VastMM-Tag: Semantic Indexing and Browsing of Videos for E-Learning

Mitchell J Morris

Submitted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy
in the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY
2012

ABSTRACT

VastMM-Tag: Semantic Indexing and Browsing of Videos for E-Learning

Mitchell J Morris

Quickly accessing the contents of a video is challenging for users, particularly for unstructured video, which contains no intentional shot boundaries, no chapters, and no apparent edited format. We approach this problem in the domain of lecture videos though the use of machine learning, to gather semantic information about the videos; and through user interface design, to enable users to fully utilize this new information.

First, we use machine learning techniques to gather the semantic information. We develop a system for rapid automatic semantic tagging using a heuristic-based feature selection algorithm called Sort-Merge, by using large initial heterogeneous low-level feature sets (cardinality greater than 1K).

We explore applying Sort-Merge to heterogeneous feature sets though two methods: early fusion and late fusion. Each takes different approaches to handling the different kinds of features in the heterogeneous set. We determine the most predictive feature sets for key-frame filters such as “has text”, “has computer source code”, or “has instructor motion”. Specifically we explore the usefulness of Harr Wavelets, Fast Fourier Transforms, Color Coherence Vectors, Line Detectors, Ink Features and Pan/Tilt/Zoom detectors. For evaluation, we introduce a

"keeper" heuristic for feature sets, which provides a method of performance comparison against a baseline.

Second, we create a user interface to allow the user to make use of the semantic tags we gathered though our computer vision and machine learning process. The interface is integrated into an existing video browser, which detected shot-like boundaries and presented a multi-timeline view. The content within shot-like boundaries is represented by frames to which our new interface applies the generated semantic tags. Specifically, we make accessible the semantic concepts of 'text', 'code', 'presenter', and 'person motion'. The tags are detected in the simulated shots using the filters generated with our machine learning approach and are displayed to users using a user-customizable multi-timeline view. We also generate tags based on ASR-generated transcripts that have been limited to the words provided in the index of the course text book. Each of these occurrences is aligned with the simulated shots. Each spoken word becomes a tag analogous to the visual concepts. A full Boolean algebra over the tags is provided to enable new composite tags such as 'text or code, but no presenter'.

Finally, we quantify the effectiveness of our features and our browser through user studies, both observational and task driven. We find that users that use the full suite of tools performed a search task in 60% of the time of users without access to tags. We find that when users are asked to perform search tasks they follow a nearly fixed pattern of accesses, alternating between the use of tags and Keyframes, or between the use of Word Bubbles and the media player. Based on user behavior and feedback, we redesigned the interface to group spatially interface components that are used together, removed un-used components, and redesigned the

display of Word Bubbles to match that of the Visual Tags. We found that users strongly preferred the Keyframe tool, as well as both kinds of tags. Users also either found the algebra very useful or not useful at all.

Table of Contents

List of Figures, Tables & Equations

To Kristen, my supportive wife, without whom I would have given up.

1. Introduction

1.1. Motivation

This thesis explores a way of making unstructured videos efficiently accessible, particularly those of lectures. In a modern university there is a wide range of classes that are video recorded for the benefit of students. The availability of this media makes the information in the classroom more accessible. Students who are unable to attend regular classes because of time commitments or distance can take advantage of their availability to participate in courses they would otherwise be unable to attend. Students who regularly attend classes can benefit also by their use as review material.

A video in its raw form is sequentially accessed. For students who are unable to attend the lectures, this mode of viewing may be sufficient as it replicates the lecture experience. However, the use of a lecture video as review material is impeded by the sequential access. In contrast standard review materials in the form of textbooks and lecture notes are easily accessed by being leafed though. Book users can easily skip around and skim material to find the content they wish to read in detail.

The current state of the art in video browsing incorporates features to allow a user to more quickly access information in the video. Users are able to fast forward and reverse, seek though the video with the assistance of tables of contents or selected thumbnails, or speed up the playback of a desired section. However, the limitation of these approaches is that the user is only

able to glean a small amount of semantic information without watching sections the video itself and what semantic information is available, typically as chapter titles in a table of contents, must be input by a person.

We use automated semantic information tagging, through the use of machine learning techniques, to solve this problem. Unstructured video has the complication that there is little a priori knowledge about the video that can help in the tagging process. As such, it is difficult to know what visual information would be useful for the tagging task.

1.2. Approach

This thesis explores the discovery and effective use of automatic semantic tags for navigating through video key frames in the unstructured video presentation domain. We approach this problem using machine learning and user interface techniques. We use machine learning to automatically generate classifiers for selected visual concepts, which may be useful for persons needing to review the video. These classifiers are used to categorize frames, automatically identified as shot boundaries, as belonging to one of these concepts. Automatic speech recognition techniques are also used to capture topic- related information spoken by the lecturer. All this semantic information is presented to the user through a multimodal timeline interface.

As input to the machine learning, we use a variety of low-level visual features. We explore feature-selection techniques to select only those features that are required for each classification task. We explore extensions to an existing feature-selection algorithm and build

the classifiers using Support Vector Machines (SVMs). Then, we explore the usefulness of these tags and the related user interface components through user studies.

As training data to create the classifiers, we used recorded lecture videos from the Columbia Video Network consisting of more than 2000 lectures distributed over 67 courses. Together they comprise about 3000 hours of unstructured video. For the user studies, we used 26 Columbia University students. Each participant was given the same 5 search tasks on the same video. Each participant was randomly assigned to an experiment group, each group having a different suite of search tools available.

1.3. Tags

We have identified a large number of semantic tags that we consider useful to someone watching a lecture video. These tags are organized into an ontology that groups them together by similar characteristics. Concepts were added to this ontology based on our intuition on what visual cues an audience member might remember about a lecture. We then selected a portion of these tags to implement classifiers for. We search for known useful features for picking out the information that each tag represents and built and trained classifiers based on these features.

1.4. Machine Learning

This is a machine learning problem that has a very large set of features that must be selected from. A very large set of features is unsuitable for fast and accurate classification. To generate the needed semantic tags, we use Support Vector Machines in conjunction with Sort-Merge feature selection. In order to apply Sort-Merge to a larger heterogeneous feature set, we

explored several modifications to the Sort-Merge algorithm, for example, the Keeper Heuristic, feature fusion methods, and fast cutoff.

1.4.1. Low-Level Visual Features

We relied on previous work to guide us when identifying which low-level visual features may be useful in the classification of our tags. Features were chosen that had previously been shown to perform well at classification tasks similar to those of our tags. Specifically, we explore the usefulness of Harr Wavelets, Fast Fourier Transforms, Color Coherence Vectors, Line Detectors, Ink Features, and Pan/Tilt/Zoom detectors.

1.4.2. Feature Selection

Features that are redundant increase the cost of classification but do not help the performance. Features that introduce noise negatively impact the performance and the classification time. From the very large feature set, a smaller set of features must be selected to bring classification time down and to maintain, and possibly improve, the classification accuracy. The smaller feature set will then be used to apply tags to the frames.

The state of the art is that feature selection either addresses a small number of features, or a large number of similar features. Our domain necessitates a large set of different features. Because in the feature sets we use there are a high number of redundant features, the speed of the selection is more important than the choice of the optimal feature set. As such, novel efficient feature selecting approaches were developed.

We investigate the usefulness of the Sort-Merge algorithm for this feature selection. Previous work on Sort-Merge showed several decimal orders of magnitude improvement in classification time as well as increases in classification precision when applied to a large homogeneous feature set.

1.4.2.1. Keeper Heuristic

At each step of the Sort-Merge algorithm, features are grouped into subsets. The performance of such sets at the classification task is compared to a baseline performance to determine their suitability as "keepers" (Figure 10). A *keeper* is a feature set, of any cardinality, at any level of the Sort-Merge tree, which, when evaluated performs better than the baseline performance. Keepers are then later compared against themselves to decide on the best keeper, that is, the best subset of features.

We have discovered the consequences of separate approaches: treating all feature types the same, called here *Early Fusion*; and second, selecting the 'best' features of a type and then selecting amongst those best, called here *Late Fusion*. In the experiments we conducted *Early Fusion* performs better that *Late Fusion*. However, *late fusion* is faster.

1.4.2.2. Fast Cutoff

We observed that there exists a level of the Sort-Merge tree where all attempts at merging feature sets produces decreased performance; this appears to be related to the problem of overfitting. The best keeper always comes from this level or an earlier level. This property of the

Sort-Merge algorithm can be used to terminate the Sort-Merge Tree early in a method we call *Fast Cutoff*. Early termination of the Sort-Merge process saves the ever more costly steps at the higher levels of the Sort-Merge tree.

1.4.2.3. Classifiers

The final classifiers were used to classify selected frames from the videos. These frames were selected by an algorithm designed to detect shot-like boundaries in unstructured videos. *Shot-like boundaries* are either short regions with high intensity change, similar to jump cuts, or extended regions with gradual intensity change, similar to wipe cuts. The algorithm chooses a frame at the boundary to represent the shot-like segment.

1.5. User Interface

We created a new video browser, VastMM-Tag, to provide users with data generated by the semantic tagger. The multi-Tag Timeline is the defining characteristic of VastMM-Tag. It shows the user a number of timelines, each corresponding to a semantic tag. This display gives the user an overview of where in the video semantic concepts occur.

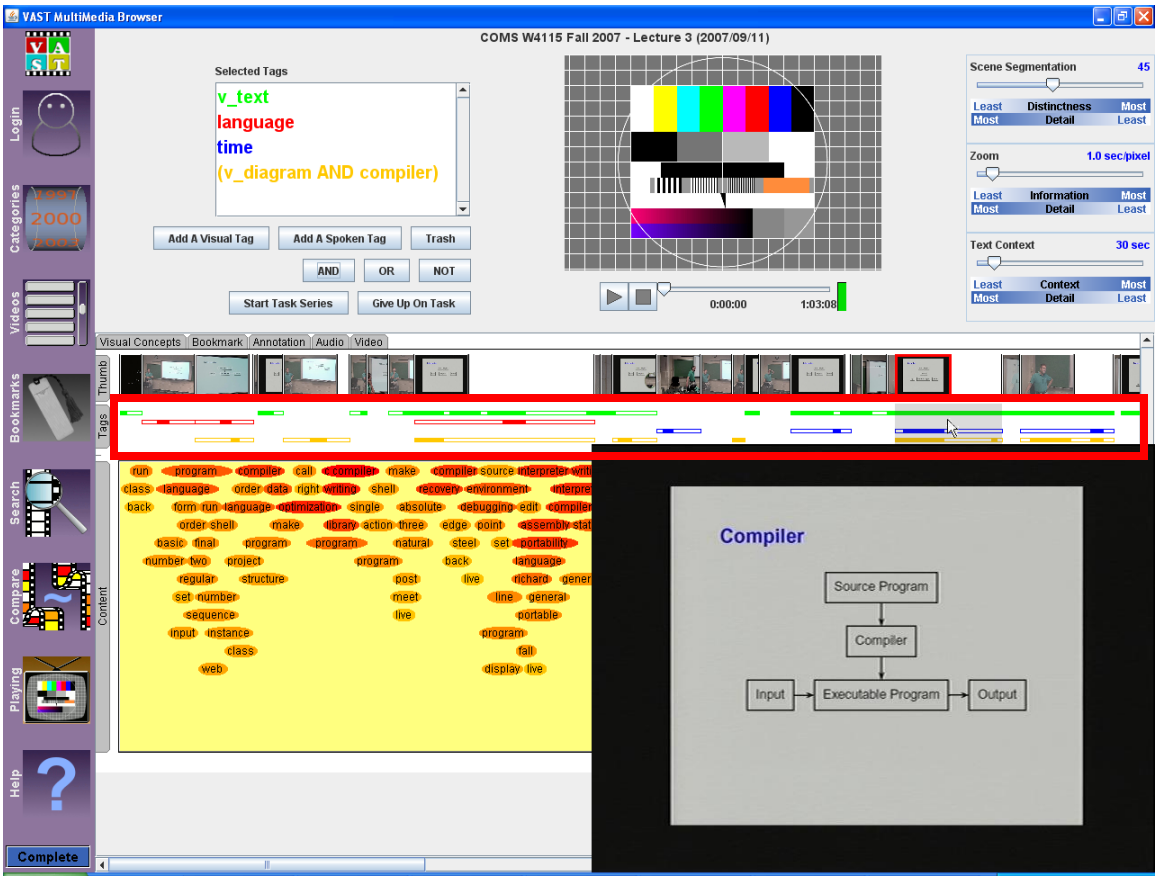

Figure 1: Screenshot of VastMM-Tag Browser with Tag Timelines highlighted with the red box.

1.5.1. Tag Timelines

The goal of the Tag Timeline display is to assist in locating sections of the video where concepts occur or where concepts co-occur with each other. An example can be seen in Figure 1. Users are able to customize their view by selecting timelines. Each timeline displays the presence or absence of the tag during segments of the video between two shot-like boundaries. Because there is a degree of uncertainty in the boundary detection, the user is able to adjust the threshold for which a boundary is considered meaningful enough to display. A change in this

threshold will also change the timeline display from solid to hollow timeline bars to show uncertainty of the tag location.

1.5.2. Tag Algebra

To assist in the goal of showing collocations of tags, a full Boolean algebra over the timelines is provided. This allows users to create timelines corresponding to more complex concepts, for example ‘text and not presenter’. Once these timelines are created, the bars they produce are displayed with the same solid bar / hollow bar method as the single concept timelines.

1.5.3. Word Tags

During initial user trials information about the spoken content of the lecture was displayed in a word bubble display. This display gave an over view of the content by showing bubbles containing words spoken by the lecturer, these bubbles were time aligned to the video. Uses suggested that the interface was difficult to use. Consequently we developed a new display method for spoken word data by modifying the display to be consistent with the tag timelines.

1.5.4. User Studies

Finally we have performed user studies to evaluate the effectiveness of the browser. The user studies took two forms. The first was a task-oriented study where participants were presented with a series of search tasks. Each task was to be completed in the shortest amount of time possible. In the second, we perform an observation study where students in a number of

lecture classes were afforded access to their recorded course materials in VastMM-Tag. Usage patterns were analyzed and visualized to evaluate what users found must useful.

1.6. Contributions

This work contributes the following to the state of the art of feature selection and video browsing for e-learning tasks.

- Application of Sort-Merge feature selection algorithm to new feature types.
- Modification of Sort-Merge algorithm to support the simultaneous use of different feature types.
- Introduction of Keeper Heuristic and fast cutoff to improve performance of Sort-Merge.
- New user interface techniques to display semantic tags to users of a video browser system and to enable Boolean algebra of those tags.
- Transition of display of spoken word related information in the browser from a bubble type display to be consistent with Tag Timelines as requested by users in the user study.
- An integrated pipeline to take lecture videos, automatically generate semantic tags, and prepare the data for user consumption through an enhanced video browser.
- Insights into usage patterns of video browser users, namely a strong co-occurrence of usage of pairs of interface components.

2. Previous Work

We review prior approaches to using machine learning and interactive browsing techniques on unstructured video. Since few papers address these topics all together, we enumerate the state of the art individually. A comprehensive survey is not attempted; rather the discussion of the state of the art will focus on the intersections of feature selection, Sort-Merge, low-level visual features, SVMs, video browsing, and evaluation methods.

2.1. Feature Selection

In the domain of unstructured video, feature selection is of paramount importance. Feature selection in the process of starting with a large number of features for use in building a classifier, and having a method of selecting the best subset of features from among the group, using those to create a the final classifier. The unstructured nature of the video makes it difficult to determine what elements of a video are of key significance for making a particular classification decision. The naive approach would be to use as many features as possible for the classification task. The problems with this method are two fold. Firstly, in a sufficiently large feature set, classification is slower than necessary. Secondly, not all features are equally useful [1], and some actually degrade performance.

Forward selection, backward elimination, and genetic algorithms are the three fundamentally types of feature-selection algorithms. Forward selection and backward elimination are the most simple [1]. Both are often wrapper methods of selection: in each step of the

selection process *wrapper methods* use the chosen classifier to evaluate the proposed feature set. Forward selection works by a greedy approach. At each step, one feature is added to the feature set. The feature chosen is the one that improves the performance the most. Backward elimination is similar but works in reverse. Initially all features are in the feature set. Features are greedily removed from the set at each successive step. Various selection methods exist that derive from forward selection and backward selection. Some methods combine elements of both forward steps and backwards steps [2].Others introduce weighting or randomness [1]. Still others user a filter method for choosing features: features are evaluated by a metric that is different from using the final intended classifier [1, 2]. For example, a filter method may use a measure of a feature's information gain, in place of the wrapper methods' use of classification performance. The intent of such methods is to use filters that take less time than the wrapper method would have [1]. Feature-selection algorithms can be classified by their method of search, generation of successors in the search, and their evaluation measure [2].

2.2. Sort-Merge

Sort-Merge takes a heuristic grouping-based approach rather than an iterative search-based approach. Like forward and backward selection, Sort-Merge is a wrapper method. Initially each grouping consists of only one feature. At each step of the selection process each grouping of features is evaluated for classification performance. In the sort step, the groupings are ranked by their performance, best performing first. In the merge step, ranked neighbors are merged into new feature groupings. The first and second groups are merged together, the third and fourth are merged together, and so on. In each successive of Sort-Merging there are half as many feature

groupings as in the prior round, and each grouping has twice as many features in it. Successive rounds of Sorting and Merging occur until a feature group is found that meets a predetermined condition [3].

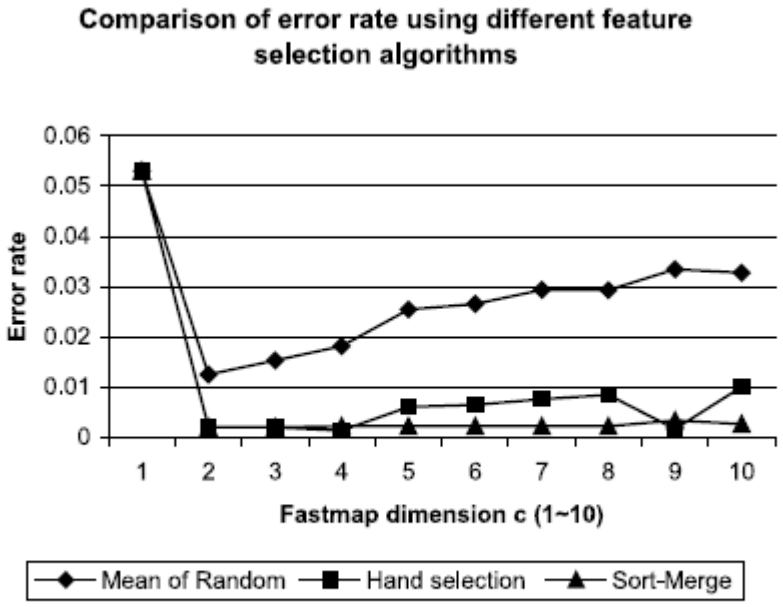

Figure **2**: **Graph taken from [3] showing the efficacy of sort merge feature selection compared to random selection and hand selection.**

Sort-Merge is designed to work in very high dimensional feature spaces. In [3] 1800 features consisting of the six values of color in each of the 300 MPEG-1 macro blocks were used. This is a large homogeneous feature set. Figure 2 shows a graph comparing the error rate of sort merge selection to the error rate of random and hand selected features sets after different amounts of dimensionality reduction [3].

This thesis explores the space of a large heterogeneous feature set with a focus on low-level visual features. In the previous work, the stopping condition of the Sort-Merge algorithm

was not defined and left to the user. We explore the Keeper Heuristic method of choosing the best feature set. Keepers are defined as feature sets that outperform a baseline measure. Keepers are then later evaluated to decide on the best subset of features.

2.3. Low-Level Features

In video recognition there many ways to generate features. Many of the features used in the video domain have been used in the image domain. These are features that deal with the content of a single image. Low-Level image features come in the following categories; color, texture, and shape.

2.3.1. Color

Color features can be a measure of the average color of an image. The average color can further be separated into averages by color channel [4]. In addition to average color, the second and third order moments (variance and skewness) can be used as color features [4]. Color histograms are another common color feature. Each color channel is divided into a number of buckets. The color of each pixel falls into a particular bucket. This forms a histogram of color channel values, which is then used as a feature vector [5]. Color Correlograms express how the spatial correlation of colors changes with respect to distance between those colors in an image [6,7].

2.3.2. Texture & Shape

Texture in an image can be measured in many ways. One is by using a co-occurrence matrix of grayscale pixel values [8]. Texture neighborhood is a measure of the probability of

increased brightness in a pixel's 8-neighborhood [9]. Wavelet features sit in the middle of color and texture features. A wavelet can be used to measure color, texture, or a combination of both. Wavelet color covariance is a combined measure of texture and color [8]. Wavelet K-means clustering is a texture measure [10] that finds pattern of similar spatial frequencies. The shape of an image can be measured by edge direction histograms or their co-occurrence matrices [10]. Fast Fourier transforms can also be used to measure shape. An FFT feature can be made rotation or affine transformation invariant through modification: Polar Fourier features are rotation invariant, Log-polar Fourier features are affine transformation invariant [9]. Wavelet features and FFT features have proved to be very useful thus far. Currently they have been used to recognize frames with readable text and computer code [11].

2.3.3. Motion

There are also features that go beyond the image domain. Features that detect various kinds of motion are unique to video. Several approaches for detecting motion are currently accomplished by deriving features from the block motion vectors used in MPEG encoding. Since block motion vectors are a heuristic tool for image compression [12], recording of motion is not their purpose: using block motion vectors as a motion measure is a simplified approximation. Alternatively, motion in an image can be broken up into background and foreground motion. Background motion is often measured using the MPEG encoding motion vectors in macro blocks that are on the periphery of the video. Foreground motion is often measured by the difference between the motion in the center of the frame and the background motion [13]. These measures depend on availability of foreground and background, segmentation in a shot, and the relative

sizes of each. In the unstructured video domain often the frames contain pure background. Motion features appear to be useful in applying tags that involve actions by the speaker, for example, the actions of emphasizing information on the blackboard or slide.

2.3.4. Camera Motion

The motion of the camera can also be captured by visual features. There are three basic camera motions that relate to the effect on the image reproduced in the video: Tilt, Pan and Zoom. Tilt corresponds to a vertical movement of the image, pan corresponds to a horizontal movement of the image and zoom corresponds to the increase or decrease in size of the subject of the image. These features produce a response that indicates the number of pixels of change in the corresponding type of camera motion [14]. These measures are unable to distinguish between different camera moments that can produce similar results. For example, both a zoom and a dolly would be understood by this feature to be a zoom. Camera motion features appear to be useful for pinpointing when the motion detected in a video comes from movement of the subject verses movement of the camera or background.

2.3.5. Dimensionality Reduction

A difficulty of the video domain is the data set size. There are a number of algorithms that seek to reduce the problem into a manageable size through a method that does not affect the final performance of the final classifier. One method combines random feature subspace selection with training example subspace sampling. Multiple subsets are created and used to train independent classifiers. These classifiers are then used in concert to perform the classification

task [13]. Subsampling randomly chooses images from the training corpus, again constructing multiple classifiers [13]. Training example subsampling has proved useful in combination with the Sort-Merge Algorithm and has decreased the running time markedly [11]. However the combination of Sort-Merge with boosting is likely to be too costly in terms of running time to pursue.

2.4. Support Vector Machines

In the classification of unstructured video frames it is difficult to know which features of a frame are important and to what degree the importance is. Thus we must rely on machine learning to make these determinations. Currently Support Vector Machines (SVMs) are among the most well-understood forms of machine learning available.

