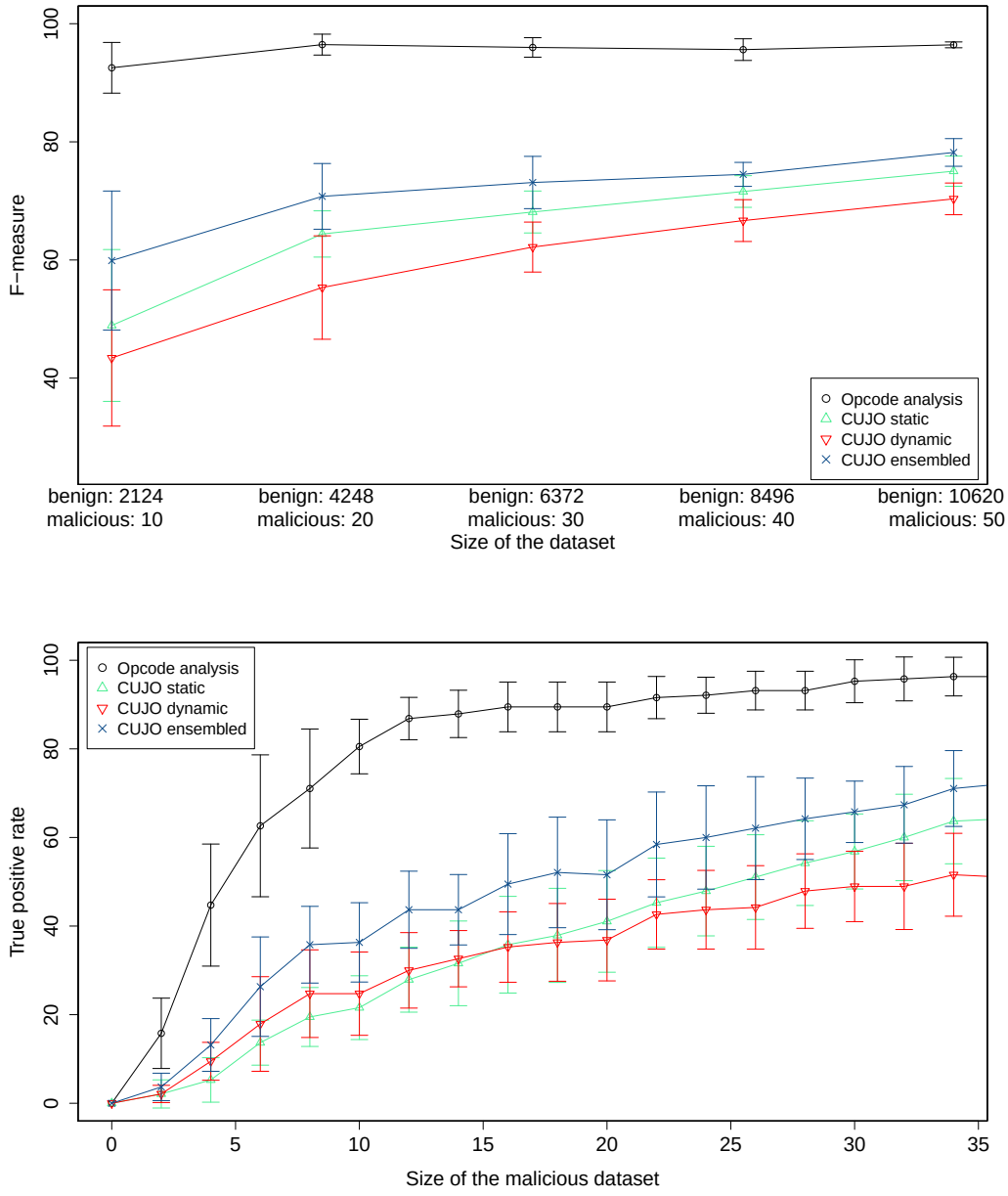


Figure 5.17: The effect of increasing the number of training instances for Opcode *analysis, and* CUJO *on* F-measure *and the* true positive rate. *The learning curve on the top uses proportional growth of malicious to benign training instances, while the curve on the bottom uses the entire benign dataset and* $2/3$*s of the malicious samples for training, and remaining* $1/3$ *of the malicious samples constitute the test dataset.*

Weight	Feature	% appearance in benign samples	% appearance in malicious samples
0.0899	`setlocal, moreiter`	75.5	100.0
0.0897	`forlocal, getarg`	88.4	100.0
0.0821	`int8, bitor`	72.5	98.2
0.0821	`uint16, zero`	72.6	98.2
0.0821	`bitor, int8`	72.4	98.2
0.0821	`getthisprop, setelem`	72.8	98.2
0.0814	`goto, js_true_str`	75.3	98.2
0.0808	`setelem, try`	72.8	98.2
0.0804	`getprop, setelem`	73.8	98.2
0.0797	`getelem, length`	80.1	98.2

Table 5.3: Top 10 weights towards malicious classification for Opcode *analysis and their coverage in the dataset presented in the Section 5.4.1*

.

algorithm is remarkably effective with very few malicious training instances showing low training data requirement and high generalisability of the Opcode algorithm.

Evasion

Table 5.3, shows the 10 features with the highest weight towards a malicious classification in Opcode analysis. As shown in the Figure 5.14 and the table, the Opcode analysis uses a feature space with a high coverage. Therefore, Opcode analysis is less susceptible to attacks that attempt to avoid features with a weight toward a malicious classification. However, in system call analysis, methods to mount attacks by executing legitimate system call sequences have been shown to be effective in prior work [Mutz et al., 2006]. Similar strategies could be used against Opcode analysis. Retraining the algorithm with an up-to-date dataset, and including features derived from arguments passed to Opcode function calls may be an option to reduce the likelihood that such attack would circumvent detection.

5.5 Effectiveness and Time Efficiency Tradeoffs

The proposed approach for dynamic detection of new drive-by download attacks is effective and space efficient, but with a higher average run-time compared to the baseline methods. The time efficiency, which is primarily affected by length of the Opcode function call trace may be improved by randomly sampling n-grams from the log trace with a probability of p. Hence, from an Opcode function call sequence of length m, approximately $(m - n + 1) \times p$ n-grams are selected and processed. Both test and training data are randomly sampled. The new workflow is illustrated in Figure 5.18, and the modified pseudocode for data transformation is shown in Algorithm 7.

