Feature Selection for Prediction of User-Perceived Streaming Media Quality

Amy Csizmar Dalal and Jamie Olson
Department of Computer Science, Carleton College
email: {adalal, olsonja}@carleton.edu

Keywords: quality of service (QoS), multimedia applications, multimedia measurement, subjective quality, streaming media

Abstract

This paper considers the selection of features, measurements collected from an instrumented media player application, that most accurately predict the user-perceived quality of a media stream. The features are utilized by a nearest-neighbor stream quality prediction algorithm using a distance metric of dynamic time warping. We explore three ways of selecting features from this data: manually, by observing how application-layer measurements change with changing network congestion conditions; correlation-based; and a mathematically-based technique using principal component analysis (PCA). We compare the prediction algorithm's accuracy obtained using the features selected by each method, using a performance evaluation metric we term *hit rate*. Our results show that each method selects one feature set that, when used by our predictor, yields very high hit rates (typically 70-90%), and that each of these feature sets includes one particular feature in common: retransmitted packets. We also show that the correlation-based and PCA-based methods of selecting features do not consistently select acceptable feature sets for our stream quality predictor, in terms of the hit rates generated by the predictor.

1. INTRODUCTION

Determining the subjective, user-perceived quality of a media stream in a scalable and quantifiable way is a difficult problem. As with all Internet-based applications, there is a complex interplay between network congestion conditions and the effect these congestion conditions have on application performance. Knowing how end users perceive the quality of audio and video streamed on-demand over computer networks, and the relationship between stream quality and network congestion, can lead to better design of streaming protocols, computer networks, and content delivery systems. The standard measure of user-perceived audio and video quality, the five-point scale Mean Opinion Score [15], or MOS, is inherently unscalable, and produces only a partial picture of stream quality, without context.

To best discern the user-perceived quality of a media stream, we want to combine the ease and convenience of objective metrics with the information offered by a subjective rating such as the MOS. A number of studies have explored this option. Some, like [20] and [1], correlate measurements on both the sender and receiver sides. Others, like [5] and [3], use the Emodel [10], an objective mechanism for assessing audio quality using transmission parameters. An alternate approach is to utilize application-layer objective metrics, taken at the client's machine through an instrumented media player application [3, 13, 14, 19]. These approaches allow one to take measurements as close to the user as possible, in some cases without requiring the user's participation, providing a more accurate assessment of the state of the application at any given time.

In previous work [6, 8], we demonstrate that objective data collected from an instrumented media player application can be used to *predict* subjective quality ratings with a high degree of accuracy (typically 70-90%). While our success rates are quite high, we manually selected the objective measurements, or *features*, used by our subjective quality prediction algorithm, by observing how application-layer measurements change with changing network congestion conditions. We are interested in determining whether an even better set of features can be identified, using a systematic algorithm, that yields higher hit rates than those obtained using the manually selected features.

In this work, we explore two alternate mechanisms for selecting features to be used by our subjective quality prediction algorithm. The first method selects the features that show the highest correlations with subjective quality ratings [8]. The second method is based on a set of algorithms that mathematically analyze the data to determine which attributes influence the user quality ratings most strongly. The latter algorithms, collectively named CLeVer [21], utilize *principal component analysis*, or PCA. We demonstrate that, while the correlation method and PCA method each produce one candidate feature set that generates comparable results in predicting user quality ratings, they do not consistently select acceptable feature sets, in terms of increased accuracy by the stream quality predictor. We evaluate each mechanism using a metric called *hit rate*, which measures how accurately the prediction algorithm selects the correct user quality rating, within a specified tolerance.

Section 2. discusses the relevant background on using application-layer metrics to predict subjective stream quality and the stream quality prediction algorithm used in this study. Section 3. presents the three mechanisms for selecting

 ISBN # 1-56555-317-9

features from the available set of objective application-layer measurements. How we obtained the objective and subjective data for this study, and how we apply the features selection methods to this data, is discussed in Section 4. Section 5. presents the predictor accuracies using the selected features from all three methods, with a summary and areas of future study in Section 6.

2. BACKGROUND

2.1. The Utility of Application-Layer Metrics

The best and most accurate way to assess a user's perception of stream quality is to ask the user directly. This solution, however, is neither practical nor scalable. A good alternative, then, is to take measurements as close to the user as possible. In this context, "close" can mean geographically close or conceptually close. Both goals can be achieved by taking measurements at the user's desktop, from the application itself.

We have developed an instrumented version of Windows Media Player [7] that collects application-layer data about the state of a media stream at predefined intervals (currently, one second) using ActiveX hooks. The state information we collect includes information on the number of lost, retransmitted, and correctly-received packets[1]; the instantaneous bandwidth, in kilobits per second, and frame rate, in frames per second, as measured by the player; and the number of times the player has entered *buffer starvation* mode, in which it does not play out the stream but instead waits for the playout buffer to fill up so that playback can resume. [2]

To better understand how these measurements reflect the amount of congestion on the underlying computer network, Figure 1 shows an example of the data collected by our tool for a stream several minutes in duration that experiences a moderate level of network congestion. The plot shows that most of the reaction to network-level congestion is reflected in the packet-level data. For instance, the bandwidth and frame rate do not change significantly over the duration of the stream; in fact, the frame rate actually *increases* slightly over the lifetime of the stream. The number of retransmitted packets increases well before the number of lost packets. This demonstrates that the media player's first response to network congestion is to request that the media server resend missing packets, and that packets are only declared lost once the player has unsuccessfully received these packets in time to be played out.