2.4.1. Basics

Support Vector Machines [3] work by trying to find a separating hyperplane in a high dimensional feature space. On one side of this hyperplane lie positive examples, on the other lie negative examples. The hyperplane is selected so that it is maximally distant from both classes. It is not always possible to find a hyperplane that completely separates the classes.

To solve this problem, SVMs apply a Kernel Function to the feature space to transform it into a higher dimensional space in an attempt to find a hyperplane of complete separation. The inputs that lie closest to the hyperplane are termed Support Vectors. They are saved to define the plane and are used in the classification task [5]. There are instances where a complete separation cannot be achieved. In such cases a penalty is applied to the fitness measure when positive

examples exist on the negative side and when negative examples exist on the positive side. The penalty amount is a tunable parameter [5], as are the specifics of kernels.

2.4.2. Enhancements

Ways to improve SVM accuracy and speed are extremely useful in the video domain. One way to improve SVM accuracy is to generate artificial training examples. This is first done by training an SVM and using the chosen support vectors as a basis for generating the artificial positive examples. Transforms that modify the image, without changing it to a negative example, are also used to generate the artificial examples. For example, reflection or translation of the image can be used [15]. A further method to improve the speed of an SVM is to reduce the number of support vectors needed so that the classification task takes less time. The method involves finding vectors and weights for each vector with only a small loss in performance, accomplished through a search [15]. It is also possible to increase the accuracy of an SVM by combining multiple kernels, for example, using a linear weighed sum of kernels for classification [16].

We have decided to use a “normal” SVM without any of the enhancements described above. The selection of a wrapper method magnifies any additional time that the accuracy enhancements need. Similarly the reduced feature set provided by the selection algorithm proves speedy enough for our application. Because each successive level of the Sort-Merge algorithm uses the SVM on increasingly large feature sets, to include enhancements to the SVM that increase the computation cost is intractable. Additionally, due to the size of the feature sets being used, redundant features are abundant. Such redundancy allows the approximate feature sets

produced by the Sort-Merge algorithm to be adequate for the task without the above enhancements.

2.5. Video Browsers

Communicating the classification of video data is an important task. Previous work in the area of video browsing addresses issues relating to assisting the user to navigate a video [17] or to perform an information retrieval task [18]. We use classification methods to assist users in navigating through videos.

2.5.1. Browser Enhancements

At the most basic level, video browsing is accomplished in the same way as a VCR. A user can play, fast-forward, rewind, or jump to a desired time in the video. Additional methods exist to enhance each component of the play/fast-forward/rewind model. Methods exist to speed up a user's viewing of a video in play mode [17]. However, the standard fast-forward use either omits audio or distorts its pitch to be unrecognizable. There exist algorithms that allow a user to play a video at an increased rate, by either removing pauses or by speeding up the playback while keeping the audio stream intelligible [17].

The methods of seeking through the video can be enhanced by providing a user with context. In the basic model, the only context a user has to navigate is that of a timeline. To enhance this, tables of contents and shot boundaries can be employed. The user can then skip forward or backward from one division to another, or skip to an arbitrary section guided by a

table of contents heading or shot boundary key frame [17]. This method is limited by the granularity of the sections.

2.5.2. Multimodal Timeline

VASTMM is a video browser designed to work with unstructured video. Shots and spoken words are extracted from these videos and displayed along a timeline to assist in video browsing. The shot boundaries identifying keyframes and the recognized spoken words are displayed separately along the timeline of the video. A user can use either the keyframes or the words to jump to a section of interest [18]. Figure 3 shows a screen shot of the VASTMM video browser, and its multimodal timelines.

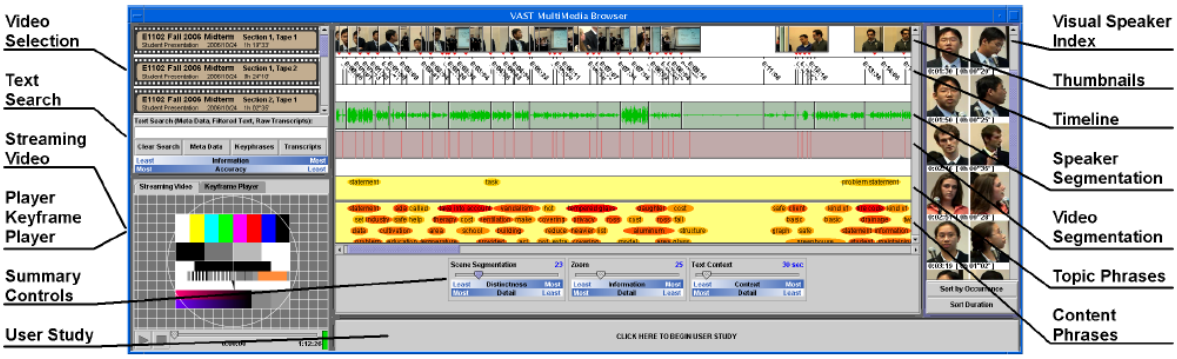

Figure **3: A screen shot of the VASTMM browser [18].**

2.5.3. Tag Display

A second multi-timeline method of video browsing has been used by MSNBC to assist users in viewing the 2008 Presidential debates. Figure 4 shows a screen shot from the viewer. On the timeline there were highlights corresponding to which candidate was speaking at the time. Also, a user was able to select a political topic of interest, and those sections would be

highlighted as well. Then when users play back the video, the frame will be highlighted with a colored border representing the tag or tags that the current frame or segment is a part of. We expect that this will have the benefit of allowing a user to see the classification while viewing and to assist in fast-forward and rewind seeking. However this debate viewer was annotated by hand [19].

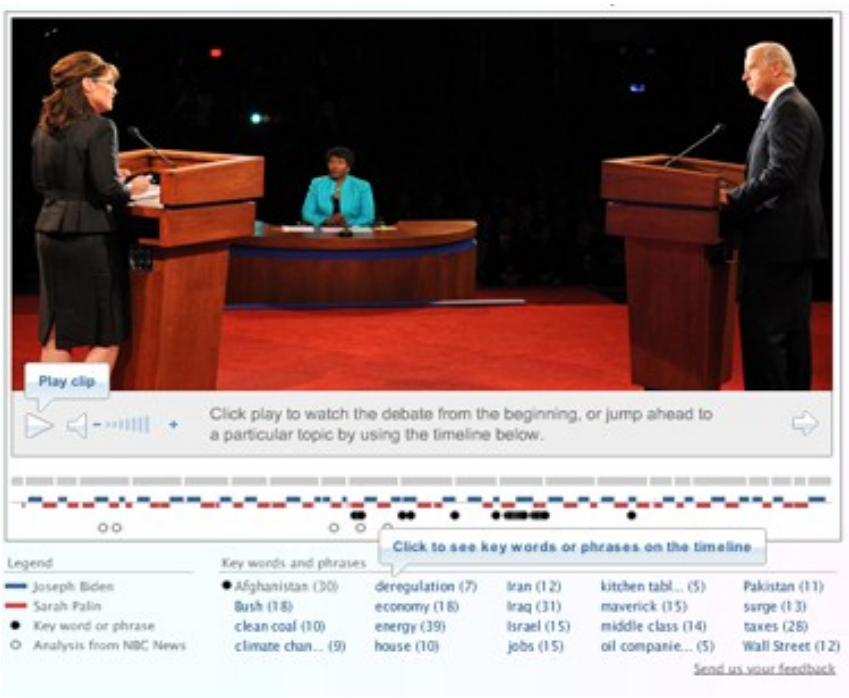

Figure 4: Screenshot of the MSNBC debate viewer [19].

2.6. Use of Information Retrieval / Classification Techniques

Currently video classification techniques are used for answering queries against a database of videos. Techniques are used to formulate a query and then display to the user a series of video segments of potential interest [20]. In the video browser context, classification of

keyframes or shots is not often used to assist the user in more quickly viewing or understanding the content of the video. This is particularly true for unstructured video. We apply automatic semantic tagging to enhance the video browsing experience through tagged frame highlighting, multimodal timelines, and tagged-segment-only playback.

2.7. Evaluation

Evaluating the efficacy of an image classification system is of major importance for assessment and improvement. Evaluation methods that compare the classifiers prediction to ground truth are borrowed from machine learning. Similarly, user studies can be used to measure the use of the full system. Notwithstanding, there is a lack of standards on how to measure such values.

2.7.1. Data-Oriented

There has been much work in evaluating the performance of the machine-learning process. The retrieval performance of a classification task can be measured in a number of ways. Precision is the percentage of correctly identified positive examples [21]. In a ranked list this measure can be applied to only a portion of the list, therefore precision at 100 only considers the first 100 items in the ranked list [22]. Recall is a measure of how many of all possible positive examples were tagged positive by the classifier. Recall is also reported as a percentage [21]. Recall may also be applied to a ranked list in a similar way to precision [22].

A Precision-Recall curve is a graph of values with precision on the y axis and recall on the x axis. Precision-Recall curves are relevant to a ranked list of retrieved positive examples.

This curve represents how Precision and Recall both vary as the number of results considered is increased from 1 to n [21]. Average precision is a measure similar in spirit to Precision-Recall curves but represented as a real number. Average Precision weights the importance of correctly retrieved documents higher in the list [21]. Receiver operating characteristic curves are another means of evaluating retrial results; a ROC curve is analogous to a Precision-Recall curve [21].

2.7.2. User-Oriented

Just as important as the above retrieval performance metrics, is the impact that the system has on users. User studies where users are asked to perform tasks are equal measures of the quality of an information retrial system [20]. User studies when applied to video browsing software observe the frequency and situations under which the browsing features are used. As a control, each user in the study is also asked to complete the browsing tasks using a default browser with only a base set of browsing tools [17]. Constraints are also placed on the browsing task in order to mimic situations where the researchers hypothesize their browser to be most useful [17]. We examine the usefulness of the enhanced video browser for the tasks of initial comprehension and of review of already viewed videos. The task completion rate is a metric used to evaluate usefulness of a browser's features. A task is considered incomplete if the test subject gives up, or fails at the task [18]. The accuracy of the tasks and time to completion are also measured for the tasks [18]. These measures are then used to determine if a browser assists the task. User satisfaction surveys are also used to assess the quality of the user interface. Such surveys ask users to report on their subjective experience using the software. Two approaches are used to gather this information. The first are specific statements where a user is asked to rate

their agreement on a scale of 1 to 5. The second are open-ended opinion questions where participants are asked to elaborate on their experience. [17].

3. Semantic Tags

We have identified several semantic categories that we believe to be useful when viewing lecture and presentation videos with the intention of using this additional information to enhance the user interface of our video browser. Users interested in the video at their first time watching, we assume, will be interested in only those parts where there is content to take in, as opposed to other content, such as introductory remarks, periods of technical difficulties, etc. Users interested in the video for review or studying purposes may have different viewpoints and recall strategies. They may be trying to find places in the video to review based on the visual cues they remember from watching the lecture the first time around. Currently we have analyzed the rapid selection of features for the tags “text”, “code” and “presenter”. The “text” tag indicates the occurrence of readable text, both hand written and computer generated, visible in the frame. The “code” tag indicates when there is text that is an example of computer visible in the frame. The “presenter” tag indicates when the frame contains a shot of the person giving the lecture.

3.1. Tag Ontology

We have constructed a tag ontology to represent the relationships between our proposed tags. The final tag ontology is show in Figure 5 . There are two classes of tags: the first class consists of tags that represent content, the second class consists of tags that represent participants. This ontology was used to inform the process of selecting tags to develop. This representation of the domain also guides the design of cascaded classifiers.

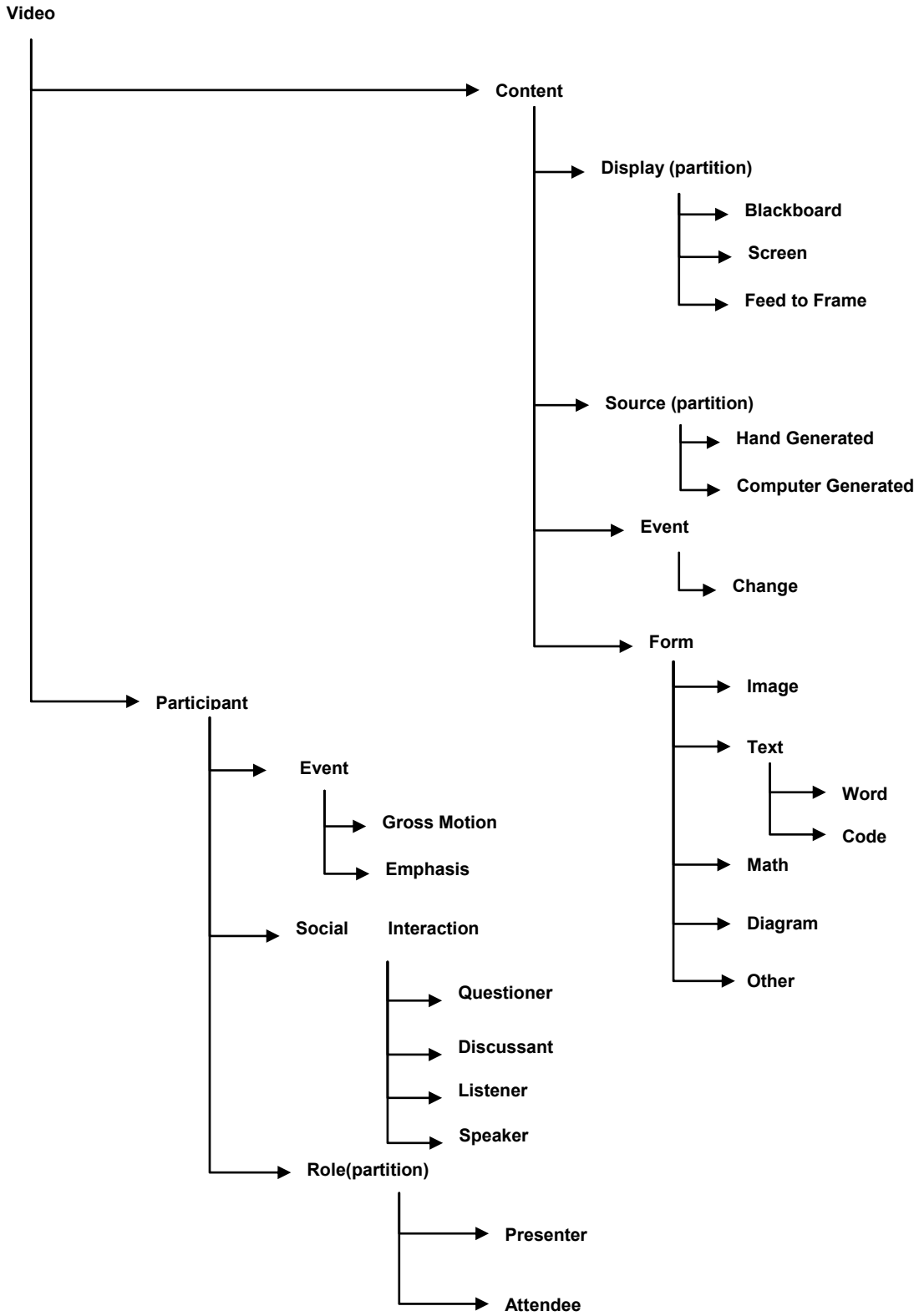

Figure 5: Tag ontology diagram

3.1.1. Content

The creation of the content section of the ontology was informed by the supposition that a student might have a search intent such as "I want to review the diagrams that the professor drew on the blackboard". This statement specifies three things about the content that the student is interested in. First, where the content was displayed: the black board. Second how the content was generated: by hand drawing. Thirdly, what content was produced: a diagram. This scenario among others, suggested three of the sub groups of content. **Display** tags relate to how the visual content is displayed to the audience. **Source** tags describe how the content was generated, either by hand or by computer. **Form** tags describe what form the content takes, e.g., text, image, etc. The final sub group, **Event** was informed by the observations that users might be only interested in viewing when new visual content has occurred. For example, a viewer might only want to skip ahead to the next slide in a power point presentation.

The **display** tag group consists of tags that describe how the visual content is displayed to the audience. Visual content used in lecturing usually takes two forms in the university setting, either content written on the blackboard, or slides projected on a screen. Some lecturers exclusively use one format or the other, but some frequently change between the two mediums. A third method of visual content, feed to frame, is when a slide that is projected on screen is mixed directly into the video, rather than having a camera record the slide projected on the screen.

The **source** tag group consists of two tags, computer generated and hand drawn. Initially the tags in the display group were considered to subsume this concept; anything on the

blackboard was by implication hand drawn. However in our dataset there were a number of times where digitally assisted handwriting showed up projected on the presenter's screen. In order to fully describe situations like the above, a separate group of tags to explicitly define computer generated versus handwritten would be needed.

The **form** tag group describes the form that the content takes, such as text and diagram. When giving a lecture and using a blackboard only, a lecturer will either write words on the board or draw a diagram to enhance explanations. However when presenters make use of digitally created slides more forms of content appear. The most notable are photographs or other images. Computer code also shows up on slides and usually fulfills a different role in the lecture than normal text. Mathematical formulas can also appear but are also distinct from text or diagrams. The form that the content takes is notably distinct from the source of the content: a diagram can be hand-drawn or computer-generated.

The **event** tag (i.e., content event) group only contains one event, namely change, this is the only event that visual content can take part in. This tag includes complete slide changes or when more content is added to a handwritten section.

3.1.2. Participant Branch

The participant branch was informed by the alternation of the camera between shots of the presenter and shots of the audience in some lectures, and by how this correlated with question and answer periods. The participant group is broken into three sub groups, **role** differentiates between the presenter and the audience, **event** (i.e., participant event) consists of

tags relation to on-screen motion, and **social interaction** has tags that indicate what the subjects on screen are doing in the context of the lecture.

The **role** subgroup, contains only two tags, presenter and audience. In a lecture there are one or more presenters, and anyone else is part of the audience. Frequently, lectures use shots of the presenter and sometimes use shots of the audience. The **event** group has one tag for gross motion, which means a person is moving around on screen rather than staying stationary. The other tag, emphasis, is for when the presenter is pointing and gesturing at visual content to emphasize a point or example. The **social interaction** tags get into specifics of what a lectures participant is doing while on screen. Listening and speaking are self-explanatory. Discussant is when there is a classroom discussion, and questioner is when an audience member is asking a question of the presenter.

3.1.3. Developed Tags

We selected “text”, “code” and “presenter” as three tags for which to create classifiers. We chose these based on the availability of previous work on their detection as well as their location in the ontology. “text” and “code” come from the content section an “presenter” from the participants section

3.2. Groundtruth Tagging Interface

In order to establish groundtruth for each of the video frames, we developed a web-based tagging tool. The tool is first loaded with all the images that need to be tagged. When a user first opens the tool, they are prompted to choose a tag to work with from a set of predefined tags, or

to create a new tag to work with. The user is then asked to step through each loaded image and decide if the image matches the tag or not. The tagging interface of the tool shows the user a large image of the frame under review as well as 8 smaller context images, to give the user a preview of what frames are to come and what frames have passed. See Figure 6.

The user indicates a decision by clicking on an interface button or by pressing an associated key. When a user tags a frame, those tagged positive are outlined in green, and those tagged negative are outlined in red. After a user tags a frame, they may edit their decisions, by pressing backspace or using the back function in the web browser, to move the interface back to the previously view frame, which they can then re-tag. The tool allows an accelerated tagging method because there often are times when there is a run of frames that all would have the same tag. In these cases the user can hold down the correct key, and frames will tick by, all getting tagged in the same way. The user can use the preview of the upcoming frames to judge when the key should be released. The tool automatically remembers the last frame that has been viewed. This way the user can exit and return to the tagging without losing their place.

This tagging interface was used to gather all the groundtruth that was used to create and evaluate the machine learning classifiers used in our experiments. The groundtruth was collected personally by the experimenters.

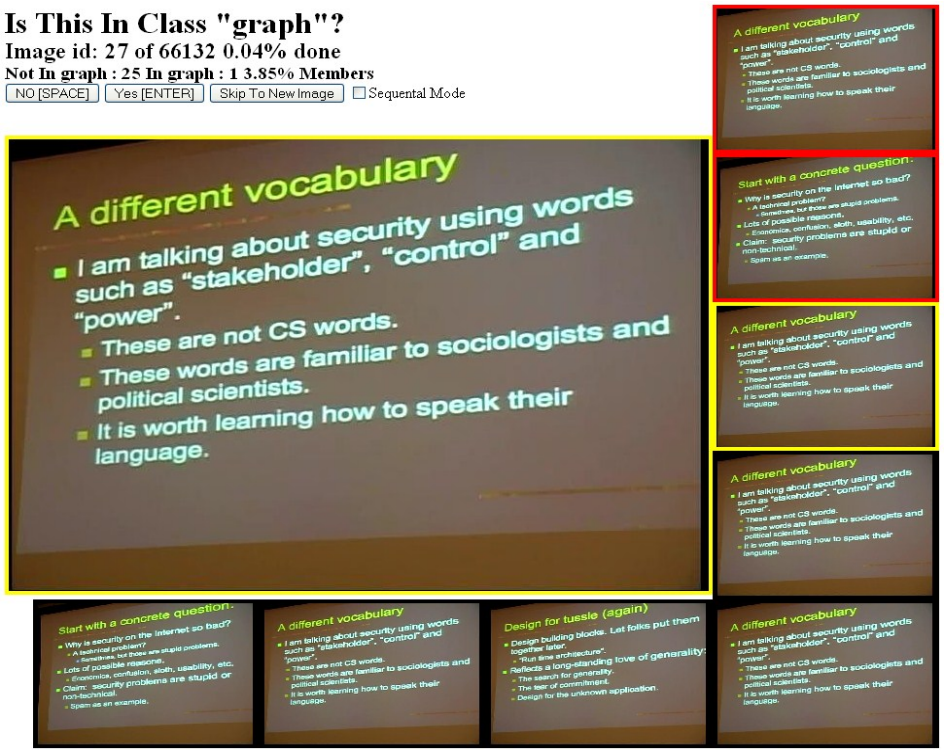

Figure 6**: Screen shot of Tagging Interface**

3.3. Visual Features

For the creation of classifiers, we examined the usefulness of a number of low level visual features. These features were used as inputs into the modified Sort-Merge feature selection algorithm for the creation of frame classifiers that were eventually used in the VastMM-Tag Browser.

3.3.1. Wavelet

For our wavelet features we have used Haar wavelets. The image is divided into 16 regions (a 4×4 grid). In each region the wavelet is calculated to three levels of spatial resolution. In each of the sub-bands (HH, LH, HL, & LL) of each level, the first 3 moments of the grayscale

pixel values are used as features. This results in 576 features per image. This feature is known to be useful in finding frames that contain readable text [23].

3.3.2. Autocorrelation

The "quick river" feature is a feature designed to detect text rendered in a fixed width font. The image is divided into 16 regions (4x4 grid). In each region, the pixel vales are summed down the columns. An auto correlation is then preformed on the series of column-sums. The autocorrelation will peak when the columns of each line up vertically. The offset at which the autocorrelation is greatest is used as a feature, along with the magnitude of the greatest auto correlation. This accounts for 32 features.

The quick river is also calculated to detect text imaged at an angle. For this inclined version of the feature, the autocorrelation is instead calculated along a direction perpendicular to the expected angle of text. For each of the 16 regions, this feature is calculated for angles -16 to 16, at every 4 degrees. There are 288 features per image (4 columns x 4 rows x 9 angles x 2 values). We developed this feature as a way to find text that formed a regular pattern. Although computer code is often displayed in a fixed width font, after several experiments we were unable to produce a reliable classifier and decided that this feature was not very useful in finding fixed width fonts.

