Nicolith: フィードを効率よく眺めるための情報提示手法の提案

草野孔希[†1] 角田博保[†2] 赤池英夫[†2]

本研究では、ユーザがフィードを受動的に視聴可能で、かつ負担を与えないフィードブラウズ手法の提案、及び提案の有用性について簡単な考察を行った。Web の発達に伴い Web 上には人が 1 人では一生かかっても読み切れないほどの情報が溢れている。そのため、情報の収集および収集した情報の取捨選択の一つの方法として、ウェブサイトのコンテンツ概要を配信用に加工した文書である「フィード (Feed)」が発案され、広く利用されている。特に最近の 5 年程で、音声記事のフィード配信や、動画投稿サイトなどが配信する動画記事のフィード配信が増えている。しかし、近年見られるフィードリーダの多くは、テキスト記事を「読む」という事を効率的にする為に開発されている一方で、音声記事や動画記事のフィード配信に関しては対応が遅れている。特に、音声記事や動画記事を受動的に視聴するためのシステムは少ない。そこで本稿では、音声記事や動画記事のフィード配信を受動的に視聴出来るシステムについて考察し、システムのケーススタディとして動画サイトのフィード：特にニコニコ動画[1)] のフィードに注目したフィードブラウザ、Nicolith（にこりす）を開発した。

Nicolith: The Efficient Browsing Method for Web-Feeds

Kouki Kusano,[†1] Hiroyasu Kakuda[†2] and Hideo Akaike[†2]

1. はじめに

現在 Web 上には、1 兆を超えるページが存在[2)] し、それらが Web 検索・ブラウジングをすることで到達できる状態にある。しかし、何か特定の情報を探しているユーザにとって、これら莫大な情報の大半は殆どが価値の無いものである。そのため、特定の情報もしくは特定のカテゴリに属する情報を効率よく取捨選択する一つの方法として、フィード技術が発達する事となった。日本で利用されるフィードの中で最もメジャーなフォーマットは RSS1.0（以下 RSS）である。RSS とは RDF[⋆1] Site Summary の略であり、RDF を、ウェブサイトのコンテンツ概要を配信するためにカスタマイズしたフォーマットである。RSS を購読する為に利用されるフィードリーダの殆どは、情報の表示方法や検索手法が「テキスト情報を読む」という能動的な情報取得を主眼に開発されている。しかし、これらのフィードリーダは、音声記事や動画記事のフィード配信において、TV などの視聴方法に代表される「付けっぱなし」をする事が困難である。ユーザは動画を見終わるたびに次の動画を見に行く作業を行う必要がある他、連続再生用にプレイリストを作成するとしても、作成に大きな手間がかかる為である。システムが付けっぱなしに代表される受動的な情報提示をするためには、これらの操作負荷を軽減する事は急務であると言える。

さて、フィードを扱う際に、受動的な情報取得を支援するには、具体的にどのようなシステムを組めば良いだろうか。注意すべきことは、一度に表示する情報量や一つの記事が占有する画面空間などによって決まる情報提示の圧力である。情報提示の圧力が低い場合、ストレスも低いと考えられが、低すぎるとユーザは情報が流れている事に気がつかず提示している意味が薄れてしまうこともある。つまり、情報提示の圧力を適切に調整できるシステムが必要である。情報提示の圧力が丁度良いメディアと言えば、TV やラジオが挙げられる。これらは情報源として強く意識する事無く、視聴者は受動的に情報を受け取ることができ、更に「付けっぱなし」が苦にならない情報提示の圧力であると言える。また、ふと特徴的な情報に気がついた時に、ユーザは情報に注視するだけで、情報の詳細を知ることができる。これらのメディアは「聞き逃す」という短所も存在することはあるものの、重要な情報が繰り返し提示されるという特徴があるため、聞き逃してもある程度補完する事ができる他、繰り返しによってユーザに強い印象を残す効果もある。

これら既存メディアの特徴を踏襲し、フィードブラ

†1 電気通信大学 電気通信学研究科 情報工学専攻
Department of Computer Science, Graduate school of Electro-Communications, The University of Electro-Communications
†2 電気通信大学 電気通信学部 情報工学科
Department of Computer Science, The University of Electro-Communication