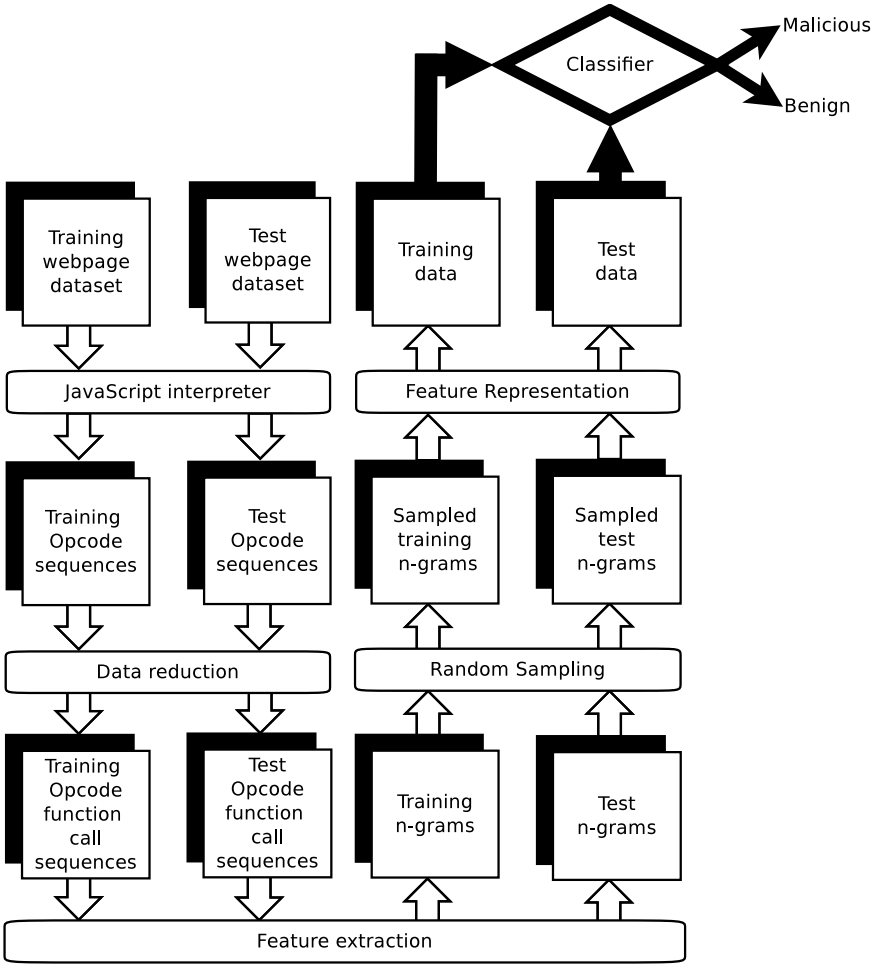


Figure 5.18: Workflow for trading off effectiveness to improve time efficiency of Opcode *analysis. The workflow consists of: data reduction, feature extraction, random sampling, feature representation, and classification.*

ALGORITHM 7 Transformation of an Opcode trace into a vector for classification with random sampling of n-grams.

```
// Input: t_i ← Opcode function call trace for the i·th webpage.
// Input: ht ← a hash table mapping corresponding n-grams to features.
x_i ← new Array[1 ... length(ht.keys())]
sliding_window ← new Queue()
for function_call in t_i do
    sliding_window.queue(function_call)
    if sliding_window.size() == n then
        if randomly_sampled then
            x_i[ht.lookup(sliding_window)] = 1
        end
        sliding_window.dequeue()
    end
end
return x_i
```

When effectiveness is traded off for better efficiency, the minimum degradation (δ) in mean effectiveness that is considered as consequential must be determined. If the confidence interval for the mean effectiveness is above this degradation, the sampling based solution is not worse than δ at the given level of confidence. Smaller sampling probabilities improve efficiency. Therefore, the smallest sampling probability having a confidence interval above δ is chosen.

However, deriving confidence bounds for mean effectiveness is not as straight forward as it might initially seem. Random sampling of n-grams makes the classifier have varying outputs for the same data input. When comparing such classifiers, does the effectiveness of the classifier significantly differ across the population of data inputs as well as on all possible repeated observations for the same data input? Recall that the standard evaluation methods only support one source of variability. In this instance, it is the variance in effectiveness on different stratified 10-fold cross validation data partitions. A conclusion derived with a $10\times$ stratified 10-fold cross validation using standard evaluation methods can be contradicted by another evaluation using the same partitions of the dataset due to sampling of the log trace. This is illustrated in Figure 5.19 using 10 repeated comparisons for each $10\times$ stratified 10-fold cross validation data partitions. Therefore, the question is answered by repeatedly observing the effectiveness for each of 10 stratified 10-fold cross validation partitions of the dataset, and performing a multivariate test of significance. With repeated observations the variations due to data partitions, and some classifiers performing better on certain data partitions than the others (the interactive effect between different classifiers and various data partitions) can be observed using

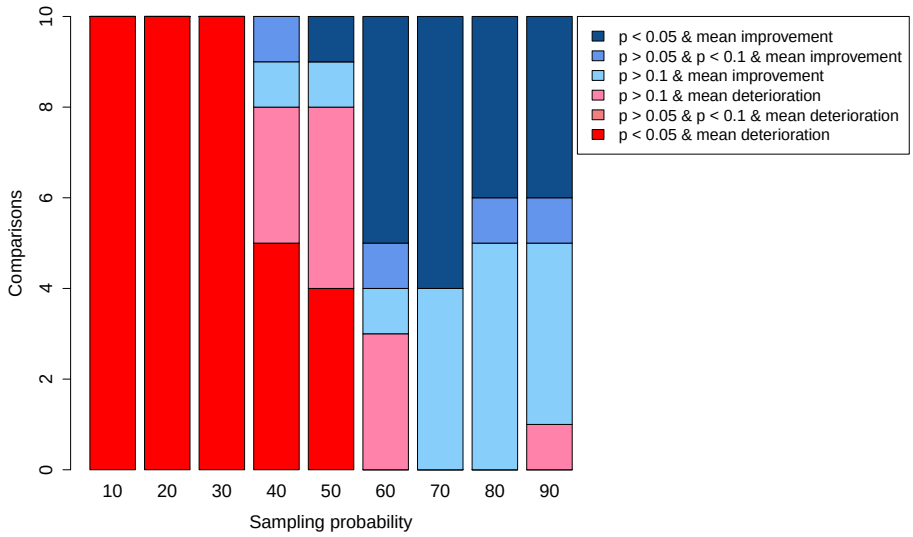


Figure 5.19: The p values for 10 comparisons of Opcode *analysis employing sampling of log trace with complete* Opcode *analysis using the same $10\times$ stratified 10-fold cross validation partitions of the dataset described in Section 5.4.1 at varying probabilities of sampling the log trace.*

the following linear mixed effect model:

$$y_{ijo} = \gamma + c_i + v_j + cv_{ij} + \varepsilon_{ijo}. \tag{5.2}$$

Here, each level i of factor c_i represents classifiers with a given probability of a sampling log trace, and each level j of factor v_j is a random stratified 10-fold cross validation partition of the dataset. Note that the same set of $10\times$ stratified 10-fold cross validation partitions of the dataset are used to assess classifiers at all sampling probabilities of the log trace. The interactive effects between classifiers and data partitions is captured in cv_{ij}. Here y_{ijo} and ε_{ijo} is the observed effectiveness and the model error for classifier i and a stratified 10-fold cross validation data partition j on the $o{\cdot}th$ repeated assessment. The factor c_i produces a fixed effect in the linear mixed effect model, while the other factors cause random effects.