3.3.3. Fast Fourier Transform

3.3.3.1. Band limited

We use a Fast Fourier Transform feature as another method for finding fixed-width rendered text. First all images were regularized to 320 pixels wide through cropping. As we noted through informal observation that text is inclined at a maximum of a 16 degree angle, we only consider a band of the full FFT that corresponds to that angle range for creating features. This restriction of angles was carried over from the experiments done with the autocorrelation feature showing that some text examples deviated slightly from true horizontal. The resulting FFT image is then decimated by a factor of 5. Each pixel value of the resulting image is used as a feature [9]. This results in 700 features.

3.3.4. Color Coherence Vector

We use color coherence vectors as a feature for picking out regions of skin color to classify frames as containing a "presenter". The image is divided into a 4x4 grid, and in each region the image is blurred by assigning the value of each pixel to the average of its 3x3 neighborhood. Connected regions of color are identified and separated into coherent regions and non-coherent regions; coherent regions are those which have a size greater than a threshold, in this case 50 pixels. We create a histogram by taking the first 3 bits of information from the red and green channels, and the first 2 bits of information from the blue channel. Only the pixels which are part of a coherent region are added to the histogram. The total number of pixels in each bin is then used as a feature giving 4096 features per image [24].

3.3.5. Pan / Tilt / Zoom Feature

To identify when the subject of the video is moving we use Pan / Tilt / Zoom features [14]. Based on pixel intensity changes in successive frames of a video the feature produces a numeric value for the amount of pan, tilt, and zoom. In order to use this measure as a proxy for motion of the subject, the feature was computed on the whole image, as a measure of gross motion, but then also on the top 1/3 of the image only, to capture camera-only motion. The response from the top 1/3 was subtracted from the response from the whole image to get an approximate value for just the remaining motion in the frame. We used only the pan values as a proxy for motion of the subject. We noticed that though use of the tags that zoom did not make a good proxy for person motion. It did not respond to the kinds of motion that we were looking at and rather responded to shot boundaries. Also, tilt did not make a good proxy for motion because most of the motion was a presenter moving horizontally on the screen or moving their arms which was also mostly horizontal.

3.4. Word Features

A second method of generating semantic data examines the words spoken by the presenter. In the VASTMM browser, there is a UI display that gives an overview of the content of the lecture called *Word Bubbles*. This display is based on an ASR generated transcript of the video and is time aligned with the video, to give the user an idea of how the content of the video changes over time. More detail is given on this UI element in Chapter 5.

Initial user studies we determined that this method of displaying the audio data was inadequate for assisting in search tasks, as opposed to the summarization tasks for which it was designed. This was discovered though the user studies we conducted, discussed in section 5.3. In studies conducted by Haubald & Kender [18] the word bubble interface was shown to be helpful for summarizing the content of the lecture. However in the search task user studies we conducted, users constantly rated word bubbles low on a satisfaction survey. Additionally study participants left commends suggesting improvements to the word bubble interface. We investigated two methods of making this data more useful for the search task domain. The first was using a filtering method based on Zipf's Law. The second was a Word Tags approach.

3.4.1. Pre-filtering

The first step in both approaches was a filtering method inherited from VASTMM. The ASR transcripts extracted from unstructured video have high noise content. The reason is the uncontrolled nature of the audio capture environment. As such the generated transcript has many words that are not relevant to the subject matter and also words that are recognized incorrectly. VASTMM dealt with this issue by filtering the transcript though a reference corpus. For the lecture videos this was the course textbook's index. This way only potentially course-related words would be considered. We expanded upon this idea by making modifications to the speech recognition environment. Instead of using the generic dictation dictionary provided by the speech recognition package, we used the course index combined with course related terms to build our own restricted dictionary. This restricted dictionary was then used to create the ASR transcript.

The course-related terms were gathered by an analysis of course syllabi from Columbia University Computer Science courses. This was done because, using only the course textbook index, words like “problem” and “homework” would be stripped out, even though they communicate searchable content. The words present in the syllabus, minus a list of stop words [25] and numerical phrases, were ranked according to the number of syllabi they appeared on.

The bottom 15% of the words (which accounted for 56% of the occurrences) were trimmed from the list. The remaining words were manually pruned for words specific to Computer Science or Columbia University. See **Table 1**.

Table 1: List of course related terms by occurrence count

Rank	Word	Count	Rank	Word	Count
1	Grade	61	38	edition	22
2	teaching assistant	61	39	study	21
3	Class	58	40	knowledge	21
4	office hour	56	41	textbook	21
5	professor	55	42	assigned	20
6	Project	50	43	requirement	20
7	Final	50	44	syllabus	20
8	instructor	47	45	group	19
9	Lecture	45	46	resource	19
10	assignment	44	47	report	19
11	homework	42	48	important	19
12	prerequisite	41	49	overview	19
13	Policy	41	50	academic	19
14	Email	40	51	lab	19
15	Reading	39	52	reference	18
16	introduction	38	53	late	18
17	Require	37	54	announcement	17
18	Midterm	36	55	building	17
19	Due	36	56	chapter	16
20	Exam	33	57	review	16
21	Text	32	58	handout	16
22	Date	32	59	understand	15
23	description	31	60	account	15
24	Credit	31	61	submit	15
25	Paper	29	62	version	15
26	Page	29	63	isbn	14
27	Room	28	64	issue	14
28	Schedule	28	65	staff	14
29	Book	27	66	appointment	13
30	Problem	27	67	answer	12
31	Location	27	68	permission	11
32	Research	27	69	require	11
33	discussion	26	70	register	11
34	Question	25	71	approval	11
35	background	23	72	website	10
36	presentation	23	73	proposal	9
37	Solution	23	74	tutorial	8

3.4.2. New Raking Method

In addition to the detected words and their timings the Word Bubble interface requires each word to be ranked for its ability to predict content. More predictive words are displayed higher up in the list and with a redder bubble. We developed a new raking method that takes advantage of Google search results. We made the assumption that words that returned fewer results on a Google search were more predictive of content than words that produced more results. Equation 1 shows the conversion from number of Google results for a phrase and its content predictiveness value (CPV). The variable *n* is the number of results returned for the current word. The variables *min* and *max* represent the minimum and maximum number of results returned for all words in the given video respectively. The constant value 15 was introduced into the equation to scale the output into the appropriate range required by the word bubble interface

$$CPV = 15e^{-15\times(n\text{-min})/(\text{max-min})}$$

Equation 1: equations for converting number of returned Google results into content predictiveness value

3.4.3. Zipf's Law Filtering

The intention of the Zipf's law filtering approach was to eliminate high frequency words from the Word Bubble display that cluttered the display and made it hard to pick out content words that accurately describe the changing nature of lectures content. For example, a course lecture from a computer science class the phrase "Computer Science" may appear many times. This data is useful for identification of the general topic of the lecture, useful in summarization,

but is distracting when trying to locate a section internal to the lecture. Similarly even more specific terms may be used repeatedly during the lecture and would indicate the general topic of the lecture, but not the content of sublecture segments.

We then set a maximum count threshold by subtracting one standard deviation from the maximum count; all words with counts greater than this threshold were not included in the final word list. Trimming the bottom end of the word list was not required due to the filtering done by the course text index. This filtering removed many common non-content words and left only course related words. See Figure 7 for a plot of the log of word frequencies versus log of word rank. The graph is somewhat linear, which is in accordance with Zipf's law.

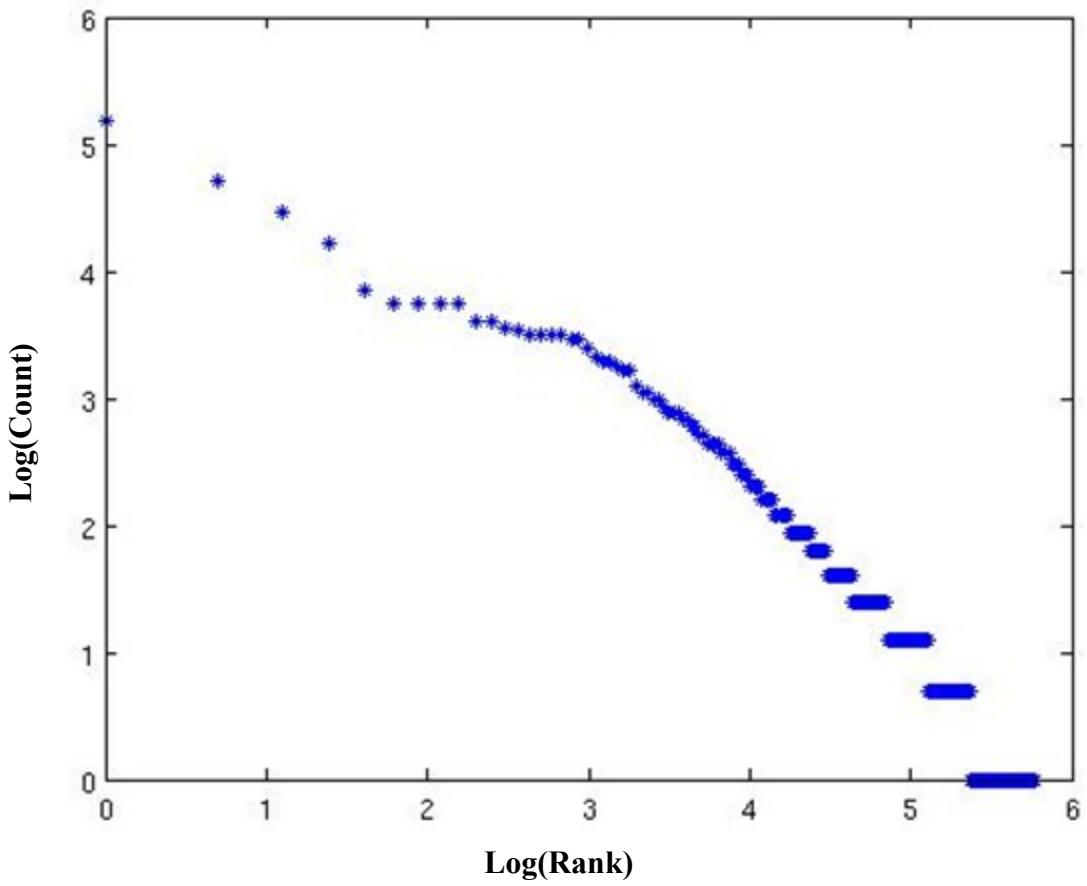

Figure 7: plot of word frequencies for one sample lecture from the data set on a log-log scale.

3.4.4. Word Tags

Due to user feedback gathered during user studies, a recurring theme emerged that the way to increase usefulness of the Word Bubbles for searching the video was by adding a method of searching the Word Bubbles. Instead, we decided to unify the interfaces and take the word and timing data used to generate the Word Bubbles and turn those into tags that operated the same way as the Visual Tags. We remolded the data so that any video segment where a word was spoken was tagged with that word as a new tag. The user implications of this change are reported in Chapter 5.

4. Feature Selection

The Sort-Merge algorithm suggests stopping criteria such as running the algorithm until a set of desired cardinality is generated or until a performance threshold is generated [3]. We explore Fast Cutoff, an improvement to the stopping condition. We produced a tool, shown in Figure 13 the Sort-Merge Genealogy Tool, to analyze the change in feature sets over the levels of the Sort-Merge tree.

The use of this tool exposed the observation that merging two feature sets had one of three outcomes. The merger could produce a feature set that performed better than both parents, a feature set that performed worse than both parents, or a feature set that performed better than one parent and worse than the other parent. It was observed that a Cutoff Level existed: there is one level of the Sort-Merge tree where all mergers produced feature sets that performed worse than both parents. It was also observed that all subsequent levels of the Sort-Merge Tree produced lesser performing feature sets as well. The feature set identified as best keeper always came from the Cutoff Level or a level previous to it. This is useful because the remaining levels of the Sort-Merge tree, which are far more expensive, can be ignored, thus increasing the speed of the selection process.

4.1. Sort-Merge

4.1.1. Implementation

We based our feature selection algorithm on the Sort-Merge algorithm presented in [3]. We first implemented the Sort-Merge algorithm essentially as outlined in that paper with some considerations for our domain. A pre-filtering step to filter out poorer features is not applicable to the Sort-Merge algorithm. The algorithm requires the features set to be large and redundant thus filtering out redundant or similar features would harm the algorithms performance. Additionally in the experiments described below, it is shown that features that are on their own poor performers, can become good performers when paired with other features. Thus filtering out poor single features would also harm the algorithms performance.

4.1.1.1. Method for Choosing Positive & Negative Examples

An important consideration for a machine-learning problem is the choice of positive and negative examples. We generated our own groundtruths using the custom-made tool described in section 3.1.1.

We made use of training example subsampling to reduce the size of data set we were working with. While randomly choosing the training examples, the proportion of positive to negative examples is maintained. The set is then partitioned into a testing set, and a validation set. The testing set is to be used during the Sort-Merge algorithm. The validation set is used to assess the performance of the feature sets selected by the Sort-Merge algorithm. 80% of the

chosen examples are used for the testing set, and the remaining 20% are used for the validation set; see Figure 8.

4.1.1.2. 3-Fold, 5-Fold & 10-Fold

In order to smooth out statistical variability in the learning process, each trial in the Sort-Merge tree uses k-fold cross validation. The testing set is partitioned into *k* sets. *K* trials are run, using one partition as the training data, and the remaining partitions as the testing data. The results from all runs are then averaged together to generate the final performance measure. In the standard implementation of x-fold cross validation, one fold is held in reserve for validating, while the rest of the data is used to build the classifier. However, due to the size of our data set, proceeding in this manner is prohibitively time consuming. Therefore we use the variation of a single fold for training because of time concerns, as the volume of data we are dealing with is large.

In our experiments, we tried 3-fold, 5-fold and 10-fold validation. We settled on using 3-fold because it provided a good tradeoff between time and reduction in variability of classification results.

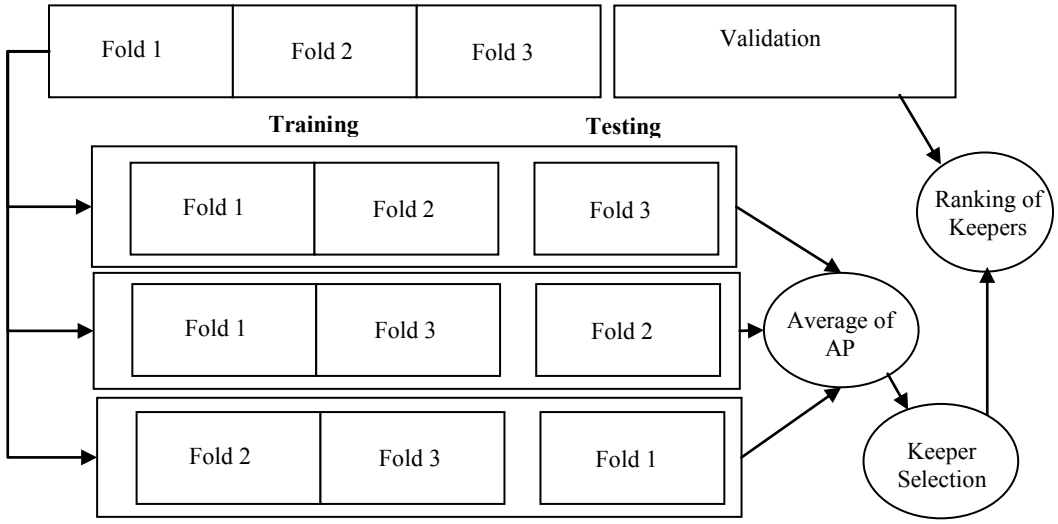

Figure 8: Diagram of how the chosen examples are split into testing and validation sets. The example shows 3-fold validation.

4.1.2. SVM Tuning

In our experiments we used an SVM with a Radial Basis Function (RBF) kernel. An RBF is a function whose value depends only on the distance from the origin. This kernel takes two parameters, gamma and C. Gamma is a scaling factor for the distance measure used by the SVM and C is a penalty factor used when a perfect separation cannot be found. Varying the parameters can have an appreciable effect on the performance of an SVM. We have performed a grid search over the gamma-C space to find a pairing of parameters which yield increased performance. See Figure 9. Due to this examination, we decided to set both gamma and C to 0. The Results show that as long as we are not in the though we will be acceptable.

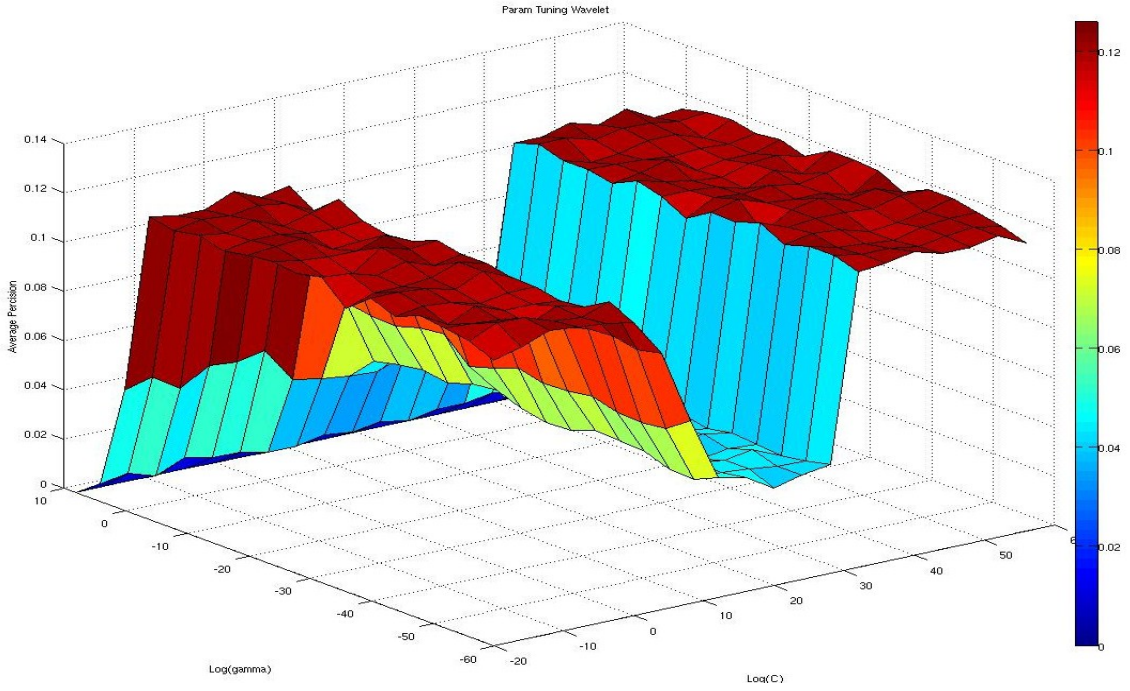

Figure 9: SVM tuning results

4.1.3. Data Set

We use a data set taken from the Columbia Video Network. These videos consist of course lectures, and student midterm and final exam presentations. We used 169 videos consisting of 172.8 hours. A segmentation algorithm developed by the VASTMM browser, set to maximum granularity, generated 66130 virtual key frames from these videos. Because we used 3-fold validation each fold therefore used 441 positive and 2864 negative examples in training the classifier; 6610 images in testing (to determine the ordering in the sort step of the Sort-Merge); and 2480 images in validation (to determine keepers at the end of the Sort-Merge algorithm).

4.1.4. Feature Correlation Coefficients

We have attempted to determine if there was any pattern in the selection of features by Sort Merge. We developed a visual tool to display which features have been selected, see Figure 14 (described later). At least for the Wavelet feature, it was unclear from the visualization if any pattern emerged, such as a preference for one side of the image, or a preference for one band of the wavelet. We also explored if there was a statistical correlation between the features that were selected. However, there was no significant correlation in any of the dimensions of the features, indicating that the features selected by Sort-Merge were not redundant.

4.2. Keepers Heuristic

The number of possible low-level image features is large, but few are relevant to a particular semantic tag. For example, our wavelet texture measure generates 567 features per image, of which only about 6% appear to be useful for detecting "code" key frames. The Sort-Merge feature selection algorithm works well on sets of such magnitude, but does not define a definite stopping condition, leaving that to the experimenter. We improve upon this method with the *Keeper Heuristic*.

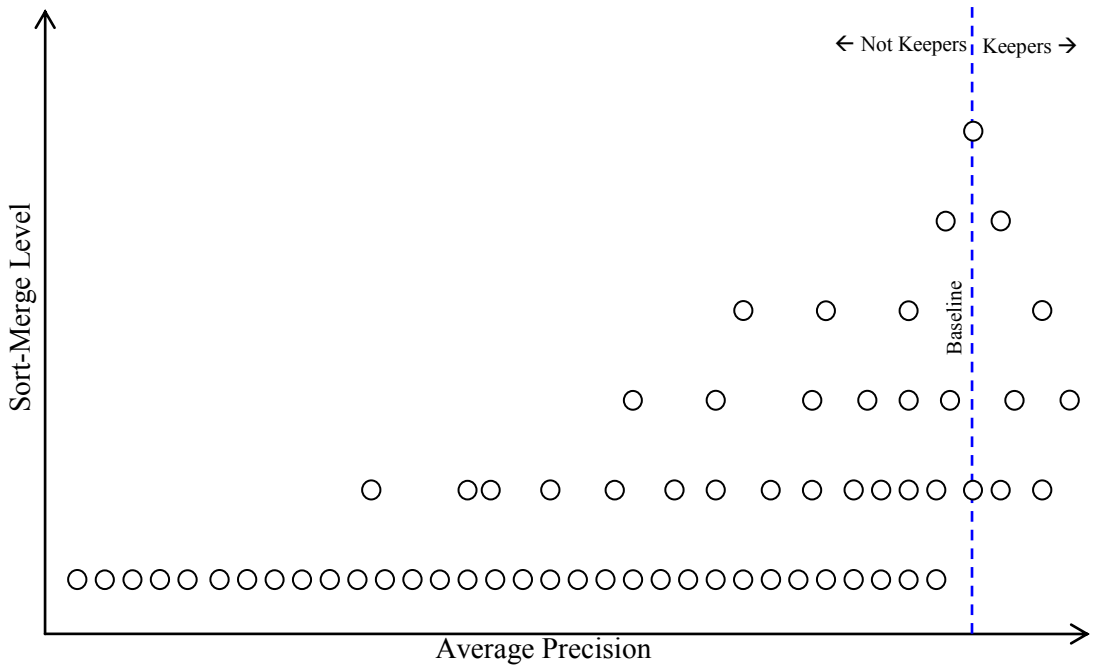

Figure 10: A mocked up Average Precision vs. Sort-Merge level graph. The blue dashed line represents the baseline Average Precision. The baseline is the Average Precision of all the features used together. Circles represent feature sets. Sets to the right of the line perform better than the baseline and are thus Keepers.

This is a heuristic method, visualized in Figure 10, of selecting good feature sets based on their comparison to a baseline of the classification performance using the entire set of features. Since the entire set is usually redundant, its absolute performance is actually suboptimal. At each level of the Sort-Merge tree, any feature set of equal or greater performance than this baseline is saved as a "keeper". When the Sort-Merge tree is completed, only these keepers (which usually vary in feature set cardinality) are tested against a validation set of images. The best performing one, regardless of feature set cardinality, is considered the best keeper.