⋆1 Resource Description Framework、ウェブ上にある「リソース」を記述するための統一された枠組み

ウザのインタフェースを提案する。第一に、フィードブラウザにおいてフィードはデスクトップ上に常に存在する。また、ユーザに過度のストレスを与えない範囲で受動的に情報を提示するものである。第二に、ユーザは多くの操作をせずにフィードのフィルタリング、及び情報の取捨選択ができるようにする。第三に、特徴的な情報に気がついた時に、注視または簡単な操作をすることよって、シームレス、ストレスレスに詳細な情報取得を可能にする。最後に、何となく眺めることによって、普段の自分なら取得しないような情報との偶発的に出会いがあるという特徴も考慮する。目標とするのは、テレビで見る朝のニュースや電車の中で見る吊り広告のようなさり気なさと、ささやかな主張である。ただし、本研究では、公共のスペースに表示する情報とは異なり、フィードはディスプレイという非常に限られた空間、更に言えば非常にプライベートな空間に情報を表示する事を想定している。そのため、ユーザが情報量や主張の強さを調整できる事、ユーザがある特定の情報について詳細をシームレスに取得できる事を重視する。

以上の点をまとめると、本研究における目標は

(1) フィード購読時のストレス軽減
(2) 持続的な情報の提示
(3) 情報の取捨選択支援
(4) 偶発的な情報との出会いを支援

である。本稿では以上の目標を達成するフィードブラウザのケーススタディとして、ニコニコ動画の配信記事を購読するフィードブラウザ、「Nicolith」の実装を試みた。

2. Nicolith とは

Nicolith はニコニコ動画の RSS で配信される動画記事を「眺める」ことを重視するフィードブラウザである。本稿ではあえて Nicolith をフィードリーダという定義には当てはめず、フィードブラウザという定義を用いる。

2.1 フィードブラウザについて

本稿では、フィードリーダとフィードブラウザを区別する。フィードリーダはユーザが購読しているサイトが配信しているフィードを定期的にクロールし[*1]、情報を収集する。基本的に購読した情報は未読、既読管理等がされる等、テキスト記事を「読む」ことを目的としていると言える。よって、情報収集の作業は受動的であると言えるが、ユーザが「読む」という点においては、ユーザが能動的に作業することが多いシステムであると言える。自動的に情報が増加するので、ユーザが「読む」ことをフィードリーダに強要されているように感じ、プレッシャを感じる場合がある。対してフィードブラウザは収集した情報を「ブラウズ(ざっと眺める)」することを目的としたシステムである。購読しているフィードに厳格な未読、既読管理等をするわけでなく、更にユーザが能動的に操作をしなくても情報を提示し続けるものである。つまり、本稿で言う「効率」とは従来のフィードリーダと単純に比較されるものではない。

2.1.1 ニコニコ動画の RSS 配信記事について

ニコニコ動画では、RSS では載せきれないデータを API を使ってアクセスできるようにすることで、一つの記事に対する情報量を増やしている。ニコニコ動画では、RSS で配信されている記事で取得できる動画 ID を元にして、

- 動画についたコメント
- 動画のサムネイル
- 動画のタグ
- 動画へのコメント回数
- 動画のマイリスト登録数[*2]
- 動画の総再生回数

のデータをニコニコ動画 API を通じて、XML 形式で得る事ができる。Nicolith では配信記事と、これらの API 情報を合わせて一つの「動画記事」として扱う。これらのデータは通常の RSS で配信されている情報のみでは手に入らないデータであり、検索や情報の提示方法の自由度を高めるものである。これらのデータを利用し、キーワード検索を補完する検索手法や、検索結果を判りやすく視覚化することで、ユーザに対してより強力な情報の取捨選択支援を行うことができる。

2.2 Nicolith 外観

図 1 に Nicolith のシステムイメージを示す。 Nicolith は RSS 配信記事を購読すると、図 2 に示した管理用のアイコン（以降管理人）が画面上に現れる。