The impact on effectiveness due to random sampling of n-grams is analysed in Figure 5.20. Here, Opcode analysis techniques with varying sampling probabilities of log trace are compared with the complete Opcode trace analysis. Although, a deterioration in effectiveness is expected with reduced sampling probabilities, a surprising significant improvement is observed for higher sampling rates. This can be due to reduction of noisy features for classification. As exemplified in Figure 5.3,

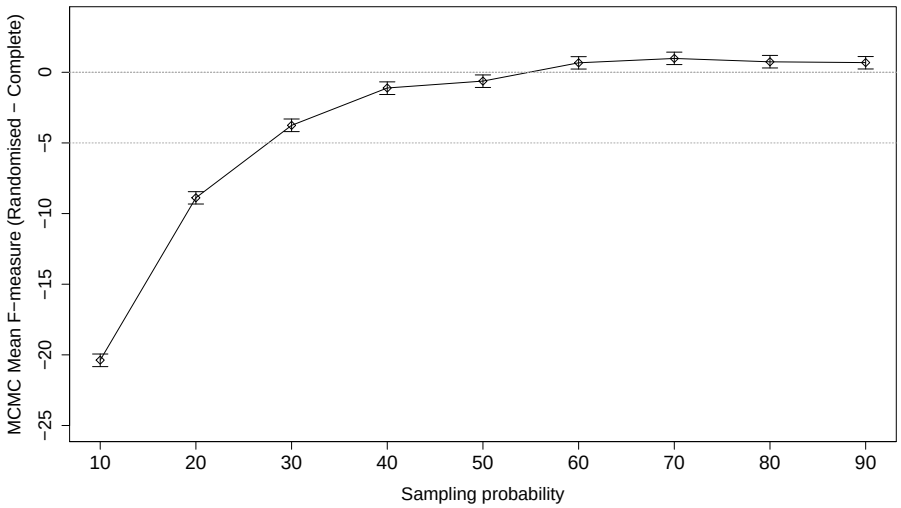


Figure 5.20: The 95% HPD interval for comparing Opcode *analysis processing complete log trace with* Opcode *analysis using varying probabilities of sampling log trace on the dataset described in Section 5.4.1.*

JavaScript drive-by download attacks often use loops to expand obfuscated code and to prepare for the attack. Loops cause the same malicious n-grams to repeat in the Opcode log trace. Therefore, high sampling rates cause a reduction in noisy features, while retaining the predictive malicious features. Such significant improvement demonstrates the potential for further improving effectiveness using better feature selection [Guyon and Elisseeff, 2003; Guyon, 2006; Liu and Motoda, 1998; 2008]. However, feature selection is not explored further in this work. As malicious n-grams are repeated many times in the log trace, deterioration in effectiveness is small compared to the reduction in probability of sampling log trace. As a consequence, run time overhead can be improved without drastically reducing effectiveness. For example, in the above instance the sampling probability can be reduced to 30% with a less than 5% degradation in F-measure at a confidence level of 95%. A sample rate of 30% reduces the length of a log trace by 70%, and causes approximately a similar improvement in time efficiency.

5.6 Summary

In this chapter, how automatically generated unbounded feature spaces and non-curated datasets

with a small number of examples from one class can lead to seemingly high effectiveness and incorrect conclusions is demonstrated using a case study of dynamically detecting drive-by download attacks. This illustrates the danger of considering evaluation as a blackbox when evaluating data mining tasks. Opcode analysis with an upper bounded feature space is proposed as a viable solution for the dynamic detection of drive-by download attacks. Opcode analysis is effective and has a low space utilization. The low training data requirements combined with automatic feature extraction make Opcode analysis an attractive alternative for detecting new drive-by download attacks.

However, the Opcode analysis has a higher runtime overhead compared to the baseline dynamic anomaly detection techniques. The runtime overhead of Opcode analysis is improved using sampling of n-grams from the log trace. Sampling causes a classifier to be non-deterministic. Ignored repeated observations can lead to incorrect conclusions. How a multivariate linear model test is applied for a non-deterministic classifier is shown. The effectiveness is traded off for efficiency using the proposed test. The runtime overhead of Opcode analysis can be reduced by 70% for a less than 5% reduction in effectiveness at a 95% level of confidence.

Chapter 6

Conclusions

This thesis illustrates the importance of reducing, and accurately accounting for uncertainty when evaluating complex IR and classifier systems. Many researchers are used to the standard blackbox style evaluation of experiments, where performance on a sample is presented for a judgement. The fundamental assumption in standard evaluation is no other uncertainty other than the one due to variance in performance for experimental units exists. However, this assumption is violated in many modern IR and classification experiments, where a considerable portion of uncertainty is from more than one source. In this thesis, uncertainty due to test collection bias in IR experiments, algorithmic variance due to non-determinism in algorithms, and overfitting of classification algorithms have been analysed.

An important aspect derived from this work is the recognition of multidimensional variance, which occurs when more than one type of experimental unit are sampled for evaluation. An example is sampling of system instances, and topics for evaluating non-deterministic IR systems. Another form of variance is due to repeated observations for the same experimental unit, caused by inconsistent experimental conditions. Therefore, the variance can be subdivided into many-dimensions with the possibility of repeated observations for each dimension. Observing effectiveness for each topic on each IR system instance from a non-deterministic IR system by many users, or for varying document sets is an example of the above scenario. Hence, the steps to remember for accurate evaluation of complex IR and classifier systems are:

1. Reduce the uncertainty that arises from each source;

2. Recognise the dimensions / repeated observations of variance; and

3. Evaluate with provisions for each source of variance.

6.1 Thesis Summary

In this thesis, the non-suitability of standard evaluation when comparing complex IR and classifier experiments have been illustrated. How researchers may be led to draw incorrect conclusions by high variance classifiers, ignoring dimensions in IR experiments, and ignoring repeated observations in classification experiments have been shown.

An area of focus in this thesis is reducing uncertainty due to the test collection bias of IR experiments. The ability to reduce uncertainty by predicting relevance of retrieved unjudged documents using existing methods in the context of graded relevance judgements have been illustrated. Furthermore, a fully automated approach for constructing pools for judging documents has been proposed. The proposed method can be used to find a considerable proportion of relevant documents that were previously pooled via manual methods, and has the potential of finding documents that are not found in current pooling approaches.

In this thesis, the variance due to repeated observations are incorporated into evaluation. A novel bootstrap approach that accounts for variance due to repeated observations is proposed, and used to compare approaches to predict the relevance of unjudged documents in retrieval results. The applicability of multivariate linear model tests for evaluating classification experiments with repeated observations have also been shown. Furthermore, variance from two dimensions have been integrated into evaluation. Two novel tests using bootstrapping, and multivariate linear modeling are proposed to evaluate non-deterministic:deterministic IR system comparison. Both approaches make similar inferences. A novel multivariate linear model test for non-deterministic:non-deterministic IR system comparison is presented. A statistically viable method for comparing two systems for similarity, when at least one system is non-deterministic is also proposed. Using this method, an elegant solution for comparing effectiveness-efficiency tradeoffs, when at least one system is non-deterministic is presented. The proposed solution is used for several different effectiveness-efficiency tradeoff comparisons in IR and classification experiments.

6.2 Future Work

A number of opportunities for future research have stemmed from the work presented in this thesis. The approaches using only automatic runs has the potential of finding about 88% of the documents

previously only found via manual runs when judging a pool. Hence, a fully automated approach for constructing pools for judging has more room for improvement.