We can divide the measurements collected by the instrumented media player into one of three categories: *lagging*

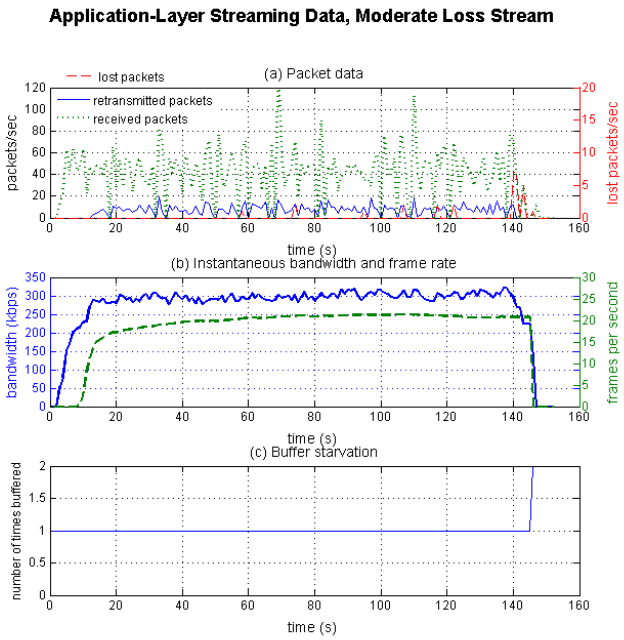

Figure 1. Time-series data collected by the instrumented media player application. (a) Packet-level data: retransmitted packets and received packets are on the left y-axis, lost packets on the right y-axis. (b) Bandwidth and frame rate, on the left y-axis and right y-axis, respectively. (c) Buffer count, including the initial startup buffering period

indicators, *leading indicators*, and *instantaneous indicators* [7, 8]. **Leading indicators** are the first to change in response to network congestion, before stream quality degrades; they are the precursors of degraded stream quality. In the discussion above, retransmitted packets and received packets are leading indicators. **Lagging indicators** are reactive metrics; they change in response to an episode of network congestion, appearing well after the onset of degraded stream quality. Examples of lagging indicators from the discussion above are frame rate and bandwidth. **Instantaneous indicators** change at the exact moment, or as close as possible, at which stream quality degrades. Examples of instantaneous indicators from the discussion above are lost packets and buffer count.

2.2. Using Application-Layer Measurements to Predict User Quality Ratings

Streams that have been exposed to similar levels of network congestion will most likely show similar patterns of retransmitted packets, lost packets, etc., even if the exact occurrences and durations do not match exactly. Streams that exhibit these similar characteristics will also exhibit similar user quality ratings, particularly once individual user biases have been accounted for.

[1] "Correctly-received packets", referred to as "received packets" throughout this paper, are the data packets that arrive at the media player application within their initial delivery window.

[2] We refer to this measurement as *buffer count*. Buffer count also includes the initial startup buffering period of the player.

One way to discern the similarity between two streams is to use the application-layer measurements collected at each second from one stream to those from another stream to calculate the "distance" between the two streams. The "closer" the two streams, the more similar they are. The challenge in this approach, besides identifying an appropriate distance metric, lies in the complexity of the distance calculation. If we are measuring six pieces of state information from a stream, for instance, then our distance calculation must operate in six dimensions. If timing information is important, as it is in the data under consideration here, then we must repeat this calculation for every measurement at every second.

Ideally, there will be one or more measurements that most strongly influence the distance calculation. Identifying these measurements reduces the complexity of the distance calculation by reducing the number of dimensions in which the distance calculation occurs. From a practical standpoint, this reduces the number of computational resources we require to determine the similarity between two streams, where resources might include CPU, memory, storage, and of course calculation time.

Our goal, then, is to explore systematic and accurate ways to reduce the number of measurements (here, the six pieces of state information collected by our instrumented media player) that our distance calculation requires. We will then utilize this information to predict user quality ratings, as described in the next section.

2.3. User Quality Rating Prediction Algorithm

Our stream quality prediction algorithm is described in detail in [6]; here, we briefly summarize its operation. Our particular problem calls for using knowledge of pre-labeled data to predict labels on new data [9,11,17]. The goal is to produce a predictor by "training," i.e. running a data mining algorithm, on a set of labeled data. The predictor can then be tested on unlabeled data. Our data consists of the set of measurements collected from the instrumented media player on a given set of streams. The labels in this case are the quality ratings assigned by users who watched these streams as measurements were being collected (see Section 4.1. for details on how this data was obtained).

Our predictor uses a *nearest neighbor* algorithm. The idea behind a nearest neighbor predictor is as follows: For an unlabeled example, locate all of the examples in the training set which are "closest" to the unlabeled example, subject to some distance metric. To assign a label to the unlabeled example, a single label is produced from the set of labels for the closest points. In this case, given an unrated stream, determine this stream's nearest neighbor(s) using the measurement data collected from the instrumented media player each second. Once a nearest neighbor stream has been identified, assign that stream's rating to the unrated stream.