図 1 システムイメージ

*1 検索エンジンのロボットが各ページの情報を取り込むために、各サイトを巡回すること

*2 ニコニコ動画が提供している、ユーザが気に入った動画を登録しておけるリスト機能を使って登録された回数を指す

図 2 管理用アイコン（管理人）

管理人は RSS で配信される動画記事を自分の周りに呼び寄せる。動画記事はサムネイル表示され、マウスカーソルをサムネイル上に移動すると動画記事のタイトル、投稿者のコメントなどを見る事ができる。動画記事は管理人の周りを指定された方法で移動する。ユーザは管理人を操作する事で動画記事の動き方を変化させたり、整列させたり、必要であれば非表示状態にするなど、多彩な設定ができる。また、一度に表示する情報量はユーザにプレッシャを与えないように制限し、時間毎に徐々に表示する情報は変化する。ユーザは、ハイライトされる動画記事を眺めながら、気になる動画記事が見つかれば、動画記事をクリックする事で動画を再生できる。もし気になる動画記事を逃しても、時間が来て非表示になった動画記事を巻き戻して表示できる他、整列させて動画記事を一覧する事もできるので、気になる情報を逃す事無くアクセス可能である。これらの機能により、ユーザの「気付き」を少ない操作で支援できる。

また、Nicolith で RSS 配信を購読すると、配信される動画記事を全て Nicolith のプレイリストに登録する。Nicolith におけるプレイリストとは登録された動画記事の動画を順番に再生する機能を指す。プレイリストによりユーザが特に操作をしなくても、動画記事の動画を再生し続けることができる。また、プレイリストは一般の CD 視聴などで利用されているお気に入りを登録して作成する方式ではなく、あらかじめ設定されたフィルタを利用して自動的にプレイリストに残す動画記事を選出する。この機能によってユーザが能動的に操作をしなくても、ユーザの趣向に合う動画を「眺める」ことができる。

2.3 評価の視覚化

動画を見て行く上で、面白い動画であるか否かを判断する為の基準として、再生回数やコメント回数などがある。そこでニコニコ動画に投稿された動画の再生回数、コメント回数、マイリスト登録者数、投稿日時などのデータを利用して評価値を算出し、その評価値を基に取得したサムネイル画像の画像サイズや、透明度、動きを変化させる。これにより、動画がどのように評価されているのかが一目で分かるようになる。詳しい評価値の定義と算出方法については 2.4 節で述べる。この検索手法はキーワード検索と組み合わせる事により強力な検索も可能である。組み合わせ検索に関しては 2.8 節で紹介する。

本システムでは評価値を記事に適用する際に、図 3 に示すような評価値が高いほど大きく画像を表示する方式を採用している。また、投稿日時が指定した日時により近いほど不透明度が高くなる設定になっている (e.g. 投稿日時 = 指定日 → 不透明度 100%となる)。

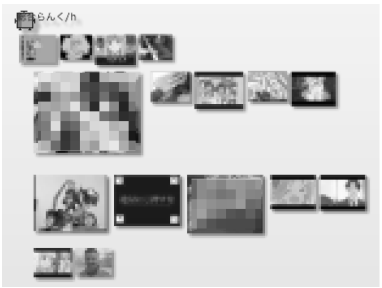

図 3 サムネイルへ評価値を適用した様子

更に、ユーザが記事を表示するか否かを決定する評価値の下限を設定できる機能を導入した。下限以下の画像は非表示になる設定をしている。下限の設定に関してユーザは簡単に設定する事ができるように考慮した。設定方法の詳細については 2.9 節で述べる。

2.4 評価値の調整と算出方法

Nicolith では、動画に登録されている再生回数、コメント回数、マイリスト登録数とユーザが指定した再生回数、コメント回数、マイリスト登録数を用いて評価値を算出する。また、キーワードを用いて評価値を調整する事も可能である。Nicolith では、ユーザが評価値を調整する機構のメタファとして、音響機器の特定の音域の音声信号を強調したり、逆に減少させたりする機構である「イコライザ」を採用した。 図 4 が

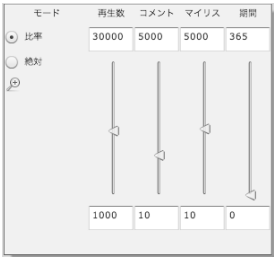

図 4 イコライザの外観

イコライザの外観である。イコライザはそれぞれの評価要素の影響を調整するためのスライダと、スライダの上限、下限を定めるテキストフォームによって構成されている。例えば、図 4 に示した状況は、再生回数：コメント回数：マイリスト登録数の比が、30:3:5 程度の比率である動画が評価値が高くなり、画像が大きく表示される。このように、ユーザは評価値の影響度（e.g. 動画の大きさの変化率や透明度の変化率）を自由に設定する事ができる。