Many directions for future work can be derived from the proposed significance tests. A bootstrap solution for non-deterministic:non-deterministic IR system comparison is not proposed in this work. Further, the proposed approaches can be extended to complex problems that have variability from many more dimensions, and repeated observations. The above methods may be extended to evaluate systems that vary with similar parameters (query expansion, stemming, term smoothing, etc.) rather than conventional evaluation of individual runs. Each parameter, topics, and interactive effects between them can be incorporated into the linearly model for such evaluation. They can also be used in other domains. For example, the proposed two dimensional significance tests can be extended to evaluate recommender systems where the variance is due to users and products, or personalised web search where variance is from users and topics.

Another unexplored area is the ideal composition of topics and system instances for a non-deterministic IR system comparison. In the thesis, tests were demonstrated using a reasonable number of topics and IR system instances. However, producing IR system instances can be costly. One future research question is: What is the minimum number of IR systems that can be used for a test with a reasonable power? One may use a large number of IR system instances instead when the cost of producing an IR system instance is low, but with a comparatively smaller number of topics. This can also be problematic as now a significance test assumes a higher power due to the large number of observations seen even though it hasn't seen many observations for different topics. What is the ideal ratio between IR system instances and topics is another research question to explore in the future.

Further, current sharding algorithms are complex and contain many tuning parameters. Evaluating such algorithms is difficult and costly. For example, in this research certain parameters were held constant to previously published values. However, these values may not be the optimal settings. Therefore, simpler algorithms with fewer parameters are desirable.

Appendix A

A.1 Sites Referred to Construct the Malicious Dataset

http://www.malwaredomainlist.com/

http://www.malwaredomains.com/

http://sucuri.net/

http://www.blade-defender.org/

http://code.google.com/p/phoneyc/

Bibliography

C. C. Aggarwal. *Data streams: Models and algorithms*, volume 31. Springer, 2007.

S. Ahn, S. H. Park, and K. H. Lee. How to demonstrate similarity by using noninferiority and equivalence statistical testing in radiology research. *Radiology*, 267(2):328–338, 2013.

R. Aly, D. Hiemstra, and T. Demeester. Taily: Shard selection using the tail of score distributions. In *Proceedings of the 36th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '13)*, pages 673–682, Dublin, Ireland, 2013. ACM.

D. R. Anderson, D. J. Sweeney, and T. A. Williams. *Statistics for business and economics*. Cengage Learning, 2008.

C. Andrieu, N. De Freitas, A. Doucet, and M. I. Jordan. An introduction to MCMC for machine learning. *Machine Learning*, 50(1-2):5–43, 2003.

J. A. Aslam and M. Montague. Models for metasearch. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '01)*, pages 276–284. ACM, 2001.

J. A. Aslam and V. Pavlu. A practical sampling strategy for efficient retrieval evaluation. Technical report, College of Computer and Information Science, Northeastern University, May 2007.

J. A. Aslam, V. Pavlu, and R. Savell. A unified model for metasearch, pooling, and system evaluation. In *Proceedings of the 12th Annual International Conference on Information and Knowledge Management (CIKM '03')*, pages 484–491. ACM, 2003.

J. A. Aslam, V. Pavlu, and E. Yilmaz. A statistical method for system evaluation using incomplete judgments. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '06)*, pages 541–548. ACM, 2006.

N. E. Ayat, M. Cheriet, and C. Suen. Automatic model selection for the optimization of SVM kernels. *Pattern Recognition*, 38(10):1733–1745, May 2005.

R. H. Baayen, D. J. Davidson, and D. M. Bates. Mixed-effects modeling with crossed random effects for subjects and items. *Journal of Memory and Language*, 59(4):390–412, 2008.

M. Banko and E. Brill. Scaling to very very large corpora for natural language disambiguation. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics (ACL-2001)*, pages 26–33. Association for Computational Linguistics, 2001.

C. M. Bishop. *Pattern recognition and machine learning*. Springer, 2006.

D. Bodoff and P. Li. Test theory for assessing IR test collections. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '07)*, pages 367–374, Amsterdam, The Netherlands, 2007. ACM.

R. R. Bouckaert. Choosing between two learning algorithms based on calibrated tests. In *Proceedings of the 20th Annual International Conference on Machine Learning (ICML '03)*, pages 51–58, 2003.

R. R. Bouckaert. Estimating replicability of classifier learning experiments. In *Proceedings of the 21st Annual International Conference on Machine Learning (ICML '04)*, page 15. ACM, 2004.

C. Buckley and E. M. Voorhees. Retrieval evaluation with incomplete information. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '04)*, pages 25–32. ACM, 2004.

C. Buckley, D. Dimmick, I. Soboroff, and E. Voorhees. Bias and the limits of pooling for large collections. *Information Retrieval*, 10(6):491–508, 2007.

C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *Proceedings of the 22nd Annual International Conference on Machine Learning (ICML '05)*, pages 89–96. ACM, 2005.

S. Büttcher, C. L. A. Clarke, and I. Soboroff. The TREC 2006 terabyte track. In *Proceedings of the 15th Text REtrieval Conference (TREC '06)*, volume 6, page 39, 2006.

S. Büttcher, C. L. A. Clarke, P. C. K. Yeung, and I. Soboroff. Reliable information retrieval evaluation with incomplete and biased judgements. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '07)*, pages 63–70. ACM, 2007.

J. Callan. Distributed information retrieval. *Advances in Information Retrieval*, pages 127–150, 2002.

J. Callan and M. Connell. Query-based sampling of text databases. *ACM Transactions on Information Systems (TOIS)*, 19(2):97–130, April 2001.

J. Callan, M. Connell, and A. Du. Automatic discovery of language models for text databases. In *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data (SIGMOD '99)*, pages 479–490, Philadelphia, Pennsylvania, USA, June 1999. ACM.

D. Canali, M. Cova, G. Vigna, and C. Kruegel. Prophiler: A fast filter for the large-scale detection of malicious web pages. In *Proceeding of the 20th Annual International Conference on World Wide Web (WWW '11)*, pages 197–206, Hyderabad, India, March 2011.

B. Carterette. Robust test collections for retrieval evaluation. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '07)*, pages 55–62. ACM, 2007.

B. Carterette. On rank correlation and the distance between rankings. In *Proceedings of the 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '09)*, pages 436–443, Boston, MA, USA, 2009. ACM.

B. Carterette. Model-based inference about IR systems. In *Advances in Information Retrieval Theory*, pages 101–112. Springer, 2011.

B. Carterette and J. Allan. Semiautomatic evaluation of retrieval systems using document similarities. In *Proceedings of the 16th Annual International Conference on Information and Knowledge Management (CIKM '07)*, pages 873–876. ACM, 2007.

B. Carterette and M. D. Smucker. Hypothesis testing with incomplete relevance judgments. In *Proceedings of the 16th Annual International Conference on Information and Knowledge Management (CIKM '07)*, pages 643–652. ACM, 2007.

B. Carterette and I. Soboroff. The effect of assessor error on IR system evaluation. In *Proceedings of the 33rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '10)*, pages 539–546, Geneva, Switzerland, 2010. ACM.