The distance metric used by our predictor is an extension to *dynamic time warping*, a generalization of Euclidean distance designed for use with time series data, that facilitates its use on multidimensional time series [18]. Briefly, dynamic time warping is based on the assumption that two time series may be quite similar, even if the precise timing between the two series is misaligned. While dynamic time warping aligns the start and end points of each time series (stream), it allows points in mid-stream to align with the closest appropriate point. This fluidity often results in more accurate predictions and pattern identifications. A stream of unknown quality that exhibits packet loss on a periodic basis, for example, is expected to have similar quality to another stream that also loses packets periodically. However, it should not be a requirement for similarity between such streams that the packet losses occur *at precisely identical times*.

To reduce the computational time and (quadratic) complexity inherent in dynamic time warping, we apply two optimizations: the popular Sakoe-Chiba band [12,16], which limits the distance that one time series can shift relative to the other; and Keogh minimum bounds [12], to quickly determine candidates for the set of nearest neighbors. Once the set of candidate streams has been selected, the predictor produces a single rating by calculating the mean of the ratings given to the candidate streams.

3. FEATURE SELECTION METHODS

In this section, we discuss three mechanisms for feature selection. Each feature defines one aspect of the state of a stream at a certain time snapshot. All three mechanisms exploit some aspect of our data in selecting the features that best characterize the data. We first discuss the manual feature selection method from [6]. We then consider two additional mechanisms: one in which we utilize the correlations between the application-layer measurements and subjective user quality ratings; and one in which we use a more mathematically-based method based on principal component analysis.

3.1. Manual Feature Selection

In Section 2., we described the application-layer measurements that our instrumented media player application collects about each stream. The leading indicators—retransmitted packets and received packets—determine the probability that stream quality will degrade in the near future, and determine the extent to which stream quality degrades. The instantaneous indicators—lost packets and buffer count—determine exactly which events lead to lower or higher user-perceived stream quality, and indicate the exact moment at which network congestion events are manifested in stream quality degradation. The lagging indicators—bandwidth and frame rate—verify that stream quality degraded at a certain point

in the stream. When selecting features for our stream quality *predictor*, we are most interested in measurements that either indicate that degraded quality is occurring at this very moment, or that degraded quality is likely if current trends continue: the *leading indicators* or *instantaneous indicators* of stream quality.

Since stream quality is impacted by the amount of congestion on the underlying network, it follows that leading and instantaneous indicators that directly reflect network congestion will be good candidate features for our predictor. As Figure 1 shows, lost packets and retransmitted packets most accurately reflect network congestion. These are accurate second-order and first-order approximations of network congestion, respectively. We thus select these two features as input to our stream quality predictor.

3.2. Correlation-Based Feature Selection

Another way to determine the strength of the relationship between application-layer measurements and user quality ratings is to calculate the correlation coefficients between them. A strong correlation, either positive or negative, between a particular measurement and the quality ratings indicates that the measurement has a strong influence on the rating [8]. This strategy removes all timing information from the stream data; we are in essence left with summary statistics from the media stream, which makes the correlation coefficients easy and fast to compute.

To identify good candidate features, we calculate the correlation coefficients between the user quality ratings and the measurements obtained from the instrumented media player application. We then take the measurements that have the highest correlation coefficients and use these features as input to our stream quality predictor.

3.3. PCA-Based Feature Selection

The state information collected from the instrumented media player application forms a *multivariate time series*. We can consider the state information over the length of a single stream to be a series of points in n-space, where n is the number of pieces of state information we collect each second. We can preprocess this time series to remove irrelevant and/or redundant features in such a way as to retain the relevant information from the original data. Mathematically, we can reduce the dimension of the n-space to a lower dimension by projecting the original measurements onto a smaller basis, and then selecting the measurements (features) that contribute most strongly to these new basis vectors.

We use a set of algorithms that performs basis reduction and that is well-suited to multivariate time series data, named CLeVer [21]. The principle behind each of these algorithms involves a mathematical procedure called *principal component analysis*, or PCA. The purpose of PCA is to reduce the number of dimensions in each "item" in a set of data (in our case, an item is single stream watched and rated by a single user), by finding a lower-dimension basis that retains the most relevant information from the original data. PCA is performed separately for each item in the set of data, after which the CLeVer algorithms find a set of basis vectors that are common to *all* items in the data set. These basis vectors are called the *descriptive common principal components*. At this stage, we project all of the original data points for all of the items into this common subspace, and then prune the features so that only those that contribute most heavily to the new basis are left.

CLeVer is actually a family of algorithms that use three different heuristics to reduce the number of features. *CLeVer-Rank* sorts the features by decreasing magnitude and selects the top features from this ranking. *CLeVer-Cluster* and *CLeVer-Hybrid* both perform k-Means clustering [9] on the basis vectors. CLeVer-Cluster selects the features that are the closest to the cluster centers. CLeVer-Hybrid uses the same ranking mechanism as CLeVer-Rank to rank the features within each cluster, and then selects the highest ranking feature from each cluster.

4. METHODOLOGY

In this section, we discuss how we applied the feature selection mechanisms presented in Section 3. to use in our stream quality prediction algorithm. In addition, we discuss how we obtained the raw data input for our predictor, as well as define our performance evaluation metric, *hit rate*.