4.3. Fast Keeper Cutoff Level

Because Sort-Merge exploits the heuristic that large volumes of features are redundant, the Average Precision of the merged pair cannot be predicted. The resulting Average Precision can be greater than, less than, or in between the separate Average Precisions of the parents. We developed a tool to visualize the average precisions across the levels of the sort merge tree. Greater detail on this tool is given i\n section 4.9.1

However, we have noticed that after a certain level, every pairing of features has a lower AP than both of its parents, probably due to over fitting. Additionally, we noticed the best keeper has always come from this level or a previous level. This criterion can be used as a much cheaper stopping point for the Sort-Merge algorithm. This obviates the costly need to train an SVM on the full feature set and use this classifier's performance to establish a performance baseline, and allows rapid exit from the algorithm without the costly training of SVMs at the highest levels, where feature sets are largest. That is, we gain the benefits of selecting highly performing subsets without needing to actually compute the baseline itself.

4.4. Application to Very Large Feature Sets

Sort-Merge itself is already designed to handle very large feature sets. The assumption made by Sort-Merge is that in a very large feature set, many features will be redundant. When Sort-Merge groups features together, it may create a poorly performing feature set. This set will be discarded, but due to the redundancy, it is rarely a loss, as another feature set will very often operate better than the lost features. But in addition to these features of the basic Sort-Merge

algorithm which ensures accuracy, we have included the keeper heuristic and fast feature cutoff to help handle the very large feature set we are selecting from which enhances speed.

4.5. Early Fusion & Late Fusion

Feature fusion is the method of combining multiple different features, for example, color features and texture features, in the feature selection process. However, Sort-Merge was designed to only use homogeneous feature sets. We enhance the Sort-Merge algorithm to deal with heterogeneous feature sets.

But first we define how relative performances among the different features are to be weighted. We explore both early fusion, which combines all the features from the different feature sets equally from the beginning and then Sort-Merges this combined set, and also late fusion, which first selects the keepers from each feature type separately (where performance measures are fairer) and then Sort-Merges these smaller sets. Early fusion performs better that late fusion. However, late fusion is faster.

4.6. Cascaded Classifiers

We used cascaded classifiers to enhance the performance of our SVMs. We noticed that using the wavelet feature, it was easier to discriminate “code” when only considering images that had text in them, than it was to discriminate “code” when considering all images. This works because of the parent-child relationship of the tags “text” and “code”. During our experiments with the “code” tag, we only considered images that had a groundtruth positive for “text”.

The performance of the best keepers for each feature class can be found in Figure 11(a) where Average Precision (AP) is the metric for feature set performance. Identifying frames with readable text ("text") using an SVM trained with features selected from wavelet transforms proved to be easy, resulting in a perfect AP of 1 for the best keeper (of size 32). Our initial attempt to identify frames that contained some form of computer code ("code": assembly, Java, Matlab, C, others) using selected wavelet features resulted in an AP of 0.667 for the best keeper (of size 32).

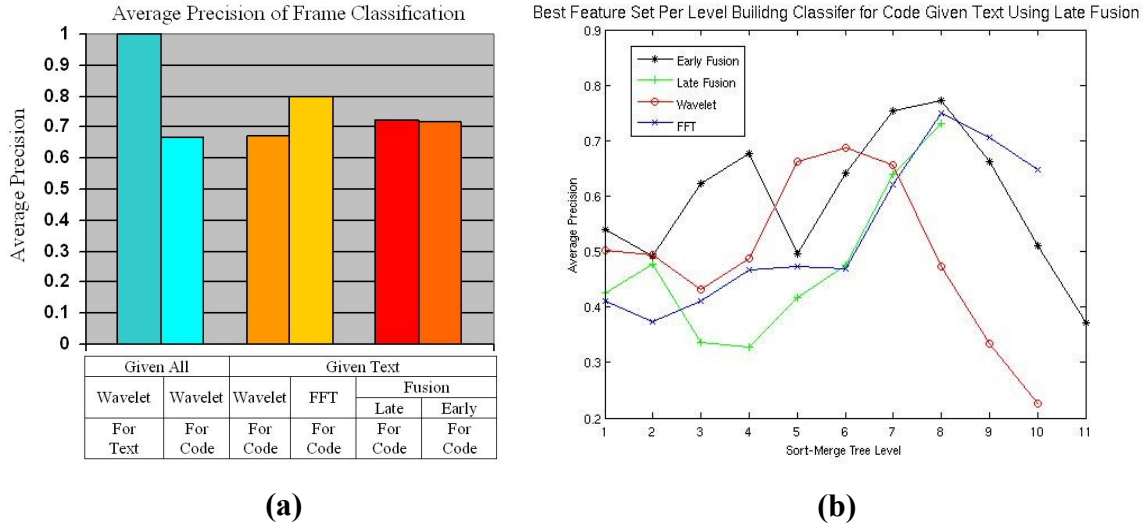

Figure 11: (a) Average Precision of Fusion Experiments (b) Average precision at each Sort-Merge level

Identifying code was more accurate if the SVMs were cascaded and "code" was sought only within the context of "text" frames, discriminating non-code text from code text. See Figure 11(a), where "for code given text" refers to this cascaded method; performance increased to an AP of 0.79.

4.7. Fusion Experiments

Given the heuristic nature of the Sort-Merge method, we needed to verify that no significant interactions of the two different feature types were overlooked. We conducted two fusion experiments: early and late. Figure 11(a) shows the results. Late fusion is not only considerably faster (96 hrs. vs. 116 hrs. running on a Dual Core 1.3 GHz Intel Chip running Microsoft Windows XP service pack 3) than early fusion but yields a higher AP. However, FFT alone performs the best; attempting to fuse a weaker feature with a strong one appears only to have weakened the better of the two.

Figure 11(b) shows the AP performance, on test data, of the best feature set at each level of the Sort-Merge tree. The graph shows that at a certain point adding more features is detrimental to classification, probably due to overfitting. (Although the peak of the early fusion curve here is greater than that of the FFT curve – the opposite of what is shown in Figure 11(a) – this is because Figure 11(b) displays test set rather than validation set results.)

4.8. Performance with Respect to Random

One way of evaluating the performance of a machine learning implementation is to compare statistical measures. We decided to use two different measures, Average Precision and Precision at 100. We decided to use Average Precision because it heavily favors correct results at the top of a ranked list. Precision at 100 is a similar measure, but does not weigh correct results higher up in the list.

We evaluate feature selection by comparing the performance of a number of random feature sets to these feature sets chosen by our feature selection algorithm. In Figure 12 we compare the AP of 100 random FFT feature sets to the results of our modified Sort-Merge algorithm. The mean and standard deviation of the random trials is show on the graph as colored lines. The performance of a Sort-Merge selected feature set is greater than two standard deviations above the mean. We used this method because there were no other reasonably alternatives. Other alternatives, such as comparing all possible feature sets would be too costly to use effectively, and standard feature selection methods take literally weeks to execute.

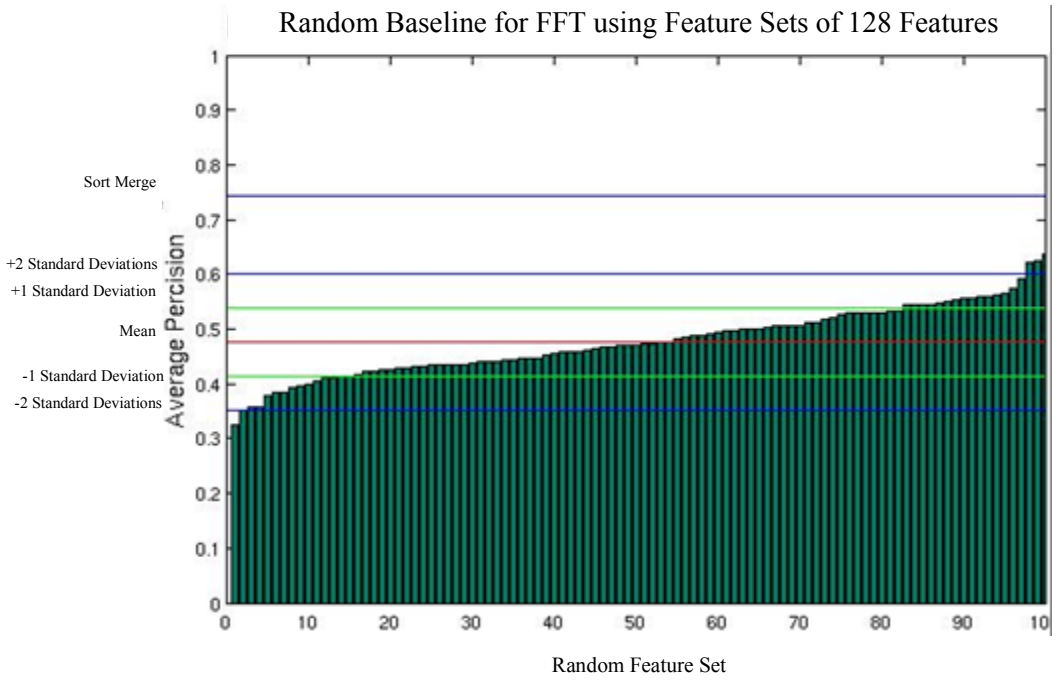

Figure 12: Comparison of AP of 100 random feature sets to AP of sort merge. The data set was "Code", using the FFT feature where gamma = 0 and C = 0.

4.9. Visualization Tools

4.9.1. Sort-Merge Genealogy Tool

In order to better understand the evolution of keepers, we designed a tool to trace the genealogy of a feature set; see Figure 13. In this tool, each box represents a feature or feature set that is operated on by sort merge. This tool shows which two feature sets were merged to produce a given set, and how well the combination preformed. The tool is interactive, allowing the trace of individual features backwards and forwards through the Sort-Merge levels.

These boxes are grouped together by the Sort-Merge level of which they are a part and are sorted by performance with the highest performing to the left. Red dots graph by their

vertical height the AP of the feature sets, and blue dots graph the AP of the merger of each pair of adjacent sets. When a feature set box is clicked on it will be highlighted in green. This will also show the genealogy trace of that feature set. Gray liens are draw between this selected feature set, the two feature sets that were merged to generate it, and the feature set that it is used as a component for merging. These gray lines are then recursively drawn up and down the graph, showing where all the components of the selected feature set came from and where the selected feature set ends up. The yellow arrow shows the Sort-Merge level from which the best keeper comes from. All feature sets after this lever perform worse than the level before it.

This tool visualizes the effect that merging two feature sets has on performance. The gray lines are helpful in picking out this relationship. The observations taken from this tool led directly to the Fast Keeper Cutoff method. Figure 16 shows one example of a Sort-Merge run. The yellow arrow shows the Sort-Merge level where the best keeper came from. It can be seen by comparing the blue and red dots at this level that the mergers at this level all produce feature sets that have performance no better that their parents. This same fact holds true of all subsequent levels. This is then a critical level, the one where merging any two sets together does not result in increased performance. From this evidence it is clear that searching for a keeper past this critical level is not needed. Using this level as a cutoff allows computational saving in the higher levels of the tree where each evaluation step is more costly, while still being confident in finding the best keeper.

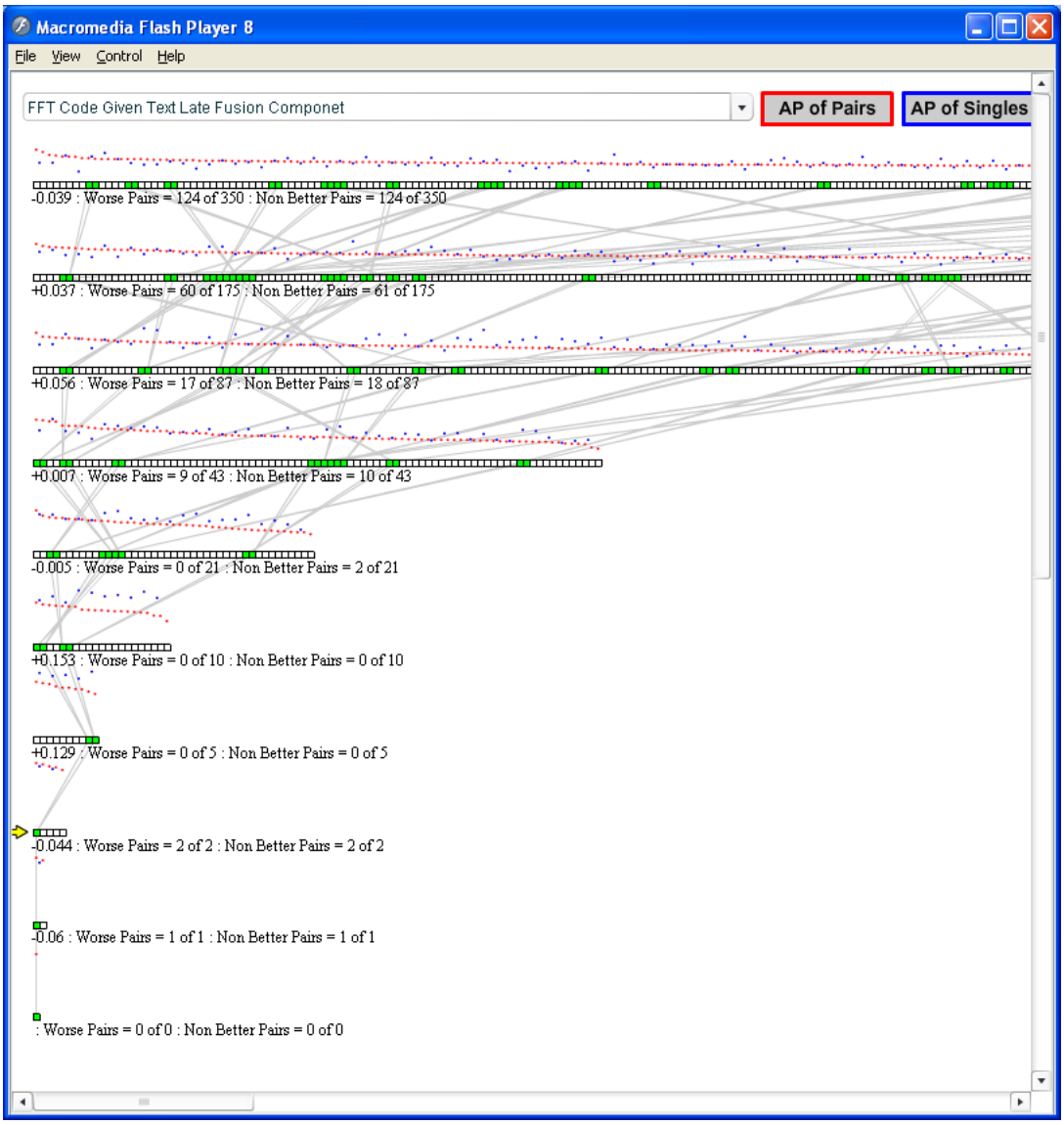

Figure 13: Screenshot of the Sort-Merge Genealogy Tracer showing the trace of the best keeper.

4.9.2. Feature Visualization Tools

Because of the overwhelming complexity of the new feature set, we developed a number of tools for feature visualization. One for FTT features, one for Wavelet features, and one for Color Coherence Vectors. These tools were created to examine which features were chosen as

the best feature set, and to look for any patterns of feature selection. We also created a tool to visualize the results of an individual sort merge run called the sort merge genealogy tool.

4.9.2.1. FFT Feature Visualizer

To create a visualization for FFT features, a feature set image was created, where the pixel corresponding to one element of the feature set was set to completely white. An inverse FFT was then calculated for this image. Feature set images and their inverse FFTs were calculated for each element in the feature vector. The inverse FFT images were then superimposed upon each other to form the visualization of the frequencies chosen. Figure 14 shows for feature sets of varying size, an image representing the vector components chosen, the visualization of chosen frequencies, and the visualization superimposed on an example image. We developed this tool in order to better understand which features were being selected by the Sort-Merge algorithm. We wanted to determined what pattern the algorithm was picking out of the data. Specifically we hypothesized that code examples on slides would be recognizable by their fixed width font. The best keeper selected by the algorithm is the 128 feature set shown in Figure 14. This visualization confirmed our suspicions by exhibiting a regularly spaced pattern in both the horizontal and vertical directions.

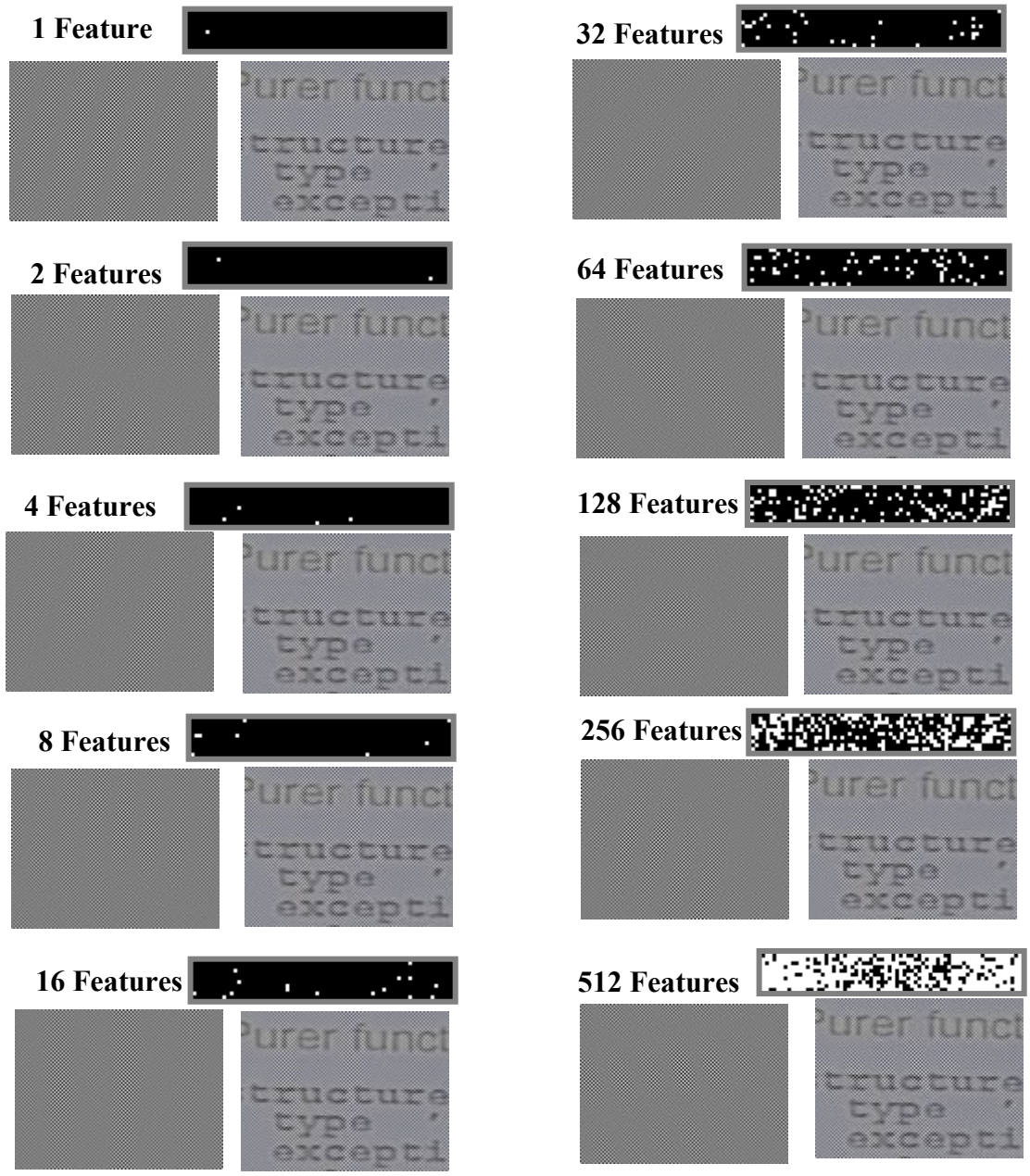

Figure 14: An example of feature sets at each level in the Sort-Merge tree while using the FFT feature to classify code given text. Each grouping chosen was the best performing feature set of that level (not necessarily a keeper). Each feature set is displayed in the frequency domain (scaled up 2x for clarity), as a spatial representation, and as that representation superimposed over an example image from the dataset.

4.9.2.2. Wavelet Feature Visualizer

To create the visualization for wavelet features, we created a grid image to represent the 4x4 grid that the wavelets were calculated in. In addition, each grid section was separated into 3 sections to represent the detail levels at which the wavelet was calculated, and three of the four sub-bands (HL,LH & HH). A separate grid was made to represent the same layout for the LL sub-band. Each one of these sections was colored in to represent which moment was part of the feature set. See Figure 15. Upon examining this visualizer output, we noticed that there was no special location to the text data. It could appear at any section of the image. Also there was no preference for the level of detail needed to detect the text.

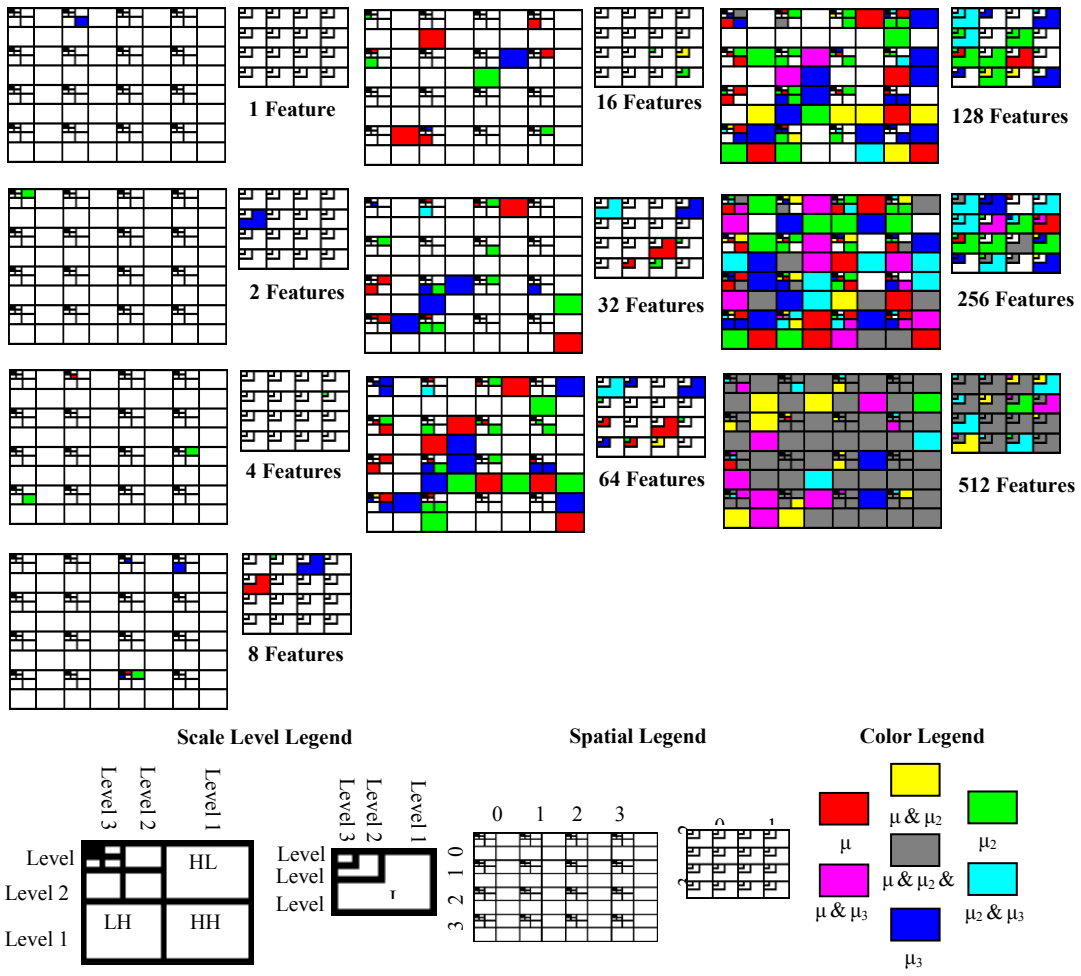

Figure 15: Best feature set at each level for the wavelet feature while classifying code given text. Each grouping chosen was the best performing feature set of that level. The features chosen are as indicated by the legends at the bottom of the figure. μ represents the mean and μn represents the nth moment about the mean of the pixel intensities of the resultant wavelet.