B. Carterette, J. Allan, and R. Sitaraman. Minimal test collections for retrieval evaluation. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '06)*, pages 268–275. ACM, 2006.

B. Carterette, V. Pavlu, E. Kanoulas, J. A. Aslam, and J. Allan. Evaluation over thousands of queries. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '08)*, pages 651–658. ACM, 2008.

B. Carterette, E. Gabrilovich, V. Josifovski, and D. Metzler. Measuring the reusability of test collections. In *Proceedings of the 3rd ACM International Conference on Web Search and Data Mining (WSDM '10)*, pages 231–240. ACM, 2010a.

B. Carterette, E. Kanoulas, V. Pavlu, and H. Fang. Reusable test collections through experimental design. In *Proceedings of the 33rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '10)*, pages 547–554. ACM, 2010b.

B. Carterette, E. Kanoulas, and E. Yilmaz. Simulating simple user behavior for system effectiveness evaluation. In *Proceedings of the 20th ACM Annual International Conference on Information and Knowledge Management (CIKM '11)*, pages 611–620, Glasgow, Scotland, UK, 2011. ACM.

R. L. Chambers and C. J. Skinner. *Analysis of survey data*. John Wiley & Sons, 2003.

M. Chen and Q. Shao. Monte carlo estimation of Bayesian credible and HPD intervals. *Journal of Computational and Graphical Statistics*, 8(1):69–92, 1999.

S. F. Chen and J. Goodman. An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th Annual Meeting on Association for Computational Linguistics (ACL '96)*, pages 310–318, Santa Cruz, California, 1996. Association for Computational Linguistics.

M. R. Chernick, V. K. Murthy, and C. D. Nealy. Application of bootstrap and other resampling techniques: evaluation of classifier performance. *Pattern Recognition Letters*, 3(3):167–178, 1985.

C. L. A. Clarke, N. Craswell, I. Soboroff, and E. M. Voorhees. Overview of the TREC 2011 web track. In *Proceedings of the 20th Text REtrieval Conference (TREC '11)*, 2011.

C. L. A. Clarke, N. Craswell, and E. M. Voorhees. Overview of the TREC 2012 track. In *Proceedings of the 21st Text REtrieval Conference (TREC '12)*, 2012.

C. W. Cleverdon. The significance of the cranfield tests on index languages. In *Proceedings of the 14th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '91)*, pages 3–12. ACM, 1991.

N. Cliff. *Ordinal methods for behavioral data analysis*. Psychology Press, 2014.

W. G. Cochran. *Sampling techniques*. John Wiley & Sons, 2007.

J. Cohen. *Statistical power analysis for the behavioral sciences*. Routledge Academic, 2013.

K. Collins-Thompson, P. Bennett, F. Diaz, C. L. A. Clarke, and E. M. Voorhees. Overview of the TREC 2013 web track. In *Proceedings of the 22nd Text REtrieval Conference (TREC '13)*, 2013.

G. V. Cormack, C. R. Palmer, and C. L. A. Clarke. Efficient construction of large test collections. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '98)*, pages 282–289. ACM, 1998.

M. Cova, C. Kruegel, and G. Vigna. Detection and analysis of drive-by-download attacks and malicious JavaScript code. In *Proceeding of the 19th Annual International Conference on World Wide Web (WWW '10)*, pages 281–290, Raleigh, North Carolina, USA, April 2010.

W. B. Croft, D. Metzler, and T. Strohman. *Search engines: Information retrieval in practice*. Addison-Wesley Reading, 2010.

C. Curtsinger, B. Livshits, B. Zorn, and C. Seifert. ZOZZLE: Low-overhead mostly static JavaScript malware detection. In *Proceeding of the 20th USENIX Security Symposium (USENIX '11)*, San Francisco, CA, USA, August 2011.

O. Dalle. On reproducibility and traceability of simulations. In *Proceedings of the 2012 Winter Simulation Conference (WSC)*, pages 1–12. IEEE, 2012.

M. Datar, A. Gionis, P. Indyk, and R. Motwani. Maintaining stream statistics over sliding windows: (extended abstract). In *Proceeding of the 13th Annual Symposium on Discrete Algorithms (SODA '02)*, pages 635–644, Philadelphia, PA, USA, 2002.

J. De Beer and M. Moens. Rpref: A generalization of Bpref towards graded relevance judgments. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '06)*, pages 637–638, Seattle, Washington, USA, 2006. ACM.

W. E. Deming. *Some theory of sampling*. Courier Corporation, 1966.

A. Dewald, T. Holz, and F. C. Freiling. ADSandbox: Sandboxing JavaScript to fight malicious websites. In *Proceeding of the 25th ACM Symposium on Applied Computing (SAC '10)*, pages 1859–1864, Sierre, Switzerland, March 2010.

I. S. Dhillon, J. Fan, and Y. Guan. Efficient clustering of very large document collections. *Data Mining for Scientific and Engineering Applications*, pages 357–381, 2001.

H. Drucker, W. Donghui, and V. N. Vapnik. Support vector machines for spam categorization. *IEEE Transactions on Neural Networks*, 10(5):1048–1054, September 1999.

C. Drummond. Replicability is not reproducibility: nor is it good science. 2009.

B. Efron. Bootstrap methods: another look at the jackknife. *The annals of Statistics*, pages 1–26, 1979.

B. Efron. *The jackknife, the bootstrap and other resampling plans*, volume 38. SIAM, 1982.

B. Efron and R. Tibshirani. *An introduction to the bootstrap*, volume 57. Chapman & Hall/CRC, 1993.

M. Egele, E. Kirda, and C. Kruegel. Mitigating drive-by download attacks: Challenges and open problems. In *Open Research Problems in Network Security – iNetSec 2009*, volume 309 of *IFIP Advances in Information and Communication Technology*, pages 52–62. Springer Boston, 2009a.

M. Egele, P. Wurzinger, C. Kruegel, and E. Kirda. Defending browsers against drive-by downloads: Mitigating heap-spraying code injection attacks. In *Detection of Intrusions and Malware, and Vulnerability Assessment*, volume 5587 of *Lecture Notes in Computer Science*, pages 88–106. Springer Berlin / Heidelberg, 2009b.

R. Fan, K. Chang, C. Hsieh, X. Wang, and C. Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, June 2008.

W. E. Feinberg. Teaching the type I and type II errors: The judicial process. *The American Statistician*, 25(3):30–32, 1971.

L. Finkelstein, E. Gabrilovich, Y. Matias, E. Rivlin, Z. Solan, G. Wolfman, and E. Ruppin. Placing search in context: The concept revisited. In *Proceedings of the 10th International Conference on World Wide Web (WWW '01)*, pages 406–414, Hong Kong, 2001. ACM.

R. A. Fisher. Statistical methods for research workers. 1934.

R. A. Fisher et al. *The design of experiments.* Edinburgh and London: Oliver & Boyd., 1935.

A. Fokkens, M. van Erp, M. Postma, T. Pedersen, P. Vossen, and N. Freire. Offspring from reproduction problems: What replication failure teaches us. 2013.

S. Forrest, S. A. Hofmeyr, A. Somayaji, and T. A. Longstaff. A sense of self for UNIX processes. In *Proceeding of the 17th IEEE Symposium on Security and Privacy (S&P '96)*, pages 120–128, May 1996.