4.1. Experimental Data

The data collection testbed consists of a set of 14 client machines on a subnet of a small campus network, and a media server on a separate subnet, isolated from the rest of the campus network by two routers. The router closest to the media server runs NIST Net software [4], which is used to apply randomly-distributed packet losses over the duration of each stream at the percentages indicated in Table 1. The media server is a 2.4 GHz Pentium processor machine with 512 MB of RAM, running Windows Media Server 2003. The NIST Net router is a 700 MHz processor machine with 512 MB of RAM, running Linux kernel 2.4.21-27 and NIST Net version 2.0.12. The client machines have 3.4 GHz Pentium processors and 1 GB of RAM, and run Windows XP SP2 and Windows Media Player version 10. Each client machine hosts our instrumented version of Windows Media Player. Before and during the experiments, we took periodic measurements on the campus network, as well as on the router, to verify that the campus network was contributing negligible loss and delays to our testbed and to verify that the router was not a bottleneck within our network.

The media streams used in this study are listed in Table 1. The congestion levels were chosen experimentally, both to overcome some of the mechanisms that Windows Media Player uses to mitigate the effects of network congestion [14] and to achieve a certain and consistent quality level in each stream under each congestion level.

We conducted a set of three experiments carried out over a period of two days in May of 2006. In each experiment, we showed a group of college-age (and several slightly older) participants each of three media streams twice, once with no loss and once with either mild, moderate, or severe loss. The participants were not aware of the loss levels shown to them for a particular stream; they also were not told which version of the stream had no loss introduced. The participants rated the overall (audio plus video) quality of each stream on a seven-point scale, with one being the worst possible quality and seven being the best possible quality. The instrumented media player collected data from each stream simultaneously. From these experiments, we collected data from a total of 38 participants and their respective client machines, yielding a total of 228 data items. Each data item consists of the state information collected from a single stream at a given congestion level watched by a single participant, as well as the quality rating that the participant assigned to the stream.

The factors that affect user ratings can vary significantly. Users may be more or less sensitive to encoding differences between streams, or they may view a stream that experiences more apparent loss due to which frames are dropped by the router. Normalizing user ratings helps mitigate these factors. We normalized the quality ratings assigned by our participants using the formula

$$r_{\text{norm},s} = \frac{r_s - \bar{r}}{\sigma_r} \quad (1)$$

where r_s is the user's quality rating for stream s, $\bar{r}$ is the average of the user's quality ratings on all streams viewed, and σ_r is the standard deviation of the user's quality ratings on all streams viewed [2].

4.2. Applying the Feature Selection Algorithms

Our data set consists of $N = 228$ items. Each item contains T observations, where T is the number of seconds in the stream.[3] At each second, there are $M = 6$ features, corresponding to the state information collected by the instrumented media player. The feature selection algorithms each reduce M.

Using manual feature selection, we reduce M from six to two by extracting the number of lost packets and retransmitted packets received at each second in the stream for each item. This results in N T-by-2 items to be input into the predictor.

Using correlation-based feature selection, we reduce M from six to one as follows. First, we calculate the total number of lost, retransmitted, and received packets over the entire length of the stream, as well as the buffer count, for each item. Next, we calculate the average bandwidth and average frame rate over the duration of the stream, for each item. We thus remove all timing information from each item, so that $T = 1$ for each item and there are six summary statistics for that item. We group the items by stream, so that the state information from all of the Ad streams, for instance, is included in a single matrix. For each such stream matrix, we calculate the correlation coefficient between the normalized user quality rating assigned to that item and each summary statistic from that item, selecting the feature that yields the highest correlation coefficient for the stream. Finally, we extract that feature from each of the original N data items. This results in N 1-by-1 items to be input into the predictor.

Using the CLeVer algorithms, we reduce M to either two or three as follows. We determine the p basis vectors that contribute 80% of the information to the original basis vectors for each item. We then separate the items by stream, as in the correlation-based feature selection method and calculate the descriptive principle common components, or common basis vectors, separately for each stream, as described in Section 3.3. To these basis vectors, we apply each of the three heuristics (CLeVer-Rank, CLeVer-Cluster, and CLeVer-Hybrid) to determine the top two or three features for each stream. We extract these features from the original N data items, resulting in N T-by-2 or T-by-3 items to be input into the predictor.

4.3. Predictor Tuning Phase

To tune the stream quality predictor, we determine optimal values for its two parameters—K, the number of nearest neighbors to consider, and w, the width of the Sakoe-Chiba bands—experimentally via a leave-one-out cross-validation procedure on each training set. We divide our data set into three subsets, each one containing all labeled data for each stream, yielding three training sets. We then read in the state information from each stream, reduced as described in Section 4.2., and store this data. In a training set containing P items, we remove one stream and predict a rating for it using the remaining $P-1$ items as a proxy training set, varying K and w between 1 and 20 and 0 and 30, respectively.[4] For each value of K and w, we record whether or not the stream is rated correctly (see Section 4.4. for a description of our evaluation metric). We repeat this procedure for each item in the training set and select the K and w values that yield the best results, as determined by our evaluation metric, overall. Ties are broken

[3] Note that T varies based on the actual stream viewed and by the level of congestion: a stream that encounters moderate to severe congestion tends to be slightly longer than the same stream under mild or no congestion.

[4] $w = 0$ is equivalent to traditional Euclidean distance.