4.9.2.3. Color Coherence Vector Visualizer

To create the visualization for the Color Coherence Vector we created 16 3D plots, one for each section of the image the feature was calculated on. The left hand side of Figure 16 shows what a plot of the full feature set would look like. Each line plotted on that graph

represents a bucket from the color histogram created from the coherent regions. We take advantage of the fact that each bucket corresponds to a color. The color of the line is a representative color from the corresponding bucket. The feature selection algorithm chooses only a subset of the buckets. For each region of the image we draw only the lines that represent he buckets that were chosen by the algorithm. This visualization showed us what the feature selection process was using to make its classification. From Figure 16 we determined that a combination of reds and greens being selected for. Coherent region of these colors correspond to the redness of skin tone (presenters faces) and the backgrounds on which they appeared.

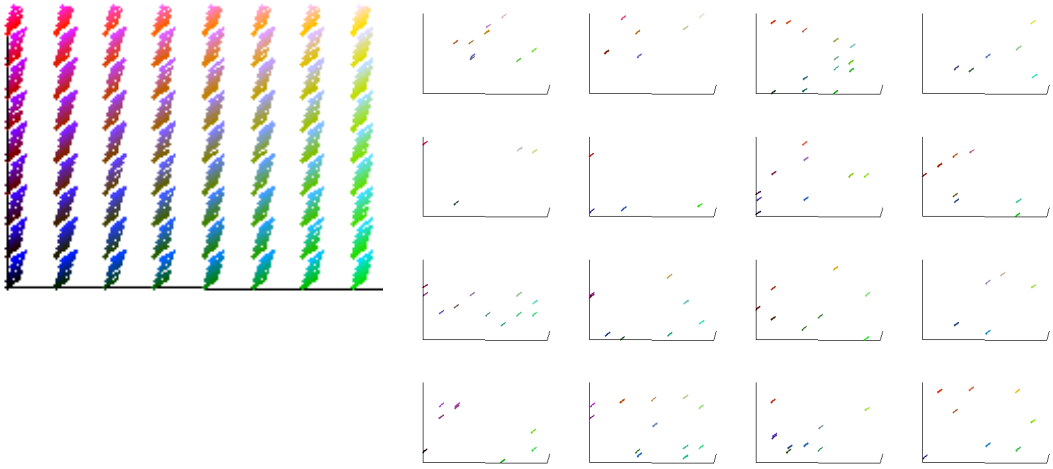

Figure 16: An example of a feature set of cordiality 128 selected by Sort-Merge as a keeper while using CCV to classify "presenter". Each set is displayed as a series of 3D plot of selected color buckets from the coherent histogram. Each plot represents one region of the image in the 4x4 grid. Each selected feature is represented as a needle colored with its bucket's representative color. On the left hand side is a graph showing all buckets plotted together for reference.

4.9.2.4. Outcome

The examination of the results of the visualizers was a key component in testing the keeper heuristic. Besides the positive comparison to baseline performance, it was important to understand what the final criterion was that was being selected for by Sort-Merge in relation to the keeper heuristic. The qualitative match between what the visualizers were showing us, and the known facts about the composition of the dataset allowed us to trust that the keeper heuristic was making correct choices.

5. VastMM-Tag Performance

5.1. The VastMM-Tag User Interface

The user interface in based on the user interface from the VASTMM browser. Inherited from that browser are a number of components, the Keyframe Array, the Keyframe inlay, the Word Bubbles and the streaming media player [18]. Added to this to create VastMM-Tag are the Tag Timelines and the tag selection mechanisms. A screenshot of the VastMM-Tag browser is provided in Figure 17.

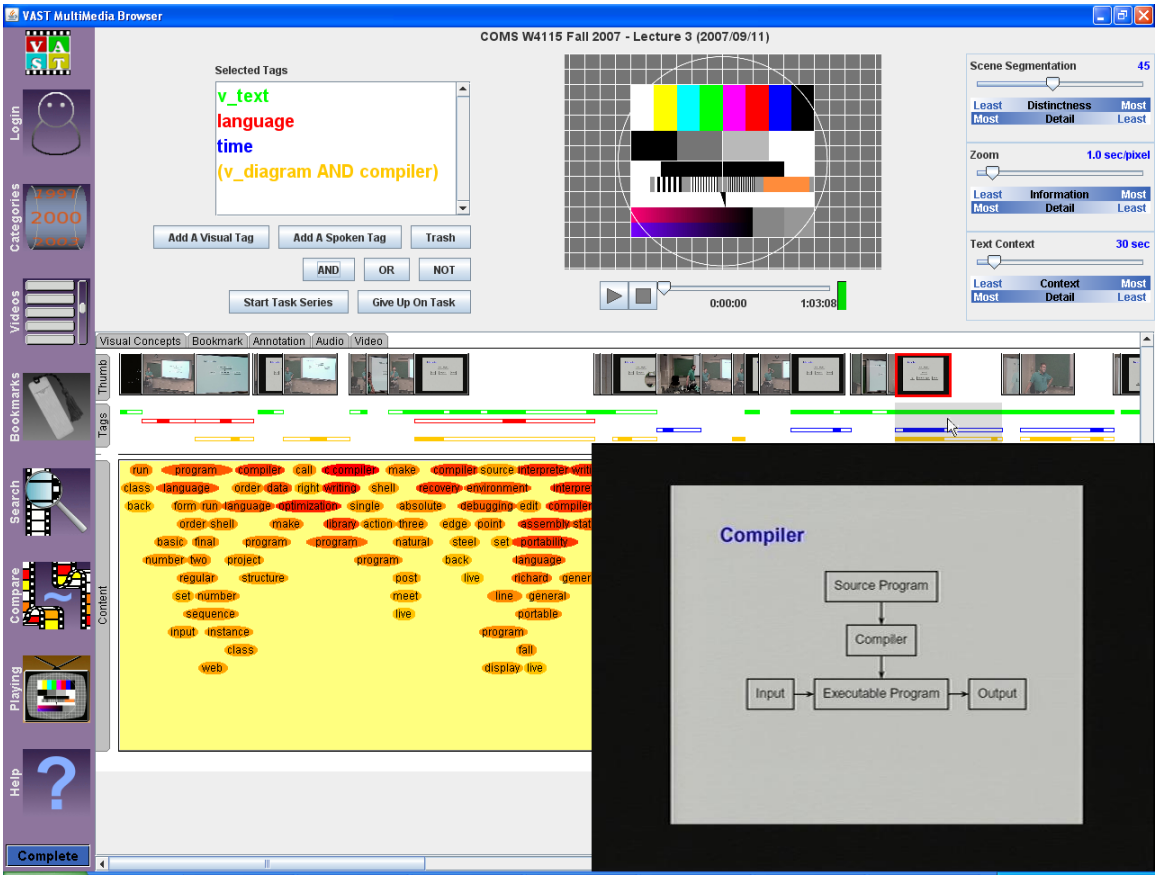

Figure 17: Screenshot of VastMMTag in action.

5.1.1. Elements Inherited from VASTMM

The Keyframe Array is a display element that shows selected keyframes from the video along a timeline. To select these keyframes, a segmentation algorithm decomposes the video into simulated shots based on filters designed to respond to fast shot transition (hard cuts) and gradual shot transition (dissolves, fades, wipes, etc.) [18]. It is unclear what response level should generate the "best" boundaries, so a scene segmentation slider is available to the user to select how many shots to show. When a shot is moused over, the user is presented with an enlarged inlay of the keyframe. Figure 18 shows a screenshot of the VastMM-Tag browser with both the Keyframe Array and the inlay highlighted.

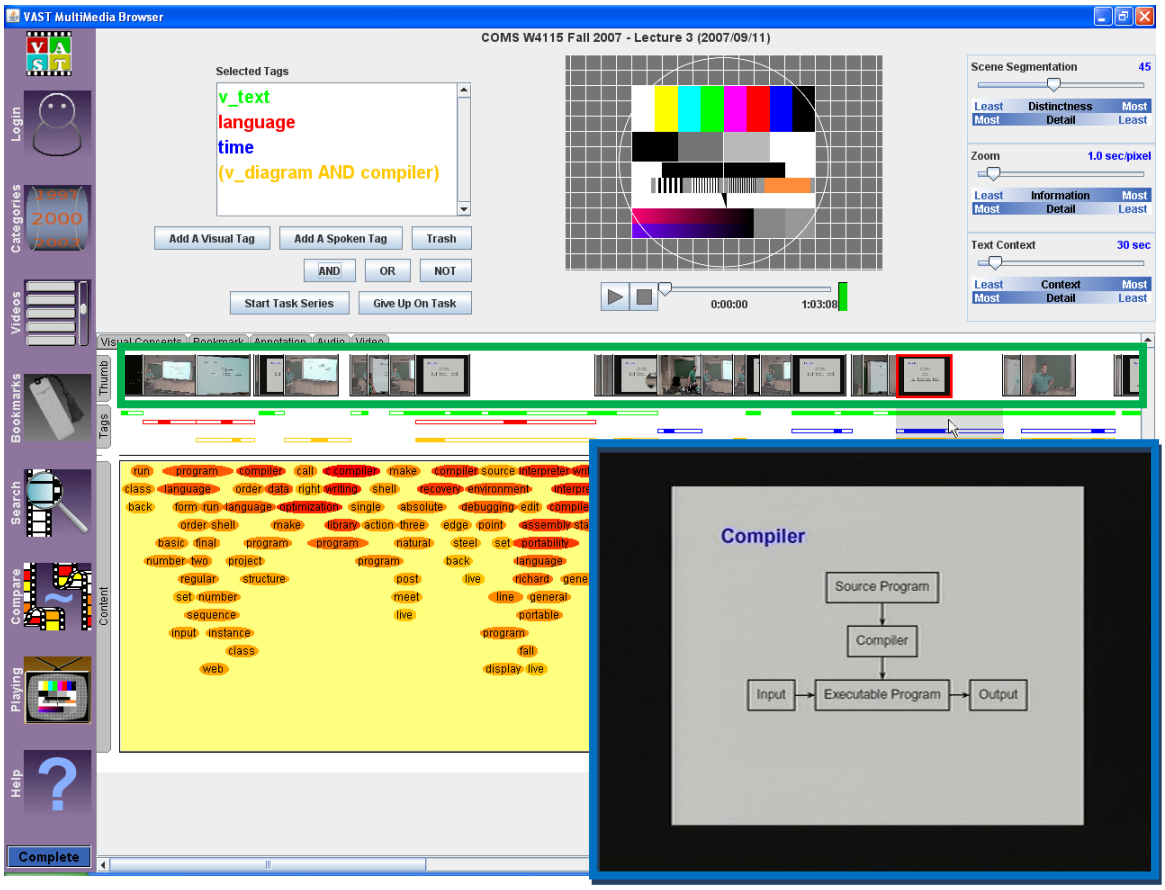

Figure 18: Screenshot of VastMMTag with the Keyframe timeline highlighted in green and the Keyframe inlay highlighted in blue.

The Word Bubbles are a way to try and capture the overall audio content of the lecture into a humanly scalable visual representation. The words are captured by automatic speech recognition that is informed by the words appearing in the index of the lecture text book. Each word that is recognized is associated with a timestamp and a content predictiveness value, see 3.4.2. The bubbles are then drawn based on these values. Words spoken frequently and close together in time are drawn with larger bubbles. Words that have a higher uniqueness value show up redder and higher up on the chart. See Figure 19 [18].

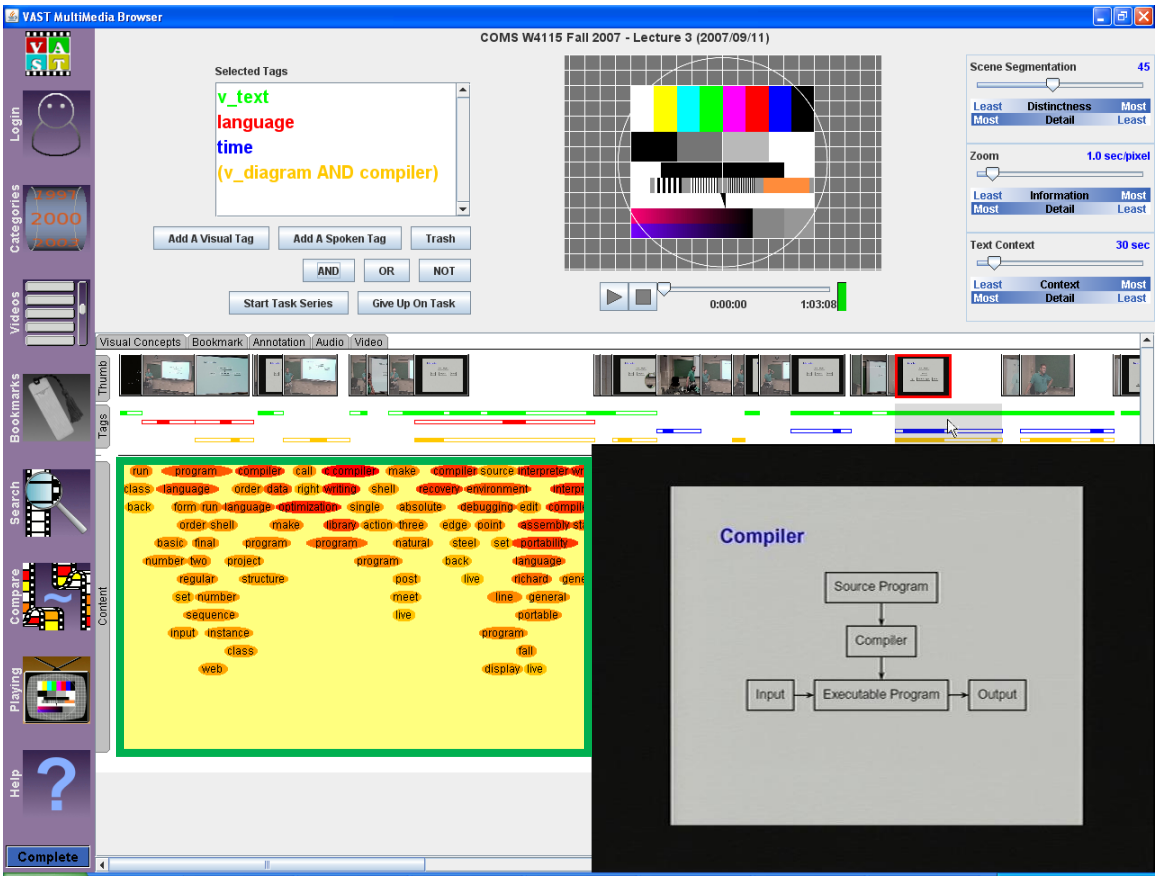

Figure 19: VastMM-Tag screenshot with Word Bubbles highlighted in green.

The media player area is purely a standard media player as seen in many commercially available products. This player is a streaming MPEG1 player that allows play/pause, stop and the ability to skip to a given time in the video by clicking on the player timeline. Figure 20 shows a screenshot of the VastMM-Tag browser with the media player window highlighted.

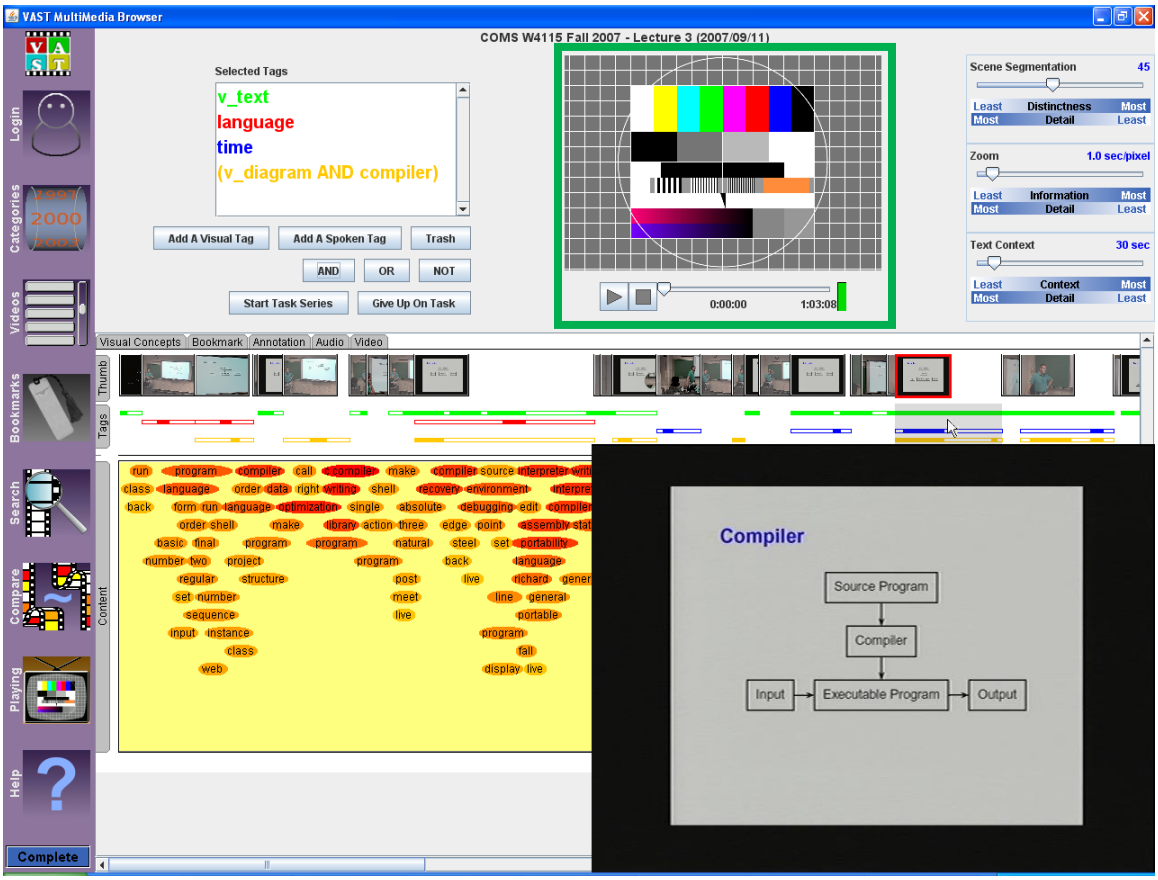

Figure 20: Screenshot of VastMMTag with the player window highlighted in green.

5.1.2. Novel Interface Elements Added by This Work

The tag selection area is used for the selection of tags for the creation of timelines that are then displayed in the multi-timeline area. This area provides a method for selection of tags from the complete set of tags recognized in the video. After experimenting with selection methods and user feedback, we used tag selection from a flat list for Visual Tags, and an auto-completing drop-down for Word Tags. The reason for the difference in selection is the difference in the size of the tag sets: Visual Tags having ~4 and Word Tags having ~400. This section of the interface is highlighted in red in Figure 21. The VASTMM browser has no functionally to display any

such timelines or to display visually derived semantic information other than recognized faces [18]. The MSNBC debate viewer used a similar multi-timeline display [19] however the timelines were based on a human's determination of the current topic and current speaker. Additionally one was limited to a fixed number of timelines and no Boolean combinations of timelines.

Once selection is made, the selected tag is added to a "selected tags" display, where the tag is color-coded based on the order in which it appears in the display window. At the same time, a timeline is added to the multi-timeline view with a corresponding ordering and color coding. The interface allows reordering of the selected tags, by using drag and drop, in the selected tags display. This display correspondingly changes the color coding of the tag in the display as well as the resultant timeline in the multi-timeline area. The tags are colored by the order in which they appear in the Selected Tags box. The colors, in order, are: green, red, blue, orange, and magenta based on experimentation with varying color schemes. Any tags selected beyond the five are colored black. The ability to remove tags from the selected tag display is also provided. Any change in ordering resulting from a removal will also result in a recoloring as described previously.

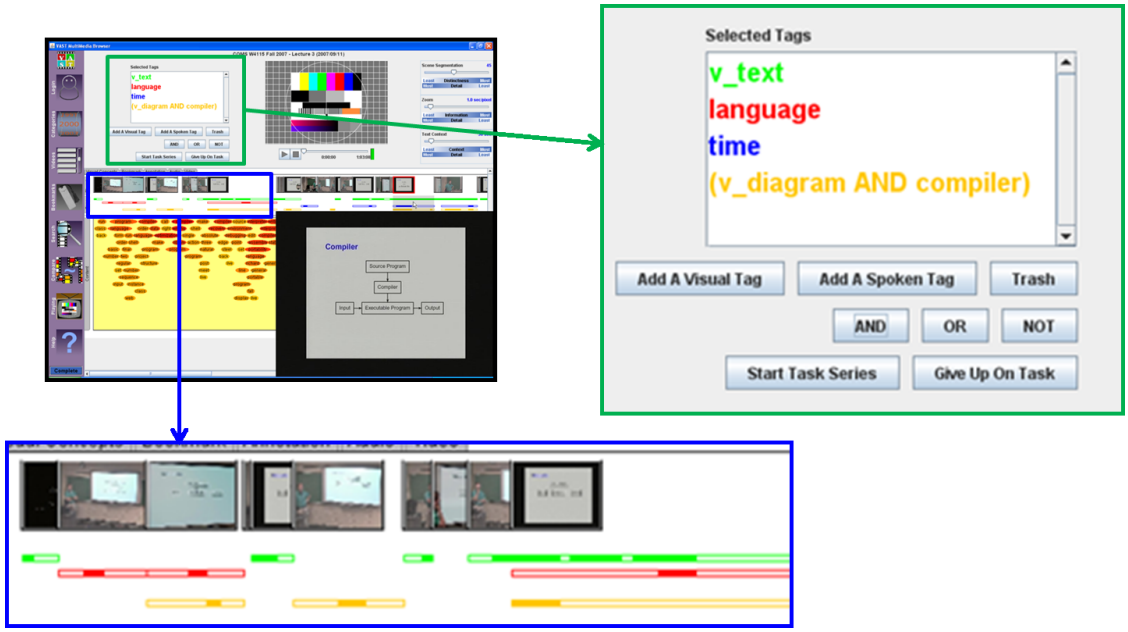

Figure 21: Screenshot of VastMMTag with the selected Tags Window highlighted in green and the Tag Timelines highlighted in blue.

This area also presents an interface for performing boolean operations upon one or more of the tags to create a new timeline. A Boolean operation is carried out by first selecting the operands in the selected tag display and then selecting the operator to perform on those operands. The resultant of the operation is then added to the selected tag display as a newly created tag and corresponding timeline, while the operand tags and corresponding timelines are removed from the selected tag display. These created timelines are then treated the same as all other timelines with respect to reordering, removing, and use as operands in boolean operations.