S. Forrest, S. Hofmeyr, and A. Somayaji. The evolution of system-call monitoring. In *Proceedings of the 24th Annual Computer Security Applications Conference (ACSAC '08)*, pages 418–430, Anaheim, California, USA, December 2008.

E. A. Fox and J. A. Shaw. Combination of multiple searches. In *Proceedings of the 2nd Text REtrieval Conference (TREC '93)*, pages 243–243. National Institute of Standards & Technology, 1993.

A. Gal. *Efficient bytecode verification and compilation in a virtual machine*. PhD thesis, University of California, Irvine, CA, USA, 2006.

A. Gal, B. Eich, M. Shaver, D. Anderson, D. Mandelin, M. Haghighat, B. Kaplan, G. Hoare, B. Zbarsky, J. Orendorff, J. Ruderman, E. W. Smith, R. Reitmaier, M. Bebenita, M. Chang, and M. Franz. Trace-based just-in-time type specialization for dynamic languages. In *Proceedings of the 2009 ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI '09)*, pages 465–478, Dublin, Ireland, 2009. ACM.

J. D. Gibbons. *Nonparametric methods for quantitative analysis*. American series in mathematical and management sciences. American Sciences Press, 1985.

A. R. Gilpin. Table for conversion of kendall's tau to spearman's rho within the context of measures of magnitude of effect for meta-analysis. *Educational and psychological measurement*, 53(1): 87–92, 1993.

O. Granichin, V. Volkovich, and D. Toledano-Kitai. *Randomized algorithms in automatic control and data mining*, volume 67. Springer, 2014.

I. Guyon. *Feature extraction: foundations and applications*, volume 207. Springer, 2006.

I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *The Journal of Machine Learning Research*, 3:1157–1182, 2003.

L. Hamel. *Knowledge discovery with support vector machines*. Wiley Series on Methods and Applications in Data Mining. John Wiley & Sons, 2009.

D. J. Hand, H. Mannila, and P. Smyth. *Principles of data mining*. A Bradford book. MIT Press, 2001.

T. Hastie, R. Tibshirani, and J. H. Friedman. *The elements of statistical learning: Data mining, inference, and prediction*. Springer Series in Statistics. Springer, 2009.

D. M. Hawkins. The problem of overfitting. *Journal of Chemical Information and Modeling*, 44(1): 1–12, 2004.

W. Hersh, C. Buckley, T. J. Leone, and D. Hickam. OHSUMED: An interactive retrieval evaluation and new large test collection for research. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '94)*, pages 192–201, 1994.

B. Hoffman and B. Sullivan. *Ajax security*. Safari Books Online. Addison-Wesley, 2008.

W. Hu, Y. Liao, and V. R. Vemuri. Robust anomaly detection using support vector machines. In *In Proceedings of the 20th International Conference on Machine Learning (ICML '03)*, 2003.

D. Hull. Using statistical testing in the evaluation of retrieval experiments. In *Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '93)*, pages 329–338. ACM, 1993.

D. C. Ince, L. Hatton, and J. Graham-Cumming. The case for open computer programs. *Nature*, 482 (7386):485–488, 2012.

J. P. Ioannidis. Why most published research findings are false. *PLoS Medicine*, 2(8), 2005.

N. Japkowicz and M. Shah. *Evaluating learning algorithms: a classification perspective*. Cambridge University Press, 2011.

K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems (TOIS)*, 20(4):422–446, 2002.

E. C. Jensen. *Repeatable evaluation of information retrieval effectiveness in dynamic environments*. PhD thesis, Illinois Institute of Technology, 2006.

T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In *Machine Learning: ECML-98*, volume 1398 of *Lecture Notes in Computer Science*, pages 137–142. Springer Berlin / Heidelberg, 1998.

T. Joachims. Training linear SVMs in linear time. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 217–226. ACM, 2006.

P. J. John, F. Yu, Y. Xie, A. Krishnamurthy, and M. Abadi. Heat-seeking honeypots: Design and experience. In *Proceeding of the 20th International Conference on World Wide Web (WWW '11)*, pages 207–216, Hyderabad, India, March 2011.

M. Johns. On JavaScript malware and related threats. *Journal in Computer Virology*, 4:161–178, 2008.

D. Jurafsky and J. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall Series in Artificial Intelligence. Pearson Prentice Hall, 2009.

D. Kang, D. Fuller, and V. Honavar. Learning classifiers for misuse and anomaly detection using a bag of system calls representation. In *Proceeding of the 6th Annual IEEE SMC Information Assurance Workshop (IAW '05)*, pages 118–125, June 2005.

S. Kaplan, B. Livshits, B. Zorn, C. Seifert, and C. Curtsinger. "NOFUS: Automatically Detecting" + String.fromCharCode(32) + "ObFuSCateD ".toLowerCase() + "JavaScript Code". Technical report, Microsoft, 2011.

S. S. Keerthi, S. Sundararajan, K. Chang, C. Hsieh, and C. Lin. A sequential dual method for large scale multi-class linear SVMs. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '08)*, pages 408–416, Las Vegas, Nevada, USA, 2008. ACM.

O. Kempthorne. *The design and analysis of experiments.* Wiley, 1952.

M. G. Kendall. *Rank correlation methods.* Griffin, 1948.

D. F. Kibler and P. Langley. Machine learning as an experimental science. In *EWSL*, pages 81–92, 1988.

R. Krovetz. Viewing morphology as an inference process. In *Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '93)*, pages 191–202, Pittsburgh, Pennsylvania, USA, 1993. ACM.

A. Kulkarni and J. Callan. Document allocation policies for selective searching of distributed indexes. In *Proceedings of the 19th ACM Annual International Conference on Information and Knowledge Management (CIKM '10)*, pages 449–458, Toronto, ON, Canada, 2010. ACM, ACM.

S. Lawrence. Context in web search. *IEEE Bulletin of the Technical Committee on Data Engineering.*, 23(3):25–32, 2000.

W. Lee and S. J. Stolfo. Data mining approaches for intrusion detection. In *Proceedings of the 7th Conference on USENIX Security Symposium (USENIX '98)*, volume 7, pages 6–20, San Antonio, Texas, USA, January 1998.

Z. Li, Y. Tang, Y. Cao, V. Rastogi, Y. Chen, B. Liu, and C. Sbisa. WebShield: Enabling various web defense techniques without client side modifications. In *Proceeding of the 18th Annual Network & Distributed System Security Symposium (NDSS '11)*, San Diego, California, USA, February 2011.

F. Liu, C. Yu, and W. Meng. Personalized web search for improving retrieval effectiveness. *IEEE Transactions on Knowledge and Data Engineering*, 16(1):28–40, January 2004.

H. Liu and H. Motoda. *Feature selection for knowledge discovery and data mining*. Kluwer International Series in Engineering and Computer Science. Kluwer Academic Publishers, 1998.

H. Liu and H. Motoda. *Computational methods of feature selection*. Data mining and knowledge discovery. Chapman & Hall/CRC, 2008.