 ISBN # 1-56555-317-9

Table 1. Description of test streams and congestion patterns

Stream information				Network congestion level (loss)		
Name	**Time (mm:ss)**	**Description**	$\overline{BW}$ **(kbps)**	**Mild**	**Moderate**	**Severe**
Ad	0:30	Moderate action	273	5%	15%	25%
Trailer	2:22	High action	273	5%	15%	25%
News	4:09	Low/moderate action	331	5%	15%	25%

by selecting the K and w values with the smallest execution times, and then by selecting the smallest w value. These K and w values are used as predictor parameters for the test sets, where the test sets are the training sets with the stream quality ratings removed. Since this entire cross-validation process is done purely on the training data, it does not in any way use information from the items whose quality we wish to predict.

4.4. Performance Evaluation

Because user quality ratings are subjective by nature, and because there is significant variation in how particular users rate streams, we measure prediction accuracy using a metric we define as *hit rate*. Hit rate is the percentage of time a prediction falls within 0.8 standard deviations of the user's *normalized* rating (Equation 1) for that stream. This corresponds to approximately plus or minus one point on the unnormalized seven-point scale.

5. RESULTS

The major goal of this work is to determine if more systematic ways of selecting features for our stream quality prediction algorithm, either the correlation-based method or the PCA-based method, leads to a more accurate predictor of stream quality. In this section, we compare the hit rates achieved by our stream quality predictor using the manually-selected features (lost and retransmitted packets) against the hit rates achieved by our stream quality predictor using the features selected by the correlation-based and PCA-based feature selection methods. Because each feature selection method yields a different number of features (between one and three), we also examine the effect of the number of features used by the predictor on the predictor's accuracy.

5.1. Manually-Selected Features

Table 2 presents the hit rates obtained by using the two manually-selected features, lost packets and retransmitted packets, in our stream quality predictor. We will use these hit rates as reference values throughout the rest of this section. The cross-validated results, those along the diagonal, show how each predictor would perform when rating an unknown stream of similar length and characteristics, or an unlabeled stream identical to the streams in the training set. In this scenario, the predictor achieves very high hit rates of 88-90%. The predictor also achieves high hit rates (above 76%) when either Trailer or News is used as the test set. The predictor's hit rates are somewhat lower (mid-to-upper 60% range) when either of the longer streams (Trailer, News) is used to predict the rating for the shortest stream (Ad). This is an artifact of dynamic time warping: we in essence "compact" the longer stream to match the shorter stream. Packet losses and retransmissions are concentrated into smaller portions of a shorter stream, whereas in a longer stream the packet losses would be spread out over the duration of the stream.

Table 2. Hit rates for the stream quality predictor, using the manually-selected feature set consisting of lost packets and retransmitted packets

Training	**Test Stream**			**Parameters**	
Stream	**Ad**	**Trailer**	**News**	K	w
Ad	88.2	79.0	77.6	6	1
Trailer	64.5	89.5	81.6	12	3
News	67.1	76.3	88.2	14	∞

5.2. Correlation-Selected Features

Table 3 lists the correlation coefficients between the stream quality ratings assigned by the participants and the measurements collected from our instrumented media player. We define a correlation to be "strong" if its magnitude is greater than 0.5. While all of the features fit the definition of strong correlation for the Trailer and News streams, only three meet that criteria for the Ad stream. Also, there is no one feature that is the most highly correlated in all three streams, although three features (retransmitted packets, frame rate, and buffer count) are one of the most highly correlated features in two of the three streams.

Because the correlation data is inconclusive, we present the hit rates for each possible feature in Table 4. The hit rates that are higher than, or equal to, the hit rates in Table 2 are in boldface. The table shows that using retransmitted packets alone yields as good or a higher hit rate than using both retransmitted and lost packets in seven of the nine trials; the difference in the remaining two trials is only 1.6%. The combination of lost packet and retransmitted packet information actually *decreases* the efficacy of the predictor over just using retransmitted packet information. The correlation coefficients indicated that frame rate and buffer count would be good potential features for our predictor, but our results show this to not be the case. Buffer starvation events occur infrequently enough

Table 3. Correlation coefficients between the measurements obtained by the instrumented media player and the normalized user quality ratings. The top three correlation values for each stream are in **bold**. This table also contains the abbreviations used for the features in the rest of the tables in this paper.

	Stream		
Feature	**Ad**	**Trailer**	**News**
Lost packets (LP)	**-0.5762**	-0.7315	-0.7533
Received packets (TP)	0.1264	**0.8402**	0.7453
Retransmitted packets (RP)	-0.1137	**-0.8386**	**-0.7622**
Bandwidth (BW)	0.2107	**-0.7973**	-0.7440
Frame rate (FR)	**0.6612**	0.7592	**0.8145**
Buffer count (BC)	**-0.7323**	-0.6779	**-0.7772**

in our data (35% of the streams contain a single mid-stream buffer starvation event, and only 4% contain more than one such event) to make the amount of information provided by buffer count to not be sufficient for our predictor. Frame rate, as a lagging predictor of stream quality, most likely suffers here for that reason.