The multi-timeline area is used for the display of timelines to indicate presence, absence, and uncertainty of presence or absence of the tags that are selected or created through boolean operations in the tag selection area. The indication is done though drawing either filled or hollow bars. When the multi-timeline area is moused over as in Figure 22 the corresponding

keyframe is displayed just like mousing over the keyframe in the Keyframe Array. When the timeline area is clicked, the video jumps to the time in the media where the click occurred.

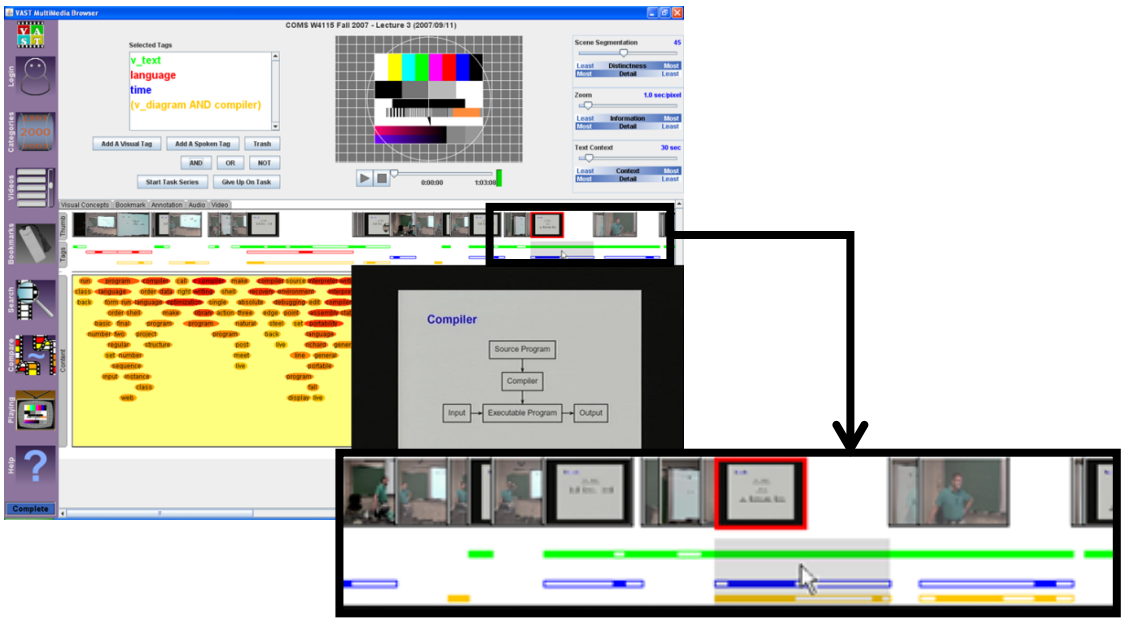

Figure 22: A close up view of the mouse over process. The frame highlighted in red is the same frame as in the large inlay

5.1.3. Interpreting Timelines

For each selected tag, a correspondingly colored timeline is displayed. Within it, each video is segmented into a series of automatically detected 'shots'. The shot boundaries are determined by an algorithm that detects both abrupt changes in intensity and more gradual changes. This algorithm also produces a confidence value for the boundary. The user interface allows the user to select the 'scene segmentation level'; which thresholds boundary confidences. When the scene segmentation level is set to its most permissive (see top of Figure 23), then for each tag, a filled colored bar and a representative thumbnail is drawn for each segment. Typically this results in many shots with solid bars. The length of the bar reflects the length of

the video segment that carries that semantic concept. The hollow bars seen in Figure 23 are drawn to indicate uncertain tag presence, resulting from a heuristic merger of video segments when the cut confidence threshold is not at its most permissive. As the cut confidence threshold has been changed to be less permissive, the number of segments in the video decreases but the locality of the tag decreases also. Typically this results in fewer longer hollow bars.

The method we use for computing inaccuracy is heuristic. In a more exact approach, these newer longer segments should be processed anew to find representative keyframes and to confirm tag presence. However, such an approach would be too computationally expensive. Rather, we make the assumption that segments that no longer are to be displayed separately should be grouped together semantically with the segment that precedes it. This can result in an arbitrary number of segments being grouped together into one larger segment. We then also assume that the thumbnail of the chronologically first segment in this larger segment, which is the only segment whose cut confidence still exceeds the segmentation threshold, can be used as the representative thumbnail for the new larger segment.

To generate tags for these merged segments, we use the union of the tags of the component segments, using the rationale that if a base segment contains a certain concept the supersegment must also contain that concept. However, hollow bars are draw to indicate the uncertainty of their tag locations resulting from this heuristic approach to segment merger, as can be seen in the bottom two shots in Figure 23.

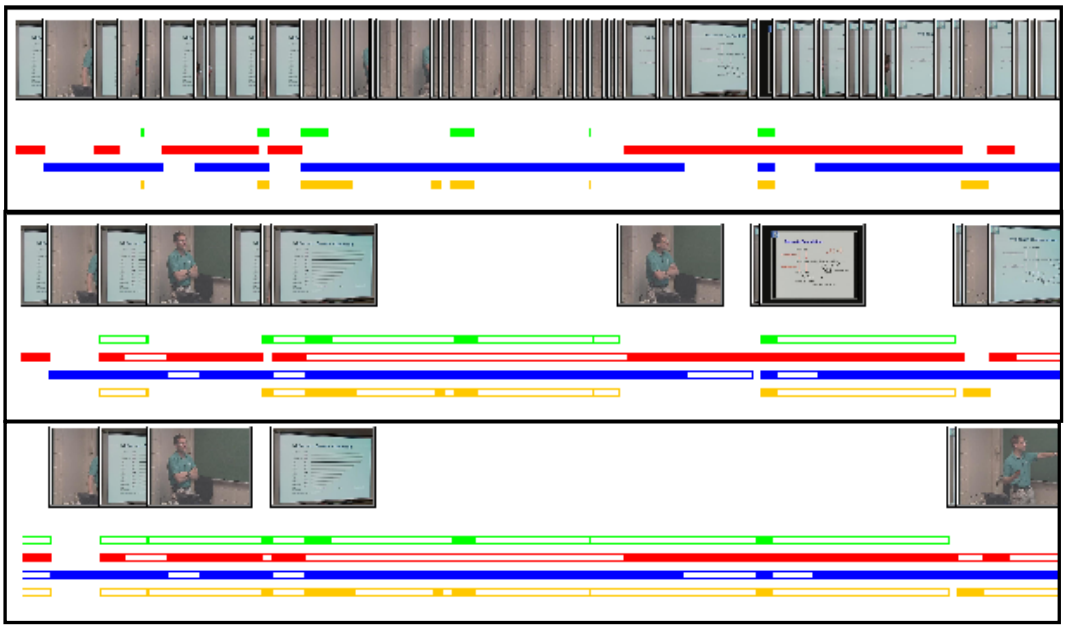

Figure 23: A close up of the timeline section of the browser at varying scene segmentation levels. From top to bottom, the shots show cut confidence 0 (most permissive), 30, and 100 (least permissive).

We found that during search tasks users did adjust the scene segmentation level, using it as a kind of semantic zoom. After locating a section of the video they are looking for, users would sometimes change the segmentation level again to show smaller segments, in order to more finely localize the segment they wished to view.

5.1.4. Using the Tag Algebra

A Tag Algebra is available to enable users to create timelines that are derived from the basic tags, using the Boolean operations of AND, OR and NOT. To create such a composite timeline, the user first selects the component tags they are interested in by adding them to the displayed list of selected timelines. Next, the user highlights the tags they want to operate on and the operation they want to perform. The two operand timelines are then removed and replaced with the resultant timeline. This timeline is labeled in the tag selection window with a textual

representation of the Boolean combination, for example, “text AND presenter” In the case of NOT, if more than one tag is selected, they are each negated. When two or more timelines are combined as a result of a Boolean operation the new resultant timeline receives the color and position of the first operand. A walkthrough of the process is shown in Figure 24. This mode of interaction is based on the UI methods used for performing Boolean operations in 3D modeling, such as in the 3D graphics editing tool Maya.

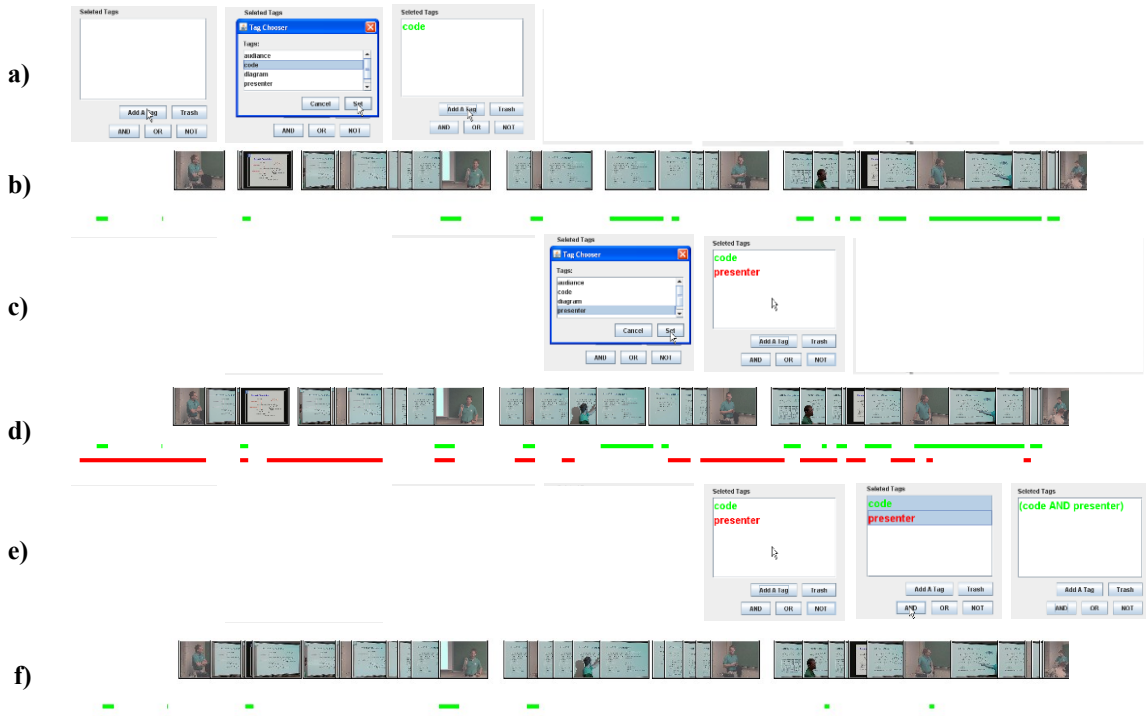

Figure 24: Walkthrough of the Tag Algebra process. a) The selection of tag ‘code’. b) The immediate display of the ‘code’ timeline, in green. c) The additional selection of the tag ‘presenter’. d) The immediate additional display of the additional ‘presenter’ timeline, in red. e) The creation of a composite ‘code and presenter’ timeline by selecting the components and selecting ‘and’. f) The immediate removal of the component timelines, and the display of the composite. ‘code and presenter’ timeline, in green.

The Boolean operation is carried out on the individual base segments, namely those that are displayed when the cut confidence is set to most permissive. Then as the cut confidence threshold is adjusted, the drawing of the timelines is carried out using the same method as for non-Boolean timelines. Thus, first the algebra is performed on the more exact data, and then the heuristic-based adjustments to represent the uncertainty of locality are applied. This form of multi-timeline display is one step beyond the state of the art. For example, the one-off MSNBC debate viewer [19] only allowed the displaying of single (hand-generated) Tag Timelines, not the union or intersection of them.

5.1.5. Linking Timeline and Frames

We have also included a highlighting tool, to assist the user in associating thumbnails and the multi-timelines. When the user mouses over the timeline, the segmented region of the timeline is highlighted in gray, its thumbnail is highlighted in red, the thumbnail is drawn in front of the other thumbnails, and a larger version of the keyframe is displayed in the browser. See Figure 25 for a close up of the timeline. The gray region extends from the start time of the current segment, to the start time of the segment that next passes the segment confidence threshold. When the user clicks on the timeline while the video is playing, the video will skip to the appropriate section of the video and continue playing from that location.

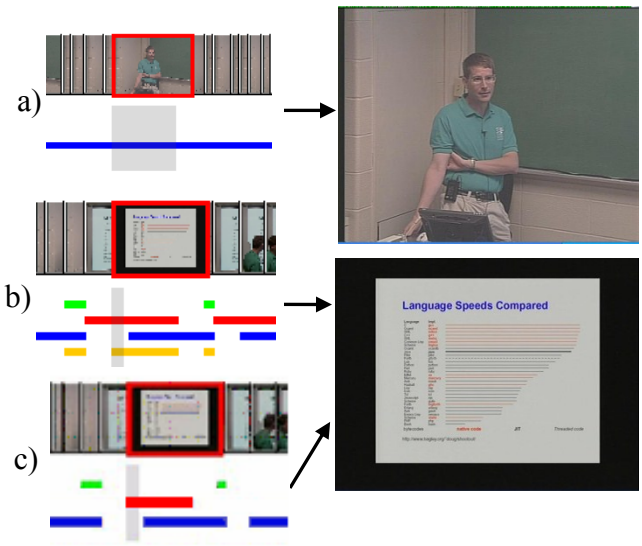

Figure 25: Close up examples of keyframes and their tag timelines. (a) Tag timeline for tag "presenter" and example highlighted frame. (b) Tag Timelines for "code", "text", "presenter" and "diagram" (from top to bottom). (c) The replacement of "text" and "diagram" from (b) with a Tag Timeline for "text AND diagram". The timelines now displayed, from top to bottom, are "code", "presenter", "text AND diagram".

5.2. Classification-Based Frame Tagging Pipeline

We enhance the VASTMM server and back end tools with the information provided by the Sort-Merge/SVM classifiers we have developed in order to power the Tag Timeline interfaces. The new backend components collect ground truth, generate classifiers, classify the keyframes, and load the data into the back end database.

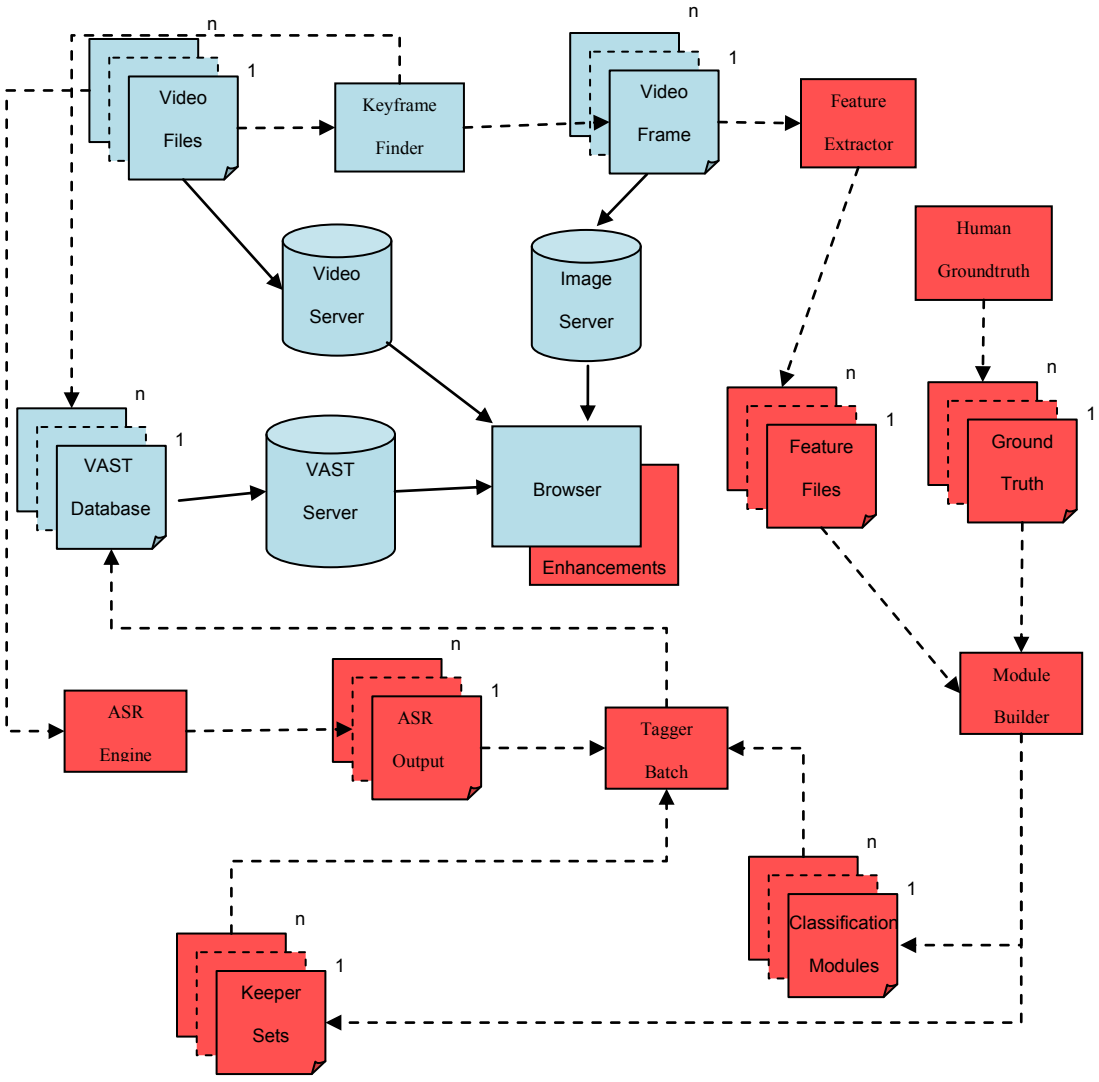

Figure 26: System diagram. Prior VASTMM components are in blue, additional components needed for VastMM-Tag are in red. Offline batch processes interactions are shown with dashed lines. Online interactions are shown with solid lines.

Figure 26 shows a diagram of the online system and how the classification modules we create are used to classify frames in the complete VastMM-Tag system. Classification speed is not fast enough to support a real-time method. Instead, an offline batch process is used to gather

all the requisite information, push it through the pipeline outlined in the figure, and finally load the tag information into the back end database.

5.3. User Studies

To evaluate the usefulness of the browser we conducted a preliminary user study and a full formal user study. The preliminary user study was done with a small set of users as a beta test of the software and as an initial data collection on which to form hypotheses for later experiments. All the participants were Columbia University students, both graduate and undergraduate, recruited though fliers around campus as well as postings to campus wide email lists. There is no inherent bias in the experiment group, because university students are the intended users of the software. As the number of participants in the study was low, none of these results can be considered statically significant. Before each experiment the subject was trained in the use of the browser though a series of training tasks.

5.3.1. Preliminary User Study

To evaluate how users take advantage of the system, a preliminary user study was conducted with five users. The purpose of this preliminary study was to observer how participants would utilize the newly developed tools. The observations gathered were used to from initial hypotheses for the following user studies. The users were each asked to complete between 15 and 20 search tasks. The tasks were completed in groups of five, and each group was performed using a different subset of the available tools. The questions were organized into five groups based on the content of the question. During the user study each set of five questions is

generated by randomly selecting, without replacement, five questions, one from each group. Consequently each question was asked at most once. Following is a list of the tasks sorted by type.

- Task Type 1 (Find a tagged segment)
    - Find a code example
    - Find a shot with text you can read on the screen
    - Find a shot of the presenter
    - Find a shot of a diagram
- Task Type 2 (Concept without others)
    - Find sections of the video where the presenter stands alone.
    - Find an unobstructed diagram (not blocked by the presenter).
    - Find a shot of a slide not blocked by the presenter.
    - Find a slide with computer code that is not part of a diagram.
- Task Type 3 (Subject Matter Identification)
    - The slide about “just-in-time compilers” is being talked about.
    - The slide about “preprocessors”.
    - The slide comparing “computer language running times”.
    - The slide about “abstract syntax trees”.
- Task Type 4 (Tag Co-location)
    - Find a segment where the presenter emphasizes a slide by pointing and gesturing.
    - Find the segment where the presenter appears with a diagram.
    - Find a diagram with words inside boxes.

    - Find a diagram incorporating computer code.
- Task Type 5 (Keyframe Finding)
    - Find a keyframe matching screen capture A (given)
    - Find a keyframe matching screen capture B (given)
    - Find a keyframe matching screen capture C (given)
    - Find a keyframe matching screen capture D (given)

The order of configuration usage was randomized to control for familiarity of the content of the video gained through subsequent search tasks. Users were allowed enough time to take each task to a successful completion, so completion rates were identically 100% throughout, and we report only completion times.

Table 2 shows the configurations used, the number of users for each configuration, and the average search time for each configuration. The first column, in yellow, shows the baseline browser configuration. The second column, in red, shows that with tags came an increase in search times, but when tags were combined with algebra (column 3 in green) performance was better than the baseline.

Table 2: Breakdown of user study browser configurations, including average time to completion in seconds.

Word Bubbles	X	X	X	X
Play	X	X	X	X
Keyframes	X	X	X	
Tags		X	X	X
Tag Algebra			X	X
Total Tasks	25	25	25	13
Total Users	5	5	5	3
Avg. Comp. (sec)	35.1	45.7	33.414	34.6

We recorded and time-stamped in fine detail how users transitioned from one interface component of the browser to another. Then we visualized the results. Figures Figure 27 (a-d) show four graphs where the nodes represent the usage of different user interface components, along with additional start, memory, and end nodes. An edge from node v to node w indicates that interface element *w* was used directly after interface element *v*. The size of the nodes represent the number of times an element was used, and the size of the edges represent the number of times the transition occurred, both aggregated over all users. Each graph is scaled separately to allow each to be readable and fit in the same area. All start nodes are of the same absolute size, however due to the possible sizes of other nodes, the start nodes could be scaled

down. This holds true for all subsequent graphs of this type. Each of the graphs details user behavior under the four browser configurations, respectively.

The nodes are labeled to correspond to interface elements. Word Bubbles indicates a use of the Word Bubbles generated by ASR; Play a use of the video (and audio) player; Keyframes a scan though the keyframes; and Tags involvement with the Tag Timelines (with or without algebra). Start and End represent the start and end of a task.

During the user study, we noticed that when users became familiar with the video after repeated search tasks, that often they would exploit a working memory of where in the video to find things. The transition between the start of a task and the first usage of one of the indicated interface objects is best attributed to such direct recall, and is indicated in the figures as Memory. In practice, this is equivalent to the Start state. We observed that in some of the tasks the user behavior went directly from Memory to End. This transition represents those completions where the user happened to be already looking at the target segment, or remembered exactly where the segment was.