S. Lohr. *Sampling: design and analysis*. Cengage Learning, 2009.

L. Lu, V. Yegneswaran, P. Porras, and W. Lee. BLADE: An attack-agnostic approach for preventing drive-by malware infections. In *Proceedings of the 17th ACM Conference on Computer and Communications Security (CCS '10)*, pages 440–450, Chicago, Illinois, USA, October 2010.

J. Ma, L. K. Saul, S. Savage, and G. M. Voelker. Beyond blacklists: Learning to detect malicious web sites from suspicious URLs. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '09)*, pages 1245–1254, Paris, France, 2009.

C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to information retrieval*, volume 1. Cambridge University Press Cambridge, 2008.

S. Marsland. *Machine Learning: An algorithmic perspective*. Chapman & Hall/CRC Machine Learning & Pattern Recognition Series. CRC Press, 2009.

D. Metzler and W. B. Croft. A Markov random field model for term dependencies. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '05)*, pages 472–479, Salvador, Brazil, 2005. ACM.

A. Moffat and J. Zobel. Rank-biased precision for measurement of retrieval effectiveness. *ACM Transactions on Information Systems (TOIS)*, 27(1):2, 2008.

A. Moffat, W. Webber, and J. Zobel. Strategic system comparisons via targeted relevance judgments. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '07)*, pages 375–382. ACM, 2007.

R. Moonesinghe, M. J. Khoury, and A. C. J. W. Janssens. Most published research findings are false – but a little replication goes a long way. *PLoS Medicine*, 4(2), 2007.

A. Moshchuk, T. Bragin, S. D. Gribble, and L. H. M. A crawler-based study of spyware on the web. In *Proceeding of the 13th Annual Symposium on Network and Distributed System Security (NDSS'06)*, San Diego, California, USA, February 2006.

A. Moshchuk, T. Bragin, D. Deville, S. D. Gribble, and H. M. Levy. SpyProxy: Execution-based detection of malicious web content. In *Proceeding of the 16th USENIX Security Symposium (USENIX '07)*, Boston, MA, USA, August 2007.

D. Mutz, F. Valeur, G. Vigna, and C. Kruegel. Anomalous system call detection. *ACM Transactions on Information and System Security (TISSEC)*, 9:61–93, February 2006.

J. Nazario. Phoneyc: A virtual client honeypot. In *Proceedings of the 2nd USENIX conference on Large-scale exploits and emergent threats: botnets, spyware, worms, and more (LEET' 09)*, pages 6–13, Boston, MA, USA, 2009.

R. Perdisci, G. Gu, and W. Lee. Using an ensemble of one-class SVM classifiers to harden payload-based anomaly detection systems. In *Proceeding of the 6th International Conference on Data Mining (ICDM '06)*, pages 488–498, Hong Kong, December 2006.

N. Provos, D. McNamee, P. Mavrommatis, K. Wang, and N. Modadugu. The ghost in the browser analysis of web-based malware. In *Proceedings of the 1st Workshop on Hot Topics in Understanding Botnets (HotBots '07)*, Cambridge, MA, April 2007.

N. Provos, P. Mavrommatis, M. A. Rajab, and F. Monrose. All your iFRAMEs point to us. In *Proceeding of the 17th conference on USENIX Security Symposium (USENIX '08)*, pages 1–15, San Jose, CA, USA, July 2008.

J. R. Quinlan. *C4.5: Programs for machine learning*. Morgan Kaufmann Series in Machine Learning. Morgan Kaufmann Publishers, 1993.

P. Ratanaworabhan, B. Livshits, and B. Zorn. NOZZLE: A defense against heap-spraying code injection attacks. In *Proceedings of the 18th conference on USENIX Security Symposium (USENIX '09)*, pages 169–186, Montreal, Canada, April 2009.

G. Richards, C. Hammer, B. Burg, and J. Vitek. The eval that men do – a large-scale study of the use of eval in JavaScript applications. In M. Mezini, editor, *ECOOP 2011 – Object-Oriented Programming*, volume 6813 of *Lecture Notes in Computer Science*, pages 52–78. Springer Berlin / Heidelberg, 2011.

K. Rieck, T. Krueger, and A. Dewald. CUJO: Efficient detection and prevention of drive-by-download attacks. In *Proceedings of the 26th Annual Computer Security Applications Conference (ACSAC '10)*, pages 31–39, Austin, Texas, USA, December 2010.

S. E. Robertson and E. Kanoulas. On per-topic variance in IR evaluation. In *Proceedings of the 35th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '12)*, pages 891–900, Portland, Oregon, USA, 2012. ACM.

T. Sakai. New performance metrics based on multigrade relevance: Their application to question answering. In *Proceedings of the NTCIR-4 workshop*, 2004a.

T. Sakai. New performance metrics based on multigrade relevance: Their application to question answering. In *Proceedings of the NTCIR-4 workshop*, 2004b.

T. Sakai. Evaluating evaluation metrics based on the bootstrap. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '06)*, pages 525–532. ACM, 2006.

T. Sakai. Alternatives to Bpref. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '07)*, pages 71–78. ACM, 2007.

T. Sakai. Comparing metrics across TREC and NTCIR: The robustness to system bias. In *Proceedings of the 17th Annual International Conference on Information and Knowledge Management (CIKM '08)*, pages 581–590. ACM, 2008a.

T. Sakai. Comparing metrics across TREC and NTCIR: The robustness to pool depth bias. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '08)*, pages 691–692, Singapore, 2008b. ACM.

T. Sakai. On the robustness of information retrieval metrics to biased relevance assessments. *Journal of Information Processing*, 17:156–166, 2009.

T. Sakai. The unreusability of diversified search test collections. *Proceedings of the 5th International Workshop on Evaluating Information Access (EVIA '13)*, 2013.

T. Sakai and N. Kando. A further note on alternatives to Bpref, 2007.

T. Sakai and N. Kando. On information retrieval metrics designed for evaluation with incomplete relevance assessments. *Information Retrieval*, 11(5):447–470, 2008.

T. Sakai and T. Mitamura. Boiling down information retrieval test collections. In *Proceedings of the 2010 Adaptivity, Personalization and Fusion of Heterogeneous Information (RIAO '10)*, pages 49–56, Paris, France, April 2010.

T. Sakai, Z. Dou, R. Song, and N. Kando. The reusability of a diversified search test collection. In *Information Retrieval Technology*, pages 26–38. Springer, 2012.

M. Sanderson. Test collection based evaluation of information retrieval systems. *Foundations and Trends in Information Retrieval*, 4(4):247–375, 2010.

M. Sanderson and J. Zobel. Information retrieval system evaluation: Effort, sensitivity, and reliability. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '05)*, pages 162–169, Salvador, Brazil, 2005. ACM.

J. Savoy. Statistical inference in retrieval effectiveness evaluation. *Information Processing & Management*, 33(4):495–512, 1997.

B. Schölkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Adaptive Computation and Machine Learning. Mit Press, 2002.

C. Seifert, I. Welch, and P. Komisarczuk. Honeyc: The low-interaction client honeypot. In *Proceedings of the 5th NZ Computer Science Research Student Conference (NZCSRSC '07)*, Hamilton, New Zealand, April 2007.