Table 4. Hit rates for the stream quality predictor, using the features chosen by correlation. Hit rates that meet or exceed those obtained by the predictor using the features {LP,RP} are in **bold.**

	Training	Test Stream			Params	
Feature	**Stream**	**Ad**	**Trailer**	**News**	K	w
LP	**Ad**	**88.2**	64.5	56.6	9	5
	Trailer	**85.5**	88.2	65.8	7	5
	News	**82.9**	**81.6**	86.8	5	10
TP	**Ad**	84.2	52.6	55.3	6	0
	Trailer	60.5	82.9	77.6	9	19
	News	**69.7**	56.6	85.5	6	1
RP	**Ad**	**88.2**	**80.3**	**80.3**	3	1
	Trailer	**72.4**	**89.5**	80.3	6	0
	News	**67.1**	**80.3**	86.8	9	17
BW	**Ad**	78.9	30.3	35.5	2	2
	Trailer	27.6	88.1	47.4	4	1
	News	51.3	46.1	77.6	3	17
FR	**Ad**	86.8	61.8	56.6	14	2
	Trailer	**73.7**	85.5	65.8	6	∞
	News	60.5	56.6	78.9	3	8
BC	**Ad**	84.2	60.5	57.9	6	0
	Trailer	60.5	88.2	57.9	5	1
	News	65.8	**76.3**	82.9	3	∞

5.3. PCA-Selected Features

5.3.1. CLeVer-Rank

The first row of Table 5 lists the top three features obtained by running CLeVer-Rank on the data from each stream, as described in Section 4.1. While the features selected by CLeVer-Rank for the three streams do not overlap perfectly, there are a number of features that appear repeatedly: received packets is selected for all three streams, while lost packets, retransmitted packets, and bandwidth are selected for two of the three streams.

We used the top two and top three features selected by CLeVer-Rank on each of the three streams as input to the stream quality predictor. The resulting hit rates obtained by the predictor using these features are presented in Table 6. None of the feature sets selected by CLeVer-Rank yields hit rates as high as the hit rates in Table 2. In particular, the hit rates when Ad is used as the training stream and either Trailer or News is used as the test stream show a significant decrease over the hit rates in Table 2. Thus, even though CLeVer-Rank mathematically represents the stream data, this does not translate into improved performance by the predictor.

One interesting result is that the hit rates for the predictor when using the feature set of received packets and retransmitted packets are identical to the results obtained for the predictor when using the feature set consisting of lost packets, received packets, and retransmitted packets. This indicates that lost packets provide no additional information to the predictor in this instance.

5.3.2. CLeVer-Cluster and CLeVer-Hybrid

The second and third rows in Table 5 list the clusters chosen by the CLeVer-Cluster and CLeVer-Hybrid algorithms on our data, along with the feature selected from each cluster by each of these algorithms (in boldface). The clustering portion of each algorithm, rather than evenly distributing the features among the clusters, creates unbalanced clusters of sizes four, one, and one. Part of our rationale for choosing a cluster size of three was to determine if the features would cluster by lagging, leading, and instantaneous indicators. This did not occur.[5] Thus, the clustering portion of the CLeVer algorithms apparently found the features to be more similar than not. CLeVer-Cluster and CLeVer-Hybrid also never select the same feature from the cluster of four. This demonstrates that the most highly-ranked feature is often not the one at the center of the chosen cluster.

The top half of Table 7 presents the hit rates obtained when the features selected by CLeVer-Cluster are used in the stream quality predictor. We see that there is no one set of features that generates higher hit rates than those in Table 2, although when the predictor uses the feature set consisting of lost pack-

[5] We saw similar results when we reduced the number of clusters to two.

 ISBN # 1-56555-317-9

Table 5. The features selected by each of the three CLeVer algorithms, and the clusters selected by CLeVer-Cluster and CLeVer-Hybrid. The features selected by CLeVer-Cluster and CLeVer-Hybrid are in **bold** within the feature clusters.

Algorithm	**Ad**	**Trailer**	**News**
Rank	TP, RP, BW	LP, TP, RP	LP, BW, TP
Cluster	{**TP**, RP, BW, BC} {**LP**} {**FR**}	{**LP**, TP, RP, BW} {**FR**} {**BC**}	{LP, TP, **BW**, BC} {**RP**} {**FR**}
Hybrid	{TP, RP, BW, **BC**} {**LP**} {**FR**}	{LP, TP, RP, **BW**} {**FR**} {**BC**}	{LP, TP, BW,**BC**} {**RP**} {**FR**}

Table 6. Hit rates for the stream quality predictor, using the features chosen by CLeVer-Rank. Hit rates that meet or exceed those obtained by the predictor using the features {LP,RP} are in **bold.**

	Training	**Test Stream**			**Params**	
Features	**Stream**	**Ad**	**Trailer**	**News**	*K*	*w*
LP, BW	**Ad**	82.9	32.9	30.3	3	1
	Trailer	27.6	**89.5**	47.4	4	1
	News	59.2	39.5	77.6	2	18
LP, TP	**Ad**	84.2	56.6	57.9	4	0
	Trailer	61.8	82.9	**81.6**	9	7
	News	**69.7**	56.6	85.5	6	1
TP, RP	**Ad**	84.2	56.6	54.0	7	0
	Trailer	60.5	86.8	**89.5**	9	13
	News	63.2	69.7	85.5	4	1
LP, TP, BW	**Ad**	82.9	59.2	43.4	9	2
	Trailer	27.6	86.8	63.2	16	0
	News	**67.1**	61.8	85.5	8	7
TP, RP, BW	**Ad**	84.2	61.8	61.8	8	3
	Trailer	**65.8**	88.2	**81.6**	16	16
	News	61.8	**77.6**	86.8	6	4
LP, TP, RP	**Ad**	84.2	56.6	54.0	7	0
	Trailer	60.5	86.8	**89.5**	9	13
	News	63.2	69.7	85.5	4	1

ets, frame rate, and buffer count (containing two instantaneous indicators and a lagging indicator), it generates higher hit rates when Ad is the test stream. When the predictor uses the feature set containing both bandwidth and frame rate, both lagging indicators, its hit rate plummets, particularly when the Ad stream is used as the predictor's training stream.