現在の評価値は、スライダで指定した再生回数：コメント回数：マイリスト登録数の比と、動画に登録された再生回数：コメント回数：マイリスト登録数の比が似ているものほど評価値が大きくなるされる方式を利用している。具体的な式としては、動画に登録されている再生回数を P、コメント回数を C、マイリスト登録数を M とし、スライダで指定した再生回数を p、コメント回数を c、マイリスト登録数を m とおく。このとき、比 $r_i(i = 1, 2)$ を以下のように求める。

$$r_1 = \frac{p/P}{c/C}, \quad r_2 = \frac{c/C}{m/M} \tag{1}$$

次に、比 r_i が 1 から離れるほど小さくなるように、以下のような変形を施して r'_i を得る。

$$r'_i = \begin{cases} r_i & (0 < r_i \leq 1) \\ r_i^{-1} & (r_i > 1) \end{cases} \tag{2}$$

最終的に求める評価値 v は以下で与えられる。

$$v = (r'_1 + r'_2)/2 \tag{3}$$

これによって、再生回数：コメント回数：マイリスト登録数の比がスライダで指定した再生回数：コメント回数：マイリスト登録数の比と同じものであれば、単純な数値の大小に影響されずに評価ができる。

2.5 動きの選択

動画記事を眺めることができるシステムで重要なのは、どのように情報が提示され、どのように動いているかである。動くことで人の注意を引き、更に省スペースで情報を表示させる事ができる。パーソナルな空間に常に存在する為に、評価値の算出と同様に、動き方や動く位置はある程度ユーザが自由に選択することができるようにする事で、ストレスを軽減できることは重要であると考えた。現在実装されている動きは、Sort、Card、Line、Swarm である。また、それぞれの設定は図 5 に示した、管理人メニューから調整可能である。

Sort

その場から動かなくなる。管理人から特に指示がある場合は、指示された位置に停止する。

Card

カードをめくるように順に上に移動してく。最もスペースを取らない。

図 5 動きの選択メニュー

Line(Vertical or Horizontal)

図 6 の左に示すように横か縦に規則的に移動する。端まで行くと、反対側へジャンプして繰り返す。

Swarm

図 6 の右に示すように管理人の周りをゆっくりとたかる動きをする。街灯に集まる羽虫のように動く

図 6 動きと配置　左:Line, 右:Swarm

2.6 ハイライト方法の選択

動画記事を表示する事を考えた場合、特定の動画記事をハイライトすることで注目しやすくなる。Nicolith ではユーザが特に指定したキーワードや、イコライザの設定に合うものほど強くハイライトする。現在実装している手法は、Scale, Alpha, Open である。

Scale

サイズを変化させる事によってハイライトする。現在は評価値を表現する方法に利用されている。

Alpha

透明度を変化させる事によってハイライトする。現在は投稿日時を表現する方法に利用されている。

Open

記事の詳細を表示することによってハイライトする。

2.7 シームレスな情報の提供

ふいに飛び込んで来た情報は、ユーザが気になった瞬間にアクセスできる事が望ましい。ユーザは一つの動画記事を表しているサムネイルをクリックするだけで、デスクトップ上で動画記事の動画を再生する事ができる。ブラウザを起動する必要も無く、再生を終了したい場合は即座に終了できる。動画の再生は図 7 に示すように、サムネイルが大きく拡大される形で再生が行われる。もしクリックするタイミングを逃した場合は、管理人を使ってサムネイルをタイル状に並べるか、表示している動画記事を巻き戻して表示させるな

どの操作を行う事で欲しい情報にアクセスする。プレイリスト機能を利用した再生中に、自分が気になる動画が流れている事に気が付かずに自動的に次に登録されている動画記事の動画が再生され始めた場合でも、一つ前の動画記事に戻って動画の再生できる他、逆に流れ始めた動画が自分に取って気に入らないものであれば、すぐに次の動画記事へスキップして動画を再生する事ができる。このようにして「気付き」から「情報の取得」までに要する時間をできる限り削減した。また、Nicolith は再生中の動画をダブルクリックする事でブラウザを起動し、本来のニコニコ動画の再生環境で動画を視聴する事もできる。これによって Nicolith では足りない情報を補完する事が容易にできる。