C. Seifert, P. Komisarczuk, and I. Welch. True positive cost curve: A cost-based evaluation method for high-interaction client honeypots. In *Proceedings of the 3rd International Conference on Emerging Security Information, Systems and Technologies (SECURWARE '09)*, pages 63–69, Athens, Greece, June 2009.

H. Shacham, M. Page, B. Pfaff, E. Goh, N. Modadugu, and D. Boneh. On the effectiveness of address-space randomization. In *Proceedings of the 11th ACM conference on Computer and communications security (CCS '04)*, pages 298–307, Washington DC, USA, 2004.

M. Shokouhi. Central-rank-based collection selection in uncooperative distributed information retrieval. In *Advances in Information Retrieval*, pages 160–172. Springer, 2007.

L. Si and J. Callan. Relevant document distribution estimation method for resource selection. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '03)*, pages 298–305. ACM, 2003.

L. Si and J. Callan. Unified utility maximization framework for resource selection. In *Proceedings of the 13th ACM Annual International Conference on Information and Knowledge Management (CIKM '04)*, pages 32–41, Washington, D.C., USA, 2004. ACM.

L. Si and J. Callan. Modeling search engine effectiveness for federated search. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '05)*, pages 83–90, Salvador, Brazil, 2005. ACM.

M. D. Smucker, J. Allan, and B. Carterette. A comparison of statistical significance tests for information retrieval evaluation. In *Proceedings of the 16th Annual International Conference on Information and Knowledge Management (CIKM '07)*, pages 623–632, Lisbon, Portugal, 2007. ACM.

M. D. Smucker, J. Allan, and B. Carterette. Agreement among statistical significance tests for information retrieval evaluation at varying sample sizes. In *Proceedings of the 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '09)*, pages 630–631, Boston, MA, USA, 2009. ACM.

I. Soboroff and S. Robertson. Building a filtering test collection for TREC 2002. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '03)*, pages 243–250. ACM, 2003.

R. Sol, C. Guillon, F. Quintão Pereira, and M. Bigonha. Dynamic elimination of overflow tests in a trace compiler. In *Compiler Construction*, pages 2–21. Springer, 2011.

Sophos. Why hackers have turned to malicious JavaScript attacks. Technical report, Sophos Ltd., December 2010.

K. Spärck Jones. *Information retrieval experiment*. Butterworth-Heinemann, 1981.

K. Spärck Jones and R. G. Bates. Report on the design study for the 'ideal' information retrieval test collection. *British Library Research and Development Report*, 5428, 1977.

K. Spärck Jones and C. J. Van Rijsbergen. Report on the need for and provision of an "ideal" information retrieval test collection. Technical report, British Library Research and Development Report, 1975.

J. W. Stokes, R. Andersen, C. Seifert, and K. Chellapilla. Webcop: Locating neighborhoods of malware on the web. In *Proceedings of the 3rd USENIX Conference on Large-scale Exploits and Emergent Threats: Botnets, Spyware, Worms, and More (LEET '10)*, pages 5–13, San Jose, California, USA, 2010.

W. W. Stroup. *Generalized Linear Mixed Models: Modern Concepts, Methods and Applications*. Chapman & Hall/CRC Texts in Statistical Science. Taylor & Francis, 2012.

A. Stuart. *Basic ideas of scientific sampling*. Charles Griffin and Company, 1976.

Student. The probable error of a mean. *Biometrika*, pages 1–25, 1908.

C. Tannert, H. Elvers, and B. Jandrig. The ethics of uncertainty. *EMBO reports*, 8(10):892–896, October 2007.

P. Thomas and M. Shokouhi. SUSHI: Scoring scaled samples for server selection. In *Proceedings of the 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '09)*, pages 419–426, Boston, MA, USA, 2009. ACM.

R. G. Thorne. The efficiency of subject catalogues and the cost of information searches. *Journal of documentation*, 11(3):130–148, 1955.

C. J. van Rijsbergen. Information Retrieval, 1997.

E. Voorhees and D. K. Harman. *TREC: Experiment and evaluation in information retrieval*, volume 63. MIT Press Cambridge, 2005.

E. M. Voorhees. Variations in relevance judgments and the measurement of retrieval effectiveness. *Information Processing & Management*, 36(5):697–716, 2000.

E. M. Voorhees. The philosophy of information retrieval evaluation. In *Evaluation of Cross-Language Information Retrieval Systems*, pages 355–370. Springer, 2002.

E. M. Voorhees and C. Buckley. The effect of topic set size on retrieval experiment error. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '02)*, pages 316–323, Tampere, Finland, 2002. ACM.

E. M. Voorhees and D. Harman. Overview of the 8th text retrieval conference (TREC-8). In *TREC*, 2000.

E. M. Voorhees and D. K. Harman. Overview of the 7th text retrieval conference (TREC-7). In *Proceedings of the 7th Text REtrieval Conference (TREC-7)*. NIST, 1998.

Y. Wang, D. Beck, X. Jiang, R. Roussev, C. Verbowski, S. Chen, and S. King. Automated web patrol with strider honeymonkeys: Finding web sites that exploit browser vulnerabilities. In *Proceeding of the 13th Annual Symposium on Network and Distributed System Security (NDSS'06)*, San Diego, California, USA, February 2006.

C. Warrender, S. Forrest, and B. Pearlmutter. Detecting intrusions using system calls: Alternative data models. In *Proceedings of the 20th IEEE Symposium on Security and Privacy*, pages 133–145, Oakland, California, USA, May 1999.

W. Webber. *Measurement in information retrieval evaluation*. PhD thesis, 2011.

W. Webber, A. Moffat, and J. Zobel. Statistical power in retrieval experimentation. In *Proceedings of the 17th Annual International Conference on Information and Knowledge Management (CIKM '08)*, pages 571–580. ACM, 2008.

W. J. Wilbur. Non-parametric significance tests of retrieval performance comparisons. *Journal of Information Science*, 20(4):270–284, 1994.

F. Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83, 1945.

I. H. Witten and E. Frank. *Data mining: Practical machine learning tools and techniques*. Morgan Kaufmann series in data management systems. Morgan Kaufman, 2005.

J. Xu and W. B. Croft. Cluster-based language models for distributed retrieval. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '99)*, pages 254–261, Berkeley, California, USA, 1999. ACM.

E. Yilmaz and J. A. Aslam. Estimating average precision with incomplete and imperfect judgments. In *Proceedings of the 15th ACM Annual International Conference on Information and Knowledge Management (CIKM '06)*, pages 102–111. ACM, 2006.

E. Yilmaz, J. A. Aslam, and S. Robertson. A new rank correlation coefficient for information retrieval. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '08)*, pages 587–594. ACM, 2008a.

E. Yilmaz, E. Kanoulas, and J. A. Aslam. A simple and efficient sampling method for estimating AP and NDCG. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '08)*, pages 603–610. ACM, 2008b.

J. Zhang, C. Seifert, J. W. Stokes, and W. Lee. ARROW: Generating signatures to detect drive-by downloads. In *Proceeding of the 20th International Conference on World Wide Web (WWW '11)*, pages 187–196, Hyderabad, India, March 2011.

J. Zobel. How reliable are the results of large-scale information retrieval experiments? In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '98)*, pages 307–314. ACM, 1998.