Table 7. Hit rates for the stream quality predictor, using the features chosen by CLeVer-Cluster (above the double line) and CLeVer-Hybrid (below the double line). Hit rates that meet or exceed those obtained by the predictor using the features {LP,RP} are in **bold.**

	Training	**Test Stream**			**Params**	
Features	**Stream**	**Ad**	**Trailer**	**News**	*K*	*w*
LP, TP, FR	**Ad**	86.8	59.2	57.9	4	0
	Trailer	60.5	82.9	80.3	9	7
	News	**71.1**	54.0	85.5	6	1
RP, BW, FR	**Ad**	81.6	30.3	34.2	2	1
	Trailer	31.6	**89.5**	60.5	14	0
	News	**67.1**	**79.0**	84.2	4	4
LP, FR, BC	**Ad**	**89.5**	52.6	55.3	4	3
	Trailer	**75.0**	85.5	54.0	6	2
	News	**79.0**	65.8	85.5	6	20
BW, FR, BC	**Ad**	79.0	29.0	34.2	2	1
	Trailer	27.6	**89.5**	47.4	4	1
	News	56.6	39.5	75.0	2	18
RP, FR, BC	**Ad**	**89.5**	72.4	**84.2**	4	4
	Trailer	**69.7**	88.2	79.0	8	0
	News	65.8	**81.6**	86.8	13	9

The bottom half of Table 7 presents the hit rates obtained when the features selected by CLeVer-Hybrid are used in the stream quality predictor. Note that one of the feature sets, {lost packets, frame rate, buffer count}, was also selected by CLeVer-Cluster. The feature set that includes one leading, one lagging, and one instantaneous indicator (retransmitted packets, frame rate, and buffer count, respectively), generates comparable hit rates to the ones in Table 2, and also comparable to the hit rates when the predictor uses only retransmitted packets as its feature.

5.4. Discussion

The manual feature selection method and PCA-based feature selection method both exploit the time-series nature of the data collected by our instrumented media player. The correlation-based method, by contrast, removes all timing information from the stream before selecting appropriate features for the stream quality predictor. Because of this, we originally expected that the predictor would generate higher hit rates when using either the manual or the PCA-based method over the correlation-based method. What we found instead is that neither the correlation-based method nor the PCA-based method consistently chose better features than the ones we manually selected, and in many cases selected worse feature sets. Each method successfully identifies one set of features—retransmitted packets, by the correlation method; retransmitted packets, frame rate, and buffer count, by the PCA method—that, when used by our stream quality predictor, generates comparable (or better) hit rates to our stream quality predictor using the manually-selected features (lost and retransmitted packets).

What all three feature sets share in common is the inclusion of retransmitted packets. In fact, the best hit rates overall came from the predictor using *only* retransmitted packets as a feature. Retransmitted packets is a leading indicator, and indeed the first indicator, of network congestion. Our results show that this particular feature is in fact quite accurate at predicting degraded stream quality on its own. Adding lost packets, or buffer count and frame rate, to the feature set still yields a highly effective predictor, but the additional information provided by these other features—instantaneous and lagging indicators—leads to slightly lower hit rates. This, in turn, has practical implications for the stream quality predictor: by reducing the set of features from six to one, the distance metric calculation becomes much simpler and much faster to compute, which should reduce the time and resources required for the predictor.

6. CONCLUSIONS AND FUTURE WORK

This paper presents three mechanisms for selecting features to be used by a stream quality prediction algorithm. The features are application-layer measurements taken from an instrumented media player application. The first mechanism selects features manually, based on information about how network congestion is manifested in application-layer measurements. The second mechanism selects features using the correlations between the application-layer measurements and subjective quality ratings assigned by users. The third mechanism uses principal component analysis to project the stream data into a lower-dimension basis, and select the features that contribute most strongly to this new basis. We evaluate the features selected by these methods by using them in a nearest-neighbor predictor to assign subjective quality ratings to media streams. Our performance evaluation metric is hit rate, or the percentage of time the predictor correctly selects a subjective rating within a small tolerance.

Our results show that neither the correlation-based method nor the PCA-based method of feature selection consistently selects features that produce higher hit rates for our predictor than when the predictor uses the features selected using stream timing information. However, both the correlation method and the PCA method successfully select one set of features that generates comparable or higher hit rates to the feature set selected by the stream timing information method. Our results also show that retransmitted packets, a leading indicator and the first visible application-layer indicator of the presence of network congestion, is the feature that has the strongest effect on the predictor's hit rates, and that in fact a predictor using only retransmitted packets can predict user quality ratings with a very high degree of accuracy.

There are several remaining questions we leave for future study. First, it is unclear how much the feature selection depends on the characteristics of the streams used in this study; one area for further exploration would be to apply these feature selection algorithms to data from other streams outside of the ones used in these experiments. More importantly, our ongoing work and ultimate goal in this area involves determining how such a prediction system could be implemented and executed in real time; to this end, we are exploring the timing requirements of each of these feature selection mechanisms, to determine the feasibility of using the feature selection mechanisms in real time.