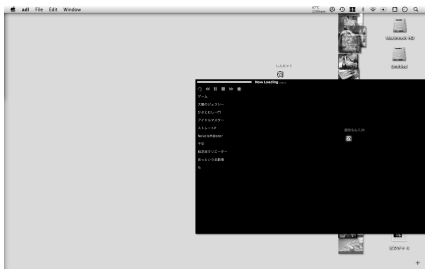

図 7 その場で動画を再生する様子

2.8 キーワード検索とイコライザの連携

Nicolith ではキーワードを用いた検索支援を行う。購読している動画記事を検索するときは、イコライザにキーワード検索用のスライダを追加する。図 8 の右部分に表示されている 2 本のスライダがキーワード検索用である。

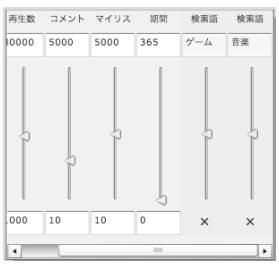

図 8 キーワード検索用のスライダを追加した様子

検索したいキーワードをスライダに入力した後、スライダを調整する事で影響を受けた動画は評価値が変化し、状態が変化する。スライダ値は-100〜100 まで用意されており、プラス方向にスライダを動かすとポジティブな、マイナス方向にスライダを動かすとネガティブな影響を評価値に対して与える。従来のキーワード検索と異なり、特にユーザが指定しない限りはキーワードにヒットしない場合でも動画が非表示になる事は無い。これは情報との偶発的な出会いを支援するためである。

2.9 プレイリスト

Nicolith では、購読中の RSS 配信から取得出来る動画記事全てがプレイリストに追加される。プレイリストとは管理人に登録された動画記事の動画を順番に再生する機能であり、よって特に何も操作しない場合は、プレイリストに登録された動画記事の動画を順次再生する機能を提供する。プレイリストの調整には、スライダやキーワードによってフィルタリングをする方法と、好みの動画記事のサムネイルをドラッグアンドドロップすることによって任意のプレイリストを作成、再生するモードを提供する。

プレイリストのフィルタリング

Nicolith では、2.4 節で述べた評価値をフィルタリングに利用する事で簡単にプレイリストを編集する事ができる。プレイリストに再生するか否かを決定する評価値の基準を与える事によって、基準値以下の評価値を持つ記事をプレイリストから削除することができる。スライダ値の調整にはキーワードも登録する事ができるので、特定のキーワードを持つ動画のみを再生する、再生しないといった高度な設定を行う事ができる。

今回の実装では基準値の選択方法として、ユーザが気に入って再生した動画を一つ選択することで、その動画の評価値を基準値にするという手法を採用した。

これによってユーザは基準値の設定を厳密に行う必要が無くなり、自然な形で設定を行う事ができる。Nicolith ではこれを「動画をピン留めする」という。お気に入りの写真をクリップボードに貼付けて行く様子に似ていることからこのように定義した。図 9 と図 10 は実際にピン留めをする前とした後のサムネイル数の変化を示したものである。また図では判りにくいが、ピン留めされた動画にはピンのアイコンが付き、それが基準の動画であることを明確にする。

図 9 ピン留めによるフィルタリングの様子（前）

図 10 ピン留めによるフィルタリングの様子（後）

お気に入り登録によるプレイリスト管理

登録している RSS から特定のお気に入りだけを再生したい場合は、Bookmark を利用する。Bookmark は動画記事のサムネイルをドラッグアンドドロップすることで動画記事を保持する事ができる。お気に入りとして登録した動画記事の動画もまた、プレイリストとして再生する事ができる。これにより、昔登録した動画記事が一定の割合で提示されることになり、お気に入りの死蔵を減らす事ができる。