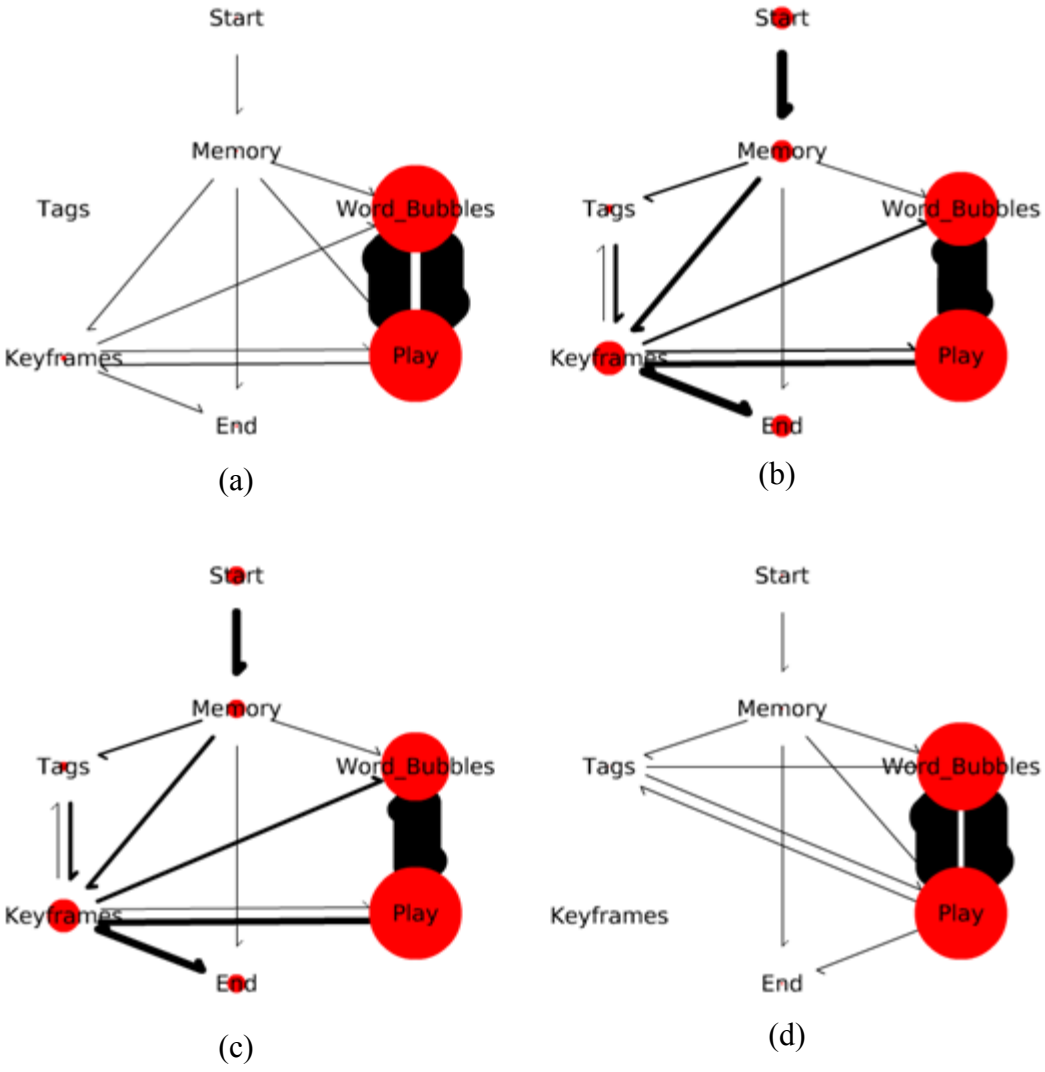

Figure 27: Interface transition graphs: (a) For 'Word Bubbles Play Keyframes' configuration. (b) For 'Word Bubbles Play Tag Keyframes' configuration. (c) For 'Word Bubbles Play Tags (with Algebra) Keyframes' configuration. (d) For 'Word Bubbles Play Tags (with Algebra)' configuration.

5.3.1.1. Paired Behaviors

From the data gathered during the user study, we noticed a number of user patterns. In the transition diagrams, there is a strong association between three pairs of interface elements: the Word Bubbles-Play pair; the Tags-Keyframes pair, and to a lesser extent the Play-Keyframes pair.

In Figure 27 Word Bubbles and Play are biconnected with heavy lines. This represents a high alternating movement of usage between these two interface elements during search tasks. It is also of note that in all four cases the only out edge from Word Bubbles is Play. When users searched with Word Bubbles, they always played the video afterwards, most often to confirm aurally the location of the words.

A similar association can be seen between Tags and Keyframes. In Figure 27, the only out edge from Tags is to Keyframes. This shows that when tags are first used, Keyframes are always used next, most often to confirm visually the location of the tags. The third pair, Play-Keyframes, also occurs, but less frequently this pair is also biconnected. This shows that sometimes Keyframes were used to localize a segment, which was then confirmed by using the player the user would often return to Keyframes if the target was not found.

This user behavior indicates that there is always a conformation step involved in these searches. The search itself may be performed solely using the abstract information provided by the browser enhancements, however users always appear to verify by either seeing or hearing actual information from the video, before being satisfied the target has been found.

5.3.1.2. Coarse-Fine Strategy

These alternating behaviors suggest that there is a three-tiered approach to completing this search task, in which a user will first use a coarser level of search before proceeding to a finer level. The coarsest level used is the user's own memory if it exists. Mid-level search is the use of either the Word Bubbles or the Tag Timelines. The finest level of search consists of confirming the target location by either watching/hearing the video, or by referencing the Keyframes.

From these observations we can conclude that there is a grammar that describes the most frequent search strategies of the users:

$$S\,M\,(B\,P \mid T\,K)^{*}\,E$$

where S=Start, M=Memory, BP=the alternation between Word Bubbles and player, TK=the alternation between tags and Keyframes and E=End. This grammar is visualized in Figure 28.

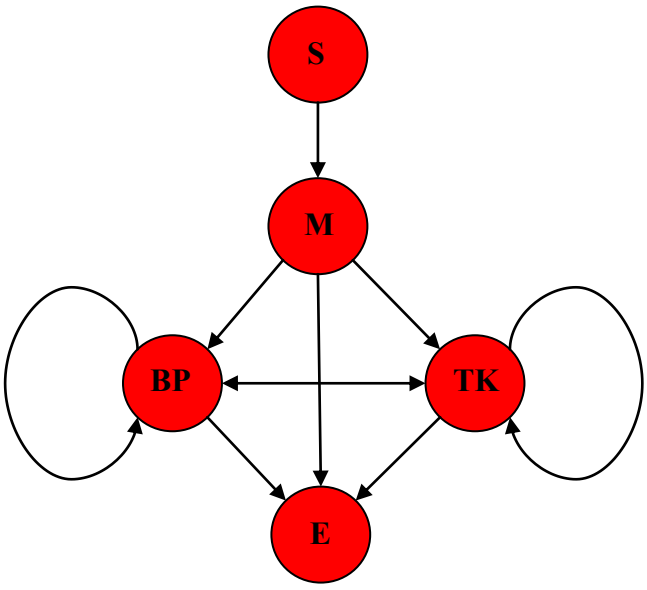

Figure 28: A composite graph of user interface transitions aggregated over all interface configurations and users.

This tiered search concept is also supported by the behavior of users when Keyframes are disabled; see Figure 27(d). In this case, there is now an out edge from Tags to Play. This suggests that users will attempt to substitute a different fine-level search for the missing one. In this figure there is also an edge from Tags to Word Bubbles, which may indicate that users switching from Tags-Keyframe search to Word Bubbles-Play search when they discover that Keyframes have been disabled in some of their tasks.

5.3.1.3. Word Bubbles are not Necessarily Popular

We also observe that in all configurations, that Word Bubbles-Play have more use than any other interface elements. This is not necessarily a good sign. When compared to the number of tasks in which each element was used (

Table 3) we can see that Tags and Word Bubbles are utilized equally at the middle level of search. So, although the Word Bubbles-Play pair gather more clicks, they are not necessarily *preferred* by users. (Anecdotal user reports confirm this preference.) Instead, the size of the Word Bubbles-Play pairs relative to the Tags-Keyframes pairs suggest that the latter is more *efficient*. (Anecdotal user reports confirm this also.).

Table 3: Table showing the number of tasks in which an interface object was used

Search Tool	# of tasks used in
Memory	58
Word Bubbles	29
Play	38
Tags	29
Keyframes	60

5.3.1.4. Algebra Required

A further observation is that Tag Timelines used without the Tag Algebra takes more time, when compared to the browser without tags at all, in the Word Bubbles-Play-Keyframe configuration; see Table 2. However, when the Tag Algebra is available, the average time to completion is reduced, compared to the Word Bubbles-Play-Keyframe configuration. This suggests that tags are useful, but only when the algebra tools are also available.

5.3.2. Tag Algebra Study

We next ran a redesigned user study with the intent to evaluate the usefulness of the Tag Algebra interface. This study included 15 users assigned to different browser configurations, as opposed to the preliminary study where each user was given multiple configurations. This design change intended to eliminate the possible effects of increased performance through experience

with the interface and the video. Each user was asked to complete the same 5 search tasks as opposed to the 20 asked before. These five consisted of "Find a shot of a diagram", "Find sections of the video where the presenter stands alone.", "The slide about just-in-time compliers is being talked about.", "Find a diagram incorporating computer code", and "Find a keyframe matching example D". Users were allowed enough time to take each task to a successful completion, so completion rates were identically 100% throughout, and we report only completion times.

Table 4 shows the configurations used, the number of users who used each configuration, and the average search time for each configuration. The frist column (in yellow) is the base line browser. An interesting observation is that Keyframes are required for efficient search. In the one configuration without access to Keyframes (column 4, labeled in red) the search time increased dramatically from 39.9 seconds to 102.4 seconds, more than double the needed time. Secondly in this experiment, users who had access to Tags (Column 3 in green), took slightly less time than the users with the baseline browser (Column 2 of in white). This shows that tags by themselves are indeed useful.

Table 4: User study browser configurations and average time to completion in seconds.

Word Bubbles	X	X	X	X
Play	X	X	X	X
Keyframes	X	X	X	
Tags		X	X	X
Tag Algebra			X	X
Total Users	4	4	4	3
Avg. Comp. (sec)	39.9	33.1	24.3	102.4

Figure 29 shows the transition diagrams generated by the users in this study. The nodes are labeled to correspond to interface elements the same as in the previous transition diagrams. These diagrams confirmed the pair-wise behavior seen in the preliminary user study. The association in this experiment is not as pronounced as in the first study, however the paths from Tags to Keyframe and from Word Bubbles to Player are still strongly represented.

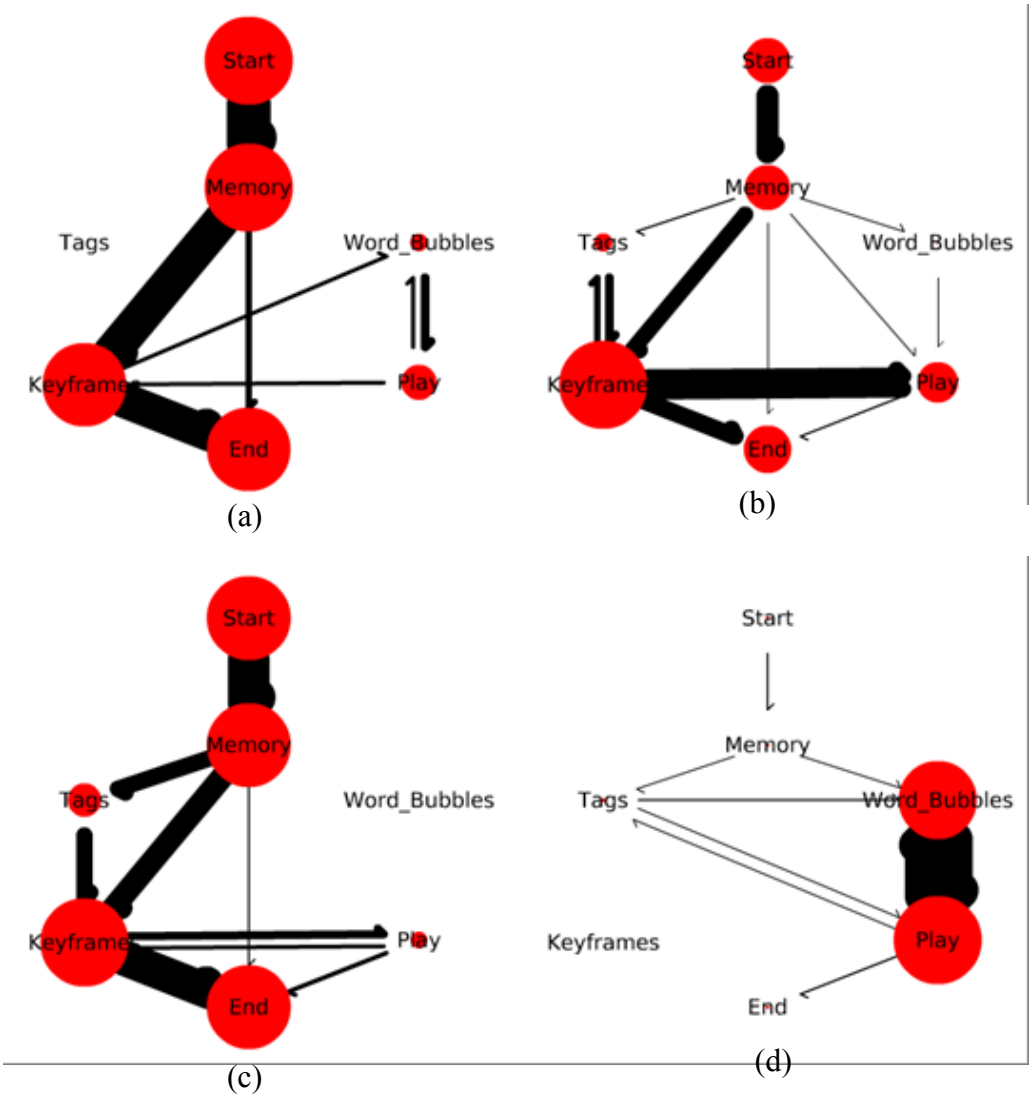

Figure 29: Interface Transition Graphs (a) For 'Word Bubbles Play Keyframes' configuration. (b) For 'Word Bubbles Play Tag Keyframes' configuration. (c) For 'Word Bubbles Play Tags (with Algebra) Keyframes' configuration. (d) For 'Word Bubbles Play Tags (with Algebra)' configuration.

For this study we also collected subjective user feedback. We asked the users to rate each tool they were allowed to use in their trial on a scale of 1 to 5 where 1 means not useful for search and 5 means extremely useful for search. These results are aggregated in Table 5. This

table shows that Keyframes and Visual Tags are rated the highest and that Player and Word Bubbles are rated lowest. This suggests that tags are a good interface in the preferences of users. However, Algebra got a mediocre score. This is because it was rated bimodally. Some users gave it a high score and the rest gave it a low score. The users who gave it a high score most often were users who did not have access to Keyframes. This suggests that Algebra does assist in searching, but this assistance is only needed when the search will take a very long time, as in the case of no Keyframes. The Keyframes make scanning the video so efficient that the benefit provided by algebra is minimized.

We also asked the users to report on any enhancements they might have wanted. We asked them "Was there anything you wanted to do to search but couldn't". Users who had access to the Word Bubbles tool consistently commented that it was difficult to use them for search. One possible reason is that the bubbles were not alphabetized but rather sorted by time of occurrence and by predictiveness value. This is also evidenced by the low usefulness rating 2.87 of Word Bubbles. Many users of Word Bubbles asked, unprompted during the experiment if there, was a way to search or filter the Word Bubbles to see only those bubbles they were interested in.

Table 5: Results of algebra user study satisfaction survey aggregated by configuration

All Configurations					
	Player	Keyframes	Word Bubbles	Visual Tags	Algebra
Avg	2.13	4.67	2.87	4.13	3.30
Std	1.01	0.47	1.54	1.15	1.79

Configuration 1: Player - Keyframe - Word Bubbles					
	Player	Keyframes	Word Bubbles	Visual Tags	Algebra
Avg	1.80	5.00	2.80		
Std	0.75	0.00	1.83		

Configuration 2: Player - Keyframes - Word Bubbles - Visual Tags					
	Player	Keyframes	Word Bubbles	Visual Tags	Algebra
Avg	1.67	4.33	2.67	3.33	
Std	0.47	0.47	1.25	0.47	

Configuration 3: Player - Keyframes - Word Bubbles - Visual Tags - Algebra					
	Player	Keyframes	Word Bubbles	Visual Tags	Algebra
Avg	2.00	4.75	2.50	3.25	1.75
Std	1.00	0.43	1.66	1.48	0.83

Configuration 4: Player - Word Bubbles - Visual Tags - Algebra					
	Player	Keyframes	Word Bubbles	Visual Tags	Algebra
Avg	2.67		3.67	4.33	5.00
Std	1.70		0.47	0.47	0.00

5.3.3. Word Tag User Study

Due to the overwhelming complaints of users about Word Bubbles, we developed an alternative method for display of the ASR content. Users asked for a way to select just one word and see the bubbles for that. Some also asked to see multiple selected words. We turned the bubbles into tags, to support this requested behavior. These tags are applied to the video

segments in an analogous way to the Visual Tags. A separate interface was developed for the selection of Word Tags which incorporated an auto-complete feature.

Therefore, we started 3 more experiment groups which used Word Tags. The participants for these new groups were drawn from the same population as the previous experiment groups. We continued to add people to the other experiment groups as well. The combined distribution of users into groups can be seen in Table 6. The first column of (in yellow) shows the performance using the baseline browser. In this expanded group, the configuration without keyframes (column 4, in red) is still the least well performing. Columns 3 and 7 (labeled in green) are the two configurations with tag algebra. Both show decreased search time compared to the baseline configuration.

Table 6: Extended User study browser configurations and average time to completion in seconds.

Word Bubbles	X	X	X	X			
Play	X	X	X	X	X	X	X
Keyframes	X	X	X		X	X	X
Visual Tags		X	X	X		X	X
Word Tags					X	X	X
Tag Algebra			X	X			X
Total Users	5	5	5	4	4	3	3
Avg. Comp (sec)	56.2	35.5	26.1	93.0	22.1	54.3	33.2

During the user study each participant was given a satisfaction survey after they completed the tasks. This survey asked the users to rate each one of the tools they had access to, for usefulness on a scale of 1 to 5, where 1 is not useful and 5 is very useful. Table 5 shows the results of the survey aggregated by the available configurations.

We continued to record the interface transitions. They are visualized in Figure 30 and Figure 31. In Figure 31 there is a new node, “Word_Tags” that replaces “Word_Bubbles”. We also continued to administer the user surveys during the study. Table 7 reports on the combined data. Table 8 shows the same data grouped not by individual configurations but also by configuration types.

Users found the Word Tags significantly more useful than the Word Bubbles they replaced, and found the newly added Word Tags equally useful to the already existing Visual Tags. The algebra was then also found to be more useful, by users who had configurations with Word Tags, Visual Tags and Algebra. We observed that the algebra was used to combine a Visual Tag with a Word Tag for a successful search. We also noticed that many users attempted to use two Word Tags with an AND operator to conduct a search, but failed at its usage. The issue here is the way the algebra and the Word Tags interact. Words are very ephemeral, they are spoken and take a fraction of a second to say. Further, by the nature of how human speech is structured, content words can never co-occur in time, they must be spoken one after another, and most of the time are separated by a variety of function words. Even though the interface will label an entire video segment with a word, rather than the exact time frame, it is very rare for two words to overlap significantly in time. Due to the way the algebra is computed, if this overlap

does not occur, a timeline like “Reference AND Manual” will show a completely blank timeline. However if the user viewed the two timelines separately they would see a section of the first timeline for “Reference” followed closely by a section of the timeline for “Manual”. That approximate intersection is the section they are looking for. The exact method to provide such an approximate intersection is not yet known an area for future research.

Table 7: Results of Word Tag user study satisfaction survey aggregated by configuration

All Configurations						
	Player	Keyframes	Word Bubbles	Visual Tags	Word Tags	Algebra
Avg	2.13	4.67	2.87	4.13	4.11	3.30
Std	1.01	0.47	1.54	1.15	0.57	1.79

Configuration 1: Player - Keyframe - Word Bubbles						
	Player	Keyframes	Word Bubbles	Visual Tags	Word Tags	Algebra
Avg	1.80	5.00	2.80			
Std	0.75	0.00	1.83			

Configuration 2: Player - Keyframes - Word Bubbles - Visual Tags						
	Player	Keyframes	Word Bubbles	Visual Tags	Word Tags	Algebra
Avg	1.67	4.33	2.67	3.33		
Std	0.47	0.47	1.25	0.47		

Configuration 3: Player - Keyframes - Word Bubbles - Visual Tags - Algebra						
	Player	Keyframes	Word Bubbles	Visual Tags	Word Tags	Algebra
Avg	2.00	4.75	2.50	3.25		1.75
Std	1.00	0.43	1.66	1.48		0.83

Configuration 4: Player - Word Bubbles - Visual Tags - Algebra						
	Player	Keyframes	Word Bubbles	Visual Tags	Word Tags	Algebra
Avg	2.67		3.67	4.33		5.00
Std	1.70		0.47	0.47		0.00

Configuration 5: Player - Keyframes - Visual Tags - Word Tags						
	Player	Keyframes	Word Bubbles	Visual Tags	Word Tags	Algebra
Avg	3.00	4.67		5.00	4.33	
Std	0.00	0.47		0.00	0.47	

Configuration 6: Player - Keyframes - Word Tags						
	Player	Keyframes	Word Bubbles	Visual Tags	Word Tags	Algebra
Avg	2.00	4.67			4.33	
Std	0.82	0.47			0.47	

Configuration 7: Player - Keyframes - Visual Tags - Word Tags - Algebra						
	Player	Keyframes	Word Bubbles	Visual Tags	Word Tags	Algebra
Avg	2.00	4.33		5.00	3.67	3.67
Std	0.82	0.47		0.00	0.47	1.89

Table 8: User satisfaction survey aggregated by configuration types.