ACKNOWLEDGMENTS

The authors would like to thank Dave Musicant, for contributing his data mining expertise on the development of the stream quality predictor, and Anna Sallstrom, for her contributions to the implementation of the CLeVer algorithms. We would also like to thank Mike Tie for his technical support, particularly in the setup and execution of the user experiments.

REFERENCES

[1] Ashmawi, W., R. Guerin, S. Wolf, and M. H. Pinson. 2001. "On the impact of policing and rate guarantees in Diff-Serv networks: A video streaming application perspective." In Proceedings of SIGCOMM 2001. San Diego, CA.

[2] Breese, J. S., D. Heckerman, and C. Kadie. 1998. "Empirical analysis of predictive algorithms for collaborative filtering." p. 43–52. In Proceedings of the Fourteenth Annual Conference on Uncertainty in Artificial Intelligence.

[3] Calyam, P., W. Mandrawa, M. Sridharan, A. Khan, and P. Schopis. 2004. "H.323 Beacon: An H.323 application

related end-to-end performance troubleshooting tool." In Proceedings of NeTS 2004. Portland, OR.

[4] Carson, M., and D. Santay. 2003. "NIST Net: a Linux-based network emulation tool." *SIGCOMM Comput. Commun. Rev.* 33 (3):111–126.

[5] Clark, A. D. 2001. "Modeling the effects of burst packet loss and recency on subjective voice quality." In Proceedings of the IP Telephony Workshop. New York, New York.

[6] Csizmar Dalal, A., D. R. Musicant, J. Olson, B. McMenamy, S. Benzaid, B. Kazez, and E. Bolan. 2007. "Predicting user-perceived quality ratings from streaming media data." In Proceedings of ICC 2007. Glasgow, Scotland.

[7] Csizmar Dalal, A., and E. Perry. 2003. "A new architecture for measuring and assessing streaming media quality." In Proceedings of PAM 2003. La Jolla, CA.

[8] Csizmar Dalal, A., and K. Purrington. 2005. "Discerning user-perceived media stream quality through application-layer measurements." In Proceedings of MSAN 2005. Orlando, Florida.

[9] Dunham, M. H. 2002. *Data Mining: Introductory and Advanced Topics*. Prentice Hall.

[10] G.107, I.-T. R. 1998. "The Emodel, a computational model for use in transmission planning." Recommendations of the ITU, Telecommunications Sector.

[11] Hand, D., H. Mannila, and P. Smyth. 2001. *Principles of Data Mining*. MIT Press, Cambridge, MA.

[12] Keogh, E., and C. Ratanamahatana. 2002. "Exact indexing of dynamic time warping." p. 406–417. In Proceedings of the 28th International Conference on Very Large Databases. Hong Kong, China.

[13] Loguinov, D., and H. Radha. 2001. "Measurement study of low-bitrate Internet video streaming." In Proceedings of IMW 2001. San Francisco, CA.

[14] Nichols, J., M. Claypool, R. Kinicki, and M. Li. 2004. "Measurement of the congestion responsiveness of Windows streaming media." In Proceedings of NOSSDAV. Kinsdale, Ireland.

[15] P.910, I.-T. R.a. "Subjective video quality assessment methods for multimedia applications." Recommendations of the ITU, Telecommunications Sector.

[16] Sakoe, H., and S. Chiba. 1978. "Dynamic programming algorithm optimization for spoken word recognition." *IEEE Trans Acoustics Speech Signal Process* 26:43–49.

[17] Tan, P., M. Steinbach, and V. Kumar. 2005. *Introduction to Data Mining*. Addison-Wesley.

[18] Vlachos, M., M. Hadjieleftheriou, D. Gunopulos, and E. Keogh. 2003. "Indexing multi-dimensional time-series with support for multiple distance measures." p. 216–225. In KDD '03. ACM Press, New York, NY, USA.

[19] Wang, Y., M. Claypool, and Z. Zuo. 2001. "An empirical study of RealVideo performance across the Internet." In Proceedings of IMW 2001. San Francisco, CA.

[20] Wolf, S., and M. H. Pinson. 1999. "Spatial-temporal distortion metrics for in-service quality monitoring of any digital video system." In Proceedings of SPIE International Symposium on Voice, Video, and Data Communications. Boston, MA.

[21] Yoon, H., K. Yang, and C. Shahabi. 2005. "Feature subset selection and feature ranking for multivariate time series." *IEEE Trans. Knowledge Data Eng.* 17 (9).

Biography

Amy Csizmar Dalal is an Assistant Professor of Computer Science at Carleton College in Northfield, Minnesota. She received the B.S. degree in Electrical Engineering from the University of Notre Dame, and the M.S. and Ph.D. degrees in Electrical Engineering from Northwestern University. Previously, she worked as a post-doctoral researcher at Hewlett-Packard Laboratories in Palo Alto, California, on measurement infrastructures for large-scale computing systems, specifically for streaming media quality assessment. She continues her work on streaming media quality assessment at Carleton. Her research interests include computer communication networks, measurement, performance analysis, streaming media, and queueing theory.

Jamie Olson is a 2007 graduate of Carleton College, where he received B.A. degrees in Computer Science and Cognitive Studies. Starting in the Fall of 2007, he will be pursuing a Ph.D in the Computation, Organization and Society program through the Institute for Software Research International in the School of Computer Science at Carnegie Mellon University.