3. 実　　装

実装には主に Flex SDK 3 および、ActionScript3.0 を利用した。サービスを提供する方式としては、インタフェースを試してもらう為の Flex バージョンと、常時起動や眺める環境を最適化する為の Adobe AIR で実装したバージョンを用意した。その他、データの管理等に XML や軽量のデータベースである SQLite3 などを利用した。更に Flex 版ではファイルアクセスなどに一部 Ruby で CGI を駆動している。

3.1 bugs フレームワーク

今回 Nicolith の実装には常駐型情報提示システム作成用に開発した「bugs フレームワーク」を利用している。 bugs フレームワークは本研究のベースとして開発したフレームワークであり、RSS や HTML などを情報源として、その情報をデスクトップ上に常時提示することを目的としている。bugs フレームワークは、記事を一つ保持する bugs と、bugs を複数管理する lights が基本となる。

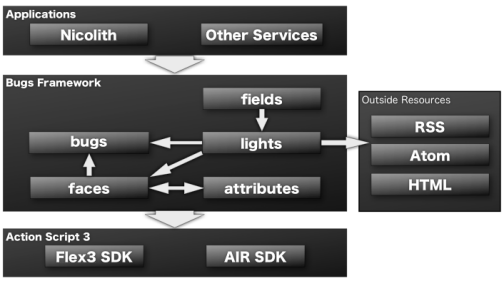

図 11 bugs フレームワーク概観

bugs インスタンスには GUI を実装するクラスである faces のインスタンスが割り当てられる。faces インスタンスの動き方を実装するのが attributes インスタンスであり、faces インスタンスは attributes インスタンスを動的に切り替える事で 2.5 節で述べたような、様々な動きを行う事ができる。

lights はインスタンスが生成されると RSS などの外部リソースを読み込み、一つ一つの記事を解析し、解析した内容を元に faces インスタンスと、bugs インスタンスを生成する機能を持つ。更に生成した faces インスタンスを、ユーザが実際に操作するウィンドウ領域を持つクラスである fields のインスタンスに追加し、ユーザに情報を提示する。また管理下にある bugs インスタンスの attributes インスタンスを変更する機能や、表示、非表示を行う機能が割り当てられている。

Nicolith はこれらのクラスをそれぞれニコニコ動画用に継承して実装したものである。今回は実装上の問題や取得できる情報量の多さからニコニコ動画の RSS を利用したが、bugs フレームワークを利用する事で、ニュースやブログなどの RSS 記事なども取得・表示する事が可能である。

4. 試　　用

著者らの所属する研究室の学生に 1 時間ほど集中的に試用して貰った。実験者（著者の 1 人）は被験者に立ち会い、被験者から質問があった場合は適宜回答をした。更に、実験者から被験者に対して、操作感などの気になる点について、いくつか質問をした。実験終了後被験者にはアンケートを実施し、アンケート結果を元にインタビューを行った。

実験環境の OS には Windows Vista を利用し、画面解像度は横 1440px、縦 900px のワイドディスプレイを利用した。Nicolith は Adobe AIR 版を利用し、ウィンドウ領域は横 700px、縦 900px、ウィンドウの配置はディスプレイ右詰めとした。初期登録されている RSS は新着リスト 10 件及び、総合ランキングの毎時集計結果が上位 30 件である。ユーザはニコニコ動画の RSS 配信を新たに購読することも可能である。また、記事の初期状態の動きは Line タイプの縦スクロールとした。

被験者はニコニコ動画を個人的に利用しており、RSS の仕組みについて理解している者である。利用期間、利用時間、趣味趣向は様々である。

4.1 アンケート

アンケートでは、

(1)　表示している情報量

(2)　付けっぱなしのストレス

(3) 動画を見るための操作数
(4) イコライザの有用性
(5) プレイリストの有用性
等の質問を行った。本節ではそれぞれに関する評価とコメントについて述べる。

第一に、表示する情報量は適切であるとの回答が多かった。回答理由として、画面領域に比べて移動領域とサムネイルサイズが適切であるため、という理由が多数を占めた。また、フィルタリングを行うとむしろ情報量が少なく、物足りないというコメントも得られた。