Word Tags Configurations (#5,#6 & #7)						
	Player	Keyframes	Word Bubbles	Visual Tags	Word Tags	Algebra
Avg	2.33	4.56		5.00	4.11	3.67
Std	0.82	0.50		0.00	0.57	1.89

Word Bubbles Configurations (#1, #2, #3 & #4)						
	Player	Keyframes	Word Bubbles	Visual Tags	Word Tags	Algebra
Avg	2.00	4.75	2.87	3.60		3.14
Std	1.10	0.43	1.54	1.11		1.73

Configurations With Algebra (#3, #4 & #7)						
	Player	Keyframes	Word Bubbles	Visual Tags	Word Tags	Algebra
Avg	2.20	4.57	3.00	4.10	3.67	3.30
Std	1.25	0.49	1.41	1.22	0.47	1.79

Configurations Without Algebra (#1, #2, #5 & #6)						
	Player	Keyframes	Word Bubbles	Visual Tags	Word Tags	Algebra
Avg	2.07	4.71	2.75	4.17	4.33	
Std	0.80	0.45	1.64	0.90	0.47	

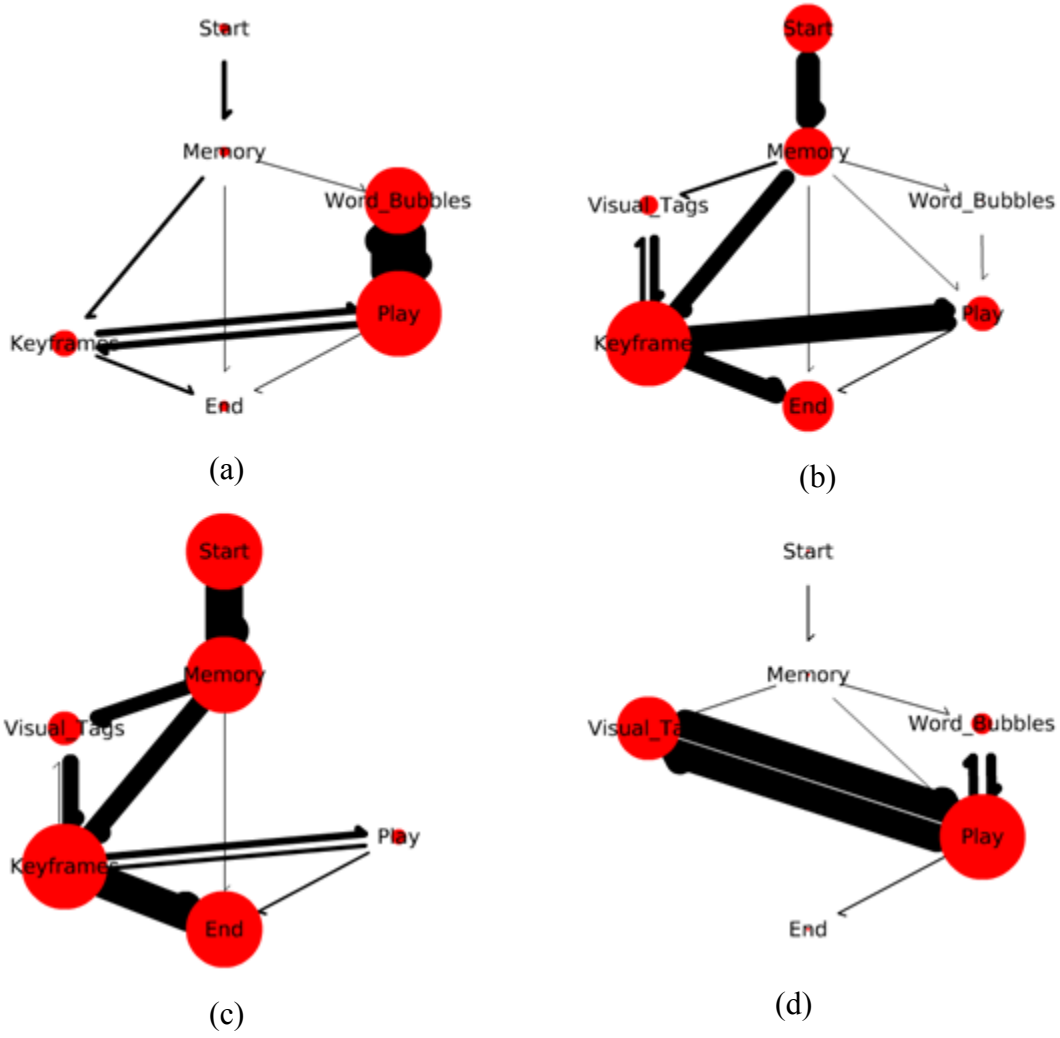

Figure 30: Interface Transition Graph (a) For 'Word Bubbles, Play, Keyframes' configuration. (b) For 'Word Bubbles, Play, Visual Tags, Keyframes' configuration. (c) For 'Word Bubbles, Play, Visual Tags (with Algebra), Keyframes' configuration. (d) For 'Word Bubbles, Play, Visual Tags (with Algebra)' configuration.

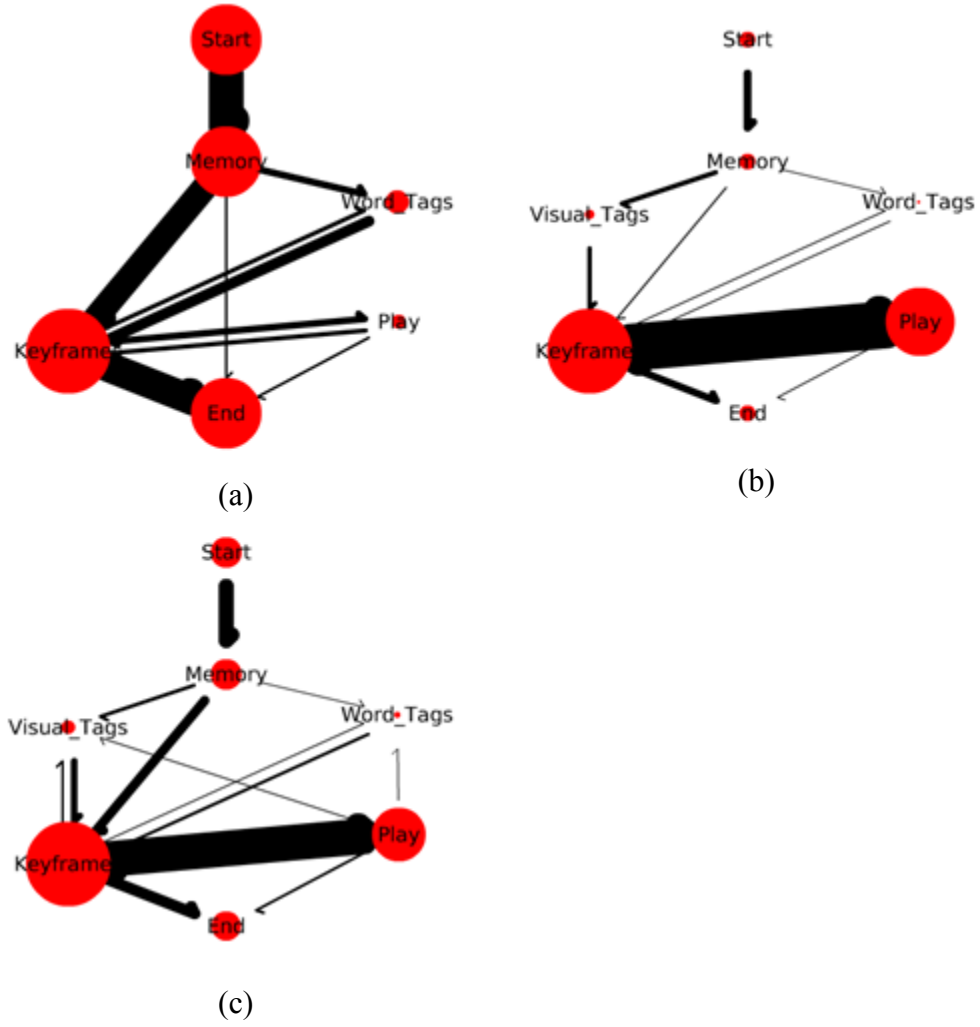

Figure 31: Interface Transition Graph (a) For 'Word Tags, Play, Keyframes' configuration. (b) For 'Word Tags, Play, Visual Tags, Keyframes' configuration. (c) For 'Word Tags, Play, Visual Tags (with Algebra), Keyframes' configuration.

These user studies support the utility of the Tag Timeline display. The timelines allowed users to improve their search speed, and were well received. The difference in user ratings and search times for Word Bubbles and Word Tags is noticiable.

6. Conclusions and Future Work

We have created a video browser for course lecture videos with the ability to assist users in completing intra-video search tasks. We have created and evaluated visual classifiers and introduce a multi-Tag Timeline user interface for expediting these search tasks. We also conducted user studies to evaluate the effectiveness of the interface.

6.1. Summary & Conclusions

The Keeper heuristic was a valuable addition to the Sort-Merge feature selection that we developed. This heuristic allowed us to identify feature sets that were useful, in that they improved classification performance over a baseline measure. Additionally the fast cutoff to Sort-Merge provided an additional advantage when performing the search for the best feature set. It provides a criterion for deciding when to stop Sort-Merge, rather than carrying out the entire process to its end state. The result is a savings of time for the search process without loss of classification performance.

As a result of our experiments with feature fusion, we determined that Late Fusion is the superior method for our data set. Late fusion had benefits in both the Average Precision of the classifier created (although slight) and also a significant time savings in performing the Sort-Merge selection of the total feature set.

We also discovered that particularly for search tasks, the Word Bubbles interface was inadequate. This was evidenced strongly by qualitative feedback during the user studies. A

repeated comment by study participants was, “Can I search the bubbles?” This was further evidenced by user reaction to the replacement interface of Word Tags. Word Bubbles received a low average satisfaction rating, while its replacement Word Tags received one that is much higher. Users also expressed that they found the Word Tags more useful. We hypothesize that the increased preference for Word Tags over Word Bubbles is due in part to a constant interface. Both types of semantic data are now displayed in the same way. Another factor in the increased preference for Word Tags is the tag pair better with the Keyframes which are the most crucial element to a timely search,

The major contribution we made to the browser interface is the Tag Timeline displays and the Tag Algebra. We hypothesized that this interface would assist the users in completing their search tasks. We also found that making the interface for visual and audio information would increase usefulness. Both Visual Tags and Word Tags scored high in user perception of usefulness. In all experiment groups, users with access to the tag interface had a reduced search time compared to the baseline browser which had only the following tools: Keyframes, Player & Word Bubbles. This showed that Tag Timelines improved the search time over the baseline browser.

Similarly we saw a beneficial effect on search time provided by the Tag Algebra. There were three groups that had access to Tag Algebra, one of which did not have access to Keyframes. The group that did not have Keyframes was the only Tag Algebra group to show an increased search time compared to baseline. From this we conclude that Keyframes are required for users to effectively use both tags and Tag Algebra. This is also evidenced by the pairwise

behaviors we noticed in users. Users preferred to use tags with Keyframes and preferred to use Word Bubbles with the player. When Keyframes were not available users were then forced to use the tags with the player, which resulted in longer search times.

A comparison between user groups whose only difference was their access to the algebra shows a decrease in search time associated with the algebra. There were two such comparisons, and in both cases the user group with the algebra had a faster average search time compared to the user group without algebra.

Despite the fact that the algebra improved average search performance in certain groups, surprisingly those groups did not all consistently rate the algebra high for usefulness. The Player - Keyframes - Word Bubbles - Visual Tags - Algebra group rated the algebra quite low on usefulness, while the Player - Word Bubbles - Visual Tags – Algebra group gave it a perfect score and Player - Keyframes - Visual Tags - Word Tags - Algebra group gave it a medium one. This shows that some groups found algebra very useful, some not at all, and some in the middle.

We hypothesized there are two factors at play. The first factor is the amount of gain a possible use of algebra can give. From average search times, it is clear that the Keyframe tool provides the greatest improvement in search time. We also notice that users without access to the Keyframes rate Algebra the highest. We hypothesized that this is because those users have the most marginal benefit from using the algebra. In other words, if algebra provides a percentage speed up consistent among all groups, those users with the longer search times will get the most reduction in actual search time. The second factor is that users who gave Algebra a medium

score had access to both Word Tags and Visual Tags. We noted that users found it useful to algebraically combine one Word Tag with one Visual Tag to form a query, thus finding it useful.

The observation of the pairwise UI component usage pattern leads us to conclude there are two different methods of search that users employ. We noticed that users would use Keyframes and Visual Tags together and would also use Word Bubbles and Player together. This could be caused by visual versus aural differences in search strategies. When the Visual Tags – Keyframe pair was employed, users appeared to be searching based on visual cues: adding tags to narrow down the search space, and using Keyframes to confirm. When the Word Bubbles – Player pair was employed, the users appeared to be searching based on audio cues: using the bubbles as an indication of where to look and then listening to the video to find the confirmation. However, an alternative explanation is that the particulars of the implementation of the interface pairs make them easy to use together. We noted that when Word Tags replaced Word Bubbles, users then used the Word Tags in conjunction with Keyframes and not with Player.

Hollow bars are a method of communicating uncertainty in tag location to the users. This method was initially designed as a debugging aid, but later turned out to be a useful cue to the users. However the method is not perfect especially when used for the Tag Algebra. The current method of creating algebraic timelines is to perform the Boolean operation only considering the high precision solid bars, then generating hollow bars based on the scene segmentation level. An AND operation will only result in a new solid bar where both concepts had solid bars to begin with. This method works well for combining Visual Tags together or combining Visual Tags with Word Tags. However, this method runs into problems when combining Word Tags with

other Word Tags, due to the short duration in which Word Tags occur. Words usually appear in adjacent segments, rather than the same segment. In a situation where the two Word Tag operands are in adjacent segments, regardless of the scene segmentation level, an AND operation will produce no response using the current method.

One of the major hurdles for the machine learning portion of the thesis was the collection of ground truth data for the feature selection experiments. The ground truth tagging tool we developed was ergonomically designed, and was an invaluable tool in collecting this data. The tagging method developed, with a constant stream of keyframes surrounded by context, allowed the tagger to view the keyframes at a rapid pace, tagging them as they went by.

The feature visualization tools were created to give insight into which features were being selected by the feature selected process. These tools turned out to be useful in the verification of the feature selection process. It also allowed us to gain insight into what the selection process was selecting for. These insights were very valuable: in the case of the presenter detector, it made it clear that it preferred regions of green, white and red. We noted that in our data set, shots with the presenter have large areas of red (skin tone) combined with green (blackboard) and white (walls and projector screens) backgrounds.

6.2. Future Work

The first direction for future work would be to expand the Visual Tag set. We developed at tag ontology of Visual Tags for lecture videos, see section 3.1. This thesis only created a small number of tags, which is enough for a proof of concept. However for a viable product a larger set

of Visual Tags would be needed. The tag ontology we created shows a framework for creating new detectors for new tags. For each of the tags low level visual features would need to be identified that are able to detect them. These features need to be implemented and tested on the lecture video data set. As part of this process, each one of these new features should have a feature visualizer tool created for it, as these tools have proved to be very valuable currently in identifying features that perform well at certain classification tasks. The current state of classifier generation using Sort-Merge feature selection with the keeper heuristic is time consuming. Methods for optimizing the process should be explored.

Once a large set of Visual Tags is available experiments can be conducted to identify which of the generated tags are useful for search. One analysis would look for correlations in the response of pairs of tags. If two tags are strongly correlated, they may provide redundant semantic information and should merged together into one tag. Another analysis would be to identify keyframes that never get tagged. This will indicate possible new tags to explore. Finally, a user study can be conducted where users are given a series of search tasks, and there tag usage behaviors are analyzed. This analysis would identify frequently used tags, tags used in combinations and tags likely to be part of a boolean expression.

Another direction for future work is to examine unconstrained user behavior. User studies thus far have consisted of directed, task driven, experiments. It would be useful to examine how student currently enrolled in a course would make use of such a tool. The videos of a lecture course could be made available to students during the second half of the semester (post midterm). The student's browser usage patterns, including how often the tool was used, for how

long, which tools were used and in which order can be recorded. This information can lend insight into which tools are preferred or are easier to use. Also impact on a student's grades can be measured. One hypothesis is that students who make heave use of the tool will see their grades improve from midterm (when they had no access to it) to final (when they were able to use it as a studying aid).

The Ground Truth Tagging Tool we developed can be expanded and made available to other researchers for use in collecting ground truth. The tool as created is easy to use for the ground truth collection. However, the import and export of the ground truth data can be expanded upon. The tool could also be extended to allow multiple simultaneous users to work on generating ground truth for the same data set.

Another area of future word based on user feedback, is to add Word Tags for words that can be seen, in addition to the words one can hear. For example, we would add the text that appear on the slides to the Word Tag set. This approach would feed frames tagged with the Visual Tag 'text' through an OCR tool to get an ASCII version of the text in that keyframe. This output would then be filtered through the course text index similarly to what is done to the ASR output.

We observed that there was no universal agreement among users of Tag Algebra as to its perceived usefulness. We posited two hypotheses about which users found algebra useful and which did not. An additional user study should be conducted to reexamine the interaction of algebra with the other search tools. The first hypothesis suggests that users who get the most marginal benefit from the algebra find it the most useful. To evaluate this effect we should

collect data on a Player-Visual Tag-Word Bubble user group. We can then compare the following group pairs to find to what extent algebra affects the search times and if there are any correlations between benefit and perceived usefulness.

1a) Player - Visual Tag - Word Bubble

1b) Player - Visual Tag - Word Bubble – Algebra

2a) Player – Keyframes – Word Bubbles

2b) Player – Keyframes – Word Bubbles – Algebra

3a) Player – Keyframes – Visual Tags – Word Tags

3b) Player – Keyframes – Visual Tags – Word Tags – Algebra

The second hypothesis suggests that users who can combine Tags found the Algebra useful. Also we noticed that users who did not have Keyframes found Algebra more useful. A new user study can be conducted limiting the users only to Word Tags and Algebra, in since previous experiments, we observed users who liked Algebra would combine one Word Tag with one Visual Tag. This new experiment would explore the effect on the user ratings of algebra if only Word Tags alone were available. The following groups should be run as a user study and the user preferences compared.

Player – Keyframes – Word Tags – Algebra

Keyframes – Word Tags – Algebra

Player – Word Tags – Algebra

But prior to doing this new experiment, we need to address the problem that the current method for computing algebra is inadequate for AND operations using two Word Tags as operands. A modification to the algebra needs to be implemented and subsequent experiments conducted to compare its usefulness to the current method. One possible modification is to not to always perform the algebraic calculations at the most fine grained scene segmentation level, but rather to take the current segmentation level into account. At an arbitrary segmentation level there will be both solid bars and hollow bars, and one can perform the algebra on these bars following these rules

1. Solid bar AND Solid bar → Solid bar
2. Solid bar AND Hollow bar → Hollow bar
3. Hollow bar AND Hollow bar → Hollow bar.

In those situations where there is a Word Tag followed closely by another Word Tag, this version of the algebra will produce a hollow bar, rather than no response. However, the response of this computation method is very sensitive to the currently selected scene segmentation level.

An experiment can be conducted to further examine the observation of the Visual Tags – Keyframe paired usage pattern and the Word Bubble – Player paired usage pattern. We hypothesized that this pairing correlates to the modality the user employing: a Tags – Keyframe search being visual and a Word Bubble – Player search being aural. The hypothesis is that the behavior a user exhibits is partly determined by the current search task, and partly that some are visual people and some are audio people. To confirm this we can design an experiment to provide subjects with a series of tasks in three categories: tasks that must be completed visually,

tasks that must be completed aurally and tasks that can be completed either way. Visual only tasks would ask the user to find something that never makes a sound and is never talked about in the video. An audio only task would ask the user to find something only ever talked about, something that is not seen in the video, nor has readable text indicating the concept.

A task that could be searched for either way could consist of something that is both spoken about, and has text on screen about it, for example. a lecturer reading from a slide. If our hypothesis is correct, the interface elements used during the search tasks that could go either way will indicate that users search preference. We would expect people to first try to use their preferred search method, even on searches that must be completed with their non-preferred method, thus taking more time to complete their search. We also predict that if a task is extremely clear about the needed search method that will override a user's presences. We expect the task "Find where the professor says the word combination" will result in an aural search, regardless of the individual user's search preference.

Another area of future work would be to enhance the UI methods for selecting tags. The current method for selecting Word Tags is an auto complete type-in box. This method relies on the user having an idea of what they want to search for. The browsing feature of this interface is a simple alphabetically sorted flat list, which is very cumbersome. A better way would combine type-in auto-complete with a view that contains some context. For example, by combining auto-complete with a tag cloud, where each tag is sized based upon its predictiveness value or its number of occurrences in the video. Another method to explore would be a way to browse tags

in a tree view of a hierarchical list. This would be particularly useful for a tag set that was very large and hierarchically based as in a table of contents.

7. References

[1] Isabelle Guyon and Andre Elisseeff, “An Introduction to Variable and Feature Selection” Journal of Machine Learning Research 3, 2003

[2] Luis Carlos Molina, Llufs Belanche, Angela Nebot, “Feature Selection Algorithms: A Survey and Experimental Evaluation” Proceedings of 2002 IEEE International Conference on Data Mining, 2002

[3] Yan Liu, John R Kender “Fast Video Segmentation Retrieval by Sort-Merge Feature Selection, Boundary Refinement and Lazy Evaluation” Computer Vision and Image Understanding, 2003

[4] Alajandro Jaimes and Shih-Fu Chang, “Automatic Selection of Visual Features and Classifiers” Storage and Retrieval for Media Databases, 2000

[5] Vakkalanka Suresh, C. Krishna Mohan, R. Kumara Swamy, and B. Yegnanarayana “Content-Based Video Classification Using Support Vector Machines” Proceedings of 11th International Conference on Neural Information Processing, 2004

[6] Rong Yan, Jelena Tesic and John R. Smith “Model-Shared Subspace Boosting for Multi-tag Classification” International Conference on Knowledge Discovery and Data Mining, 2007

[7] Jing Huang, S. Ravi Kumar, Mandar Mitra, Wei-Jing Zhu, Ramin Zabih, "Image Indexing Using Color Correlograms," pp.762, 1997 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1997

[8] DK Iakovidis, DE Maroulis, SA Karkanis, IN Flaounas “Color texture recognition in video sequences using wavelet covariance features and support vector machines” Proceedings of Euromicro Conference, 2003

[9] J. Laaksonen, E. Oja, M. Koskela and S. Brandt, “Analyzing Low-Level Visual Features Using Content-Based Image Retrieval”, International Conference on Neural Information Processing, 2000

[10] Yongqing Sun and Shinji Ozawa “Semantic-Meaningful Content-based Image Retrieval in Wavelet Domain” ACM SIGMM International Workshop on Multimedia Information Retrieval, 2003

[11] Mitchell J. Morris and John R. Kender “Sort-Merge Feature Selection and Fusion Methods for Classification” IEEE International Conference on Multimedia and Expo, 2009

[12] Kadir A. Peker, A. Aydin Altan, Ali N. Akansu "Low-Level Motion Activity Features for Semantic Characterization of Video" IEEE International Conference on Multimedia & Expo, 2000

[13] Nitesh V. Chawla and Kevin W. Bowyer "Random Subspaces And Subsampling For 2-D Face Recognition" IEEE Conference on Computer Vision and Pattern Recognition, 2005

[14] John R. Kender, Boon-Lock. Yeo "On the Structure and Analysis of Home Videos", In Proceedings of the Asian Conference on Computer Vision, 2000

[15] Chris J.C. Burges and Bernhard Scholkopf "Improving the Accuracy and Speed of Support Vector Machines" Advances in Neural Information Processing Systems, 1997

[16] Darrin P. Lewis, Tony Jebara, and William Stafford Noble "Support vector machine learning from heterogeneous data: an empirical analysis using protein sequence and structure" Bioinformatics. pp. 2753-2760, 2006.

[17] Francis Li, Anoop Gupta, Elizabeth Sanocki, Liwei He, Yong Rui. "Browsing digital video". Proceedings of the Conference on Human Factors in Computing Systems. pp. 169-176, 2000

[18] Alex Haubold, John R Kender "Analysis, User Interface, and their Evaluation for Student Presentation Videos" 2007 IEEE International Conference on Multimedia and Expo, 2007

[19] MSNBC Debate Viewer, http://www.globenewswire.com/newsroom/news.html?d=151266 Date Accessed: 10/02/2008.

[20] Hyowon Lee, Alan F. Smeaton, Jonathan Furner "User Interface Issues for Browsing Digital Video" p 1. BCS IRSG Colloquium on Information Retrieval, 1999

[21] Stephane Marcharnd-Maillet and Marcel Worring "Benchmarking Image and Video Retrieval: an Overview" pp. 297-300 ACM SIGMM International Workshop on Multimedia Information Retrieval, 2006

[22] Henning Muller, Wolfgang Muller, David McG. Squire and Thierry Pun "Performance Evaluation in Content-Based Image Retrieval: Overview and Proposals" Technical Report; Computer Vision Group, Computer Science Center, University of Geneva, 2000

[23] Huiping Li, David Doermann, Omid Kia. "Automatic text detection and tracking in digital video" IEEE Transactions on Image Processing, 2000

[24] Greg Pass, Ramin Zabih, Justin Miller, "Comparing Images Using Color Coherence Vectors" pp. 65-73, ACM international conference on Multimedia, 1997.

[25] “List of Words Ignored by Search Engines” http://www.webconfs.com/stop-words.php. Date Accessed: 05/18/2011