第二に、付けっぱなしにすることのストレスは殆ど無いという回答が殆どであった。但し、付けっぱなしによって感じるストレスはシチュエーションによって異なるというコメントが得られた。つまり集中力を求める作業中は邪魔になるが、対してリラックスした状態、例えばくつろぎながら Web ブラウズをしている時などは、適度に情報を提示されて良い刺激になるという事である。その他にストレスを感じるのは、Web ブラウザを最大化した際にスクロールバーなどが見えにくくなる場面が挙げられた。

第三に、動画を見る為の操作数に関しては意見が割れた。操作が少ないと感じた被験者は、デスクトップ上に直接情報が提示されることや、プレイリスト機能によって自動的に次の動画を表示するので操作をする必要がないことなどを理由として挙げた。対して操作が多いと感じた被験者からは、普段から細かいキーワードを指定して積極的に検索を行うので、逆に思い通りの動画が見つけにくく操作数が増えてしまったというコメントが得られた。

第四に、イコライザについては比較的良い評価を得られた。インタビューの結果、使いにくいと答えた被験者はニコニコ動画において、どのような動画が、どの程度の再生回数、コメント回数、マイリスト登録数になるか判らない被験者であった。この事が低い評価に繋がったと言える。

最後に、プレイリストについては意見が割れた。次の動画へのスキップ操作を簡単に出来ることを良いと評価する被験者がいる一方で、スキップしても思い通りの動画が再生されず、繰り返しスキップ操作をすることがストレスであると、マイナス評価をした被験者もいる。また、プレイリストフィルタリングについても賛否両論であった。否定派の主な理由としてイコライザの調整が難しく、なかなか思い通りのフィルタリングをできないというコメントが挙げられた。対して、肯定派の理由としては、思いのほかフィルタリングが良くできた、もしくは、思い通りではないが、面白い動画が再生されたので結果的に良かったというコメントが挙げられた。

5. 考　　察

Nicolith は受動型のフィードブラウザとして、方向性は適切であると言える。しかしながら情報の表示量や表示の方法について、個々のユーザからの要望に応えきれていないことも明らかになった。表示量や表示方法に関しては、ユーザがある程度自由に設定し、設定を保存する機能等を実装する事で対応することが必要であると言える。

積極的に動画を検索して視聴する「能動型」のユーザと、ランキングや適当に目についた動画をかいつまんで視聴する「受動型」のユーザとでは、後者の評価が高いことが判った。これは受動型ユーザ向けというコンセプトが正しく実装出来ていると言える。

イコライザやプレイリストなどに関して、コンセプト自体は全体的に好感を得られている事が判った。しかし調整の難しさなど、動画サイトに詳しくないユーザに対して優しくない部分があることを厳しく指摘された。イコライザの設定には、ジャンル毎にお勧めの設定をあらかじめ用意しておくことや、熟達者が作った設定を公開、共有できるようなシステムを用意するなど、初心者に対して優しい環境を構築する事が重要であると言える。

6. 関　　連

Nicolith の発想は一般にデスクトップやタスクバーに常駐し、最新のフィードが取得できた場合にポップアップやテロップなどの形式で通知する方式として知られるティッカー型を採用したフィードリーダと似ている。例えば Pyxis が開発した Ensemble Petit[3] などがある。Nicolith がティッカー型と大きく異なる点は、基本的に記事が常にデスクトップ上に表示されているという点である。更にティッカー型は全ての記事が同等に扱われるのに対し、Nicolith ではユーザの好みや記事の評価によって情報の提示方法も、大きな差異と言える。

眺めるインタフェースとして渡邊らが Memorium[4] を試作している。本研究では Memorium を、眺める・持続的に存在するインタフェースとして参考にしているが、情報のフィルタリングや情報の再生方法、提示方法が異なる。

Goromi-Web[5] や Goromi-TV[6] は、Web の情報や撮り貯めた TV 番組を「気の向くままにブラウズする」為のインタフェースである。これらの研究ではユーザのブラウジング中の移り気を重視しており、自分の定常的な好みに沿う情報とその周辺情報を眺めることを主眼にしている本研究とは異なる。

IKESU[8] はメタファの構成や、検索手法が一部似ているものの、本稿では、データベースが常に変化して行く RSS を扱っている点や、IKESU では情報の収集、整理が目的となっており、本稿とは目的が異なる。

自分の好みの RSS を登録して眺めるシステムとして FeedTV[7] が挙げられる。しかし、本稿のように再生する RSS をフィルタリングしたり、動画の評価等を視覚化していない点が異なる。

膨大なビデオライブラリから、3 種類の視覚化技術を用いてビデオの分布を示し、スライダーを利用して検索条件を変更しながら絞り込みを行う手法を、Christel[9] が提案している。この手法は、Nicolith のスライダ検索方法と比べて、評価値の算出方法がまず異なる。更に評価値を利用したフィルタリング方法や検索結果の提示手法もまた異なる。

7. 展　　望

Nicolith には以下の機能を実装したいと考えている。

7.1 傾向の利用

購読している RSS からホットタグ[⋆1]を、フィールドに配置できるようにする。ユーザが注目するホットタグを配置する事で、ユーザの好みの傾向に合うと予想される動画をハイライトする。動画は配置したタグと、周辺状況を見て動きや大きさを変化させる。具体的な手法としては、タグが近いと画像が大きくなる、動きがゆっくりになるなどの情報の提示方法がある。このような配置を可能にすることで、「どこ」でハイライトされる記事が、自分のどの好みに合う記事なのかを空間的に理解出来るようになると考えている。

7.2 評価値算出方法の見直し

評価値算出方法のコンセプトは固まっているものの、ユーザが心地の良い設定ができるようになるまでに時間がかかるというケースが多く見られた。また、今回の評価値算出はニコニコ動画の再生回数、コメント回数、マイリスト登録数を利用したものなので、youtube や dailymotion の再生回数やコメント回数、評価などを利用した場合はまた異なる評価値算出式が必要になる事は容易に想像できる。これらの定式化は、恐らく動画サイトの利用状況を調査し、その調査結果に基づいて行われるべきであると考えている。

7.3 情報の共有

先に挙げたイコライザの設定ファイルの共有、傾向による動画自動分類設定の共有などを行う。その他、動画以外での利用を考えた場合は、ニュース記事に対するコメント、評価などの共有が考えられる。イコライザの設定ファイルは誰でも公開でき、それを利用して自分好みのイコライザの設定を容易に構築できるようにする。

7.4 他の動画サイトへの対応

提案する情報提示方法や検索手法をニコニコ動画のみでなく他の動画サイトやテキストベースの RSS 配信へ応用することが必要であると考えている。今回作成したフレームワークはフィード配信全般に応用する事を前提に設計されているため、近日中に他のフィードへの応用例を実装し評価したい。

8. 終 わ り に

本研究では、フィードを効率よく眺めるためのフィードブラウザとして、ニコニコ動画用フィードブラウザ: Nicolith を開発した。これにより、ユーザがストレスレスで記事を眺めることができる環境の構築および、配信記事の持続的な提示、イコライザによるキーワード検索に限らない情報の取捨選択を可能にした。しかし、ユーザの細かい要望に応える機能の実装がまだの他、実験によって様々な課題が噴出しているため、それらへの取り組みや、展望で述べた機能の実装に取り組む必要がある。

参 考 文 献

1) ニコニコ動画 http://www.nicovideo.jp/
2) Official Google Blog, http://googleblog.blogspot.com/2008/07/we-knew-web-was-big.html
3) Pyxis(itok,oishi), http://pyxis-project.net/petit/
4) 渡邊 恵太, 安村 道晃: Memorium: 眺めるインタフェースの提案とその試作, WISS2002 論文集, pp.99-104 (2002).
5) G. Otsubo:“Goromi-Web: browsing for unexpected infor- mation on the web”, pp. 267?268 (2007).
6) 大坪五郎:“Goromi-TV”, WISS2006 論文集, pp. 47-52 (2006).
7) “FeedTV `http://feed-tv.com/`”.
8) 石井 隆昭, 望月 有人, 星野 剛史, 堀井 洋一: IKESU :「旬」な音楽を聴くための収集型ミュージックプレーヤー, WISS2005 論文集, pp.25-30 (2005)
9) M. G. Christel. Accessing News Video Libraries through Dynamic Information Extraction, Summarization, and Visualization. In Visual Interfaces to Digital Libraries, pp. 98-115. SpringerVerlag, January 2002, vol. 274, (2002)

⋆1 ユーザが購読している RSS や記事に利用されている頻出タグを抽出したもの