# Generating Associative Ripples of Relevant Information from a Variety of Data Streams by Throwing a Heuristic Stone

Xiaokang Zhou
Graduate School of Human Sciences,
Waseda University
2-579-15 Mikajima, Tokorozawa-shi,
Saitama, Japan
xkzhou@ruri.waseda.jp

Hong Chen
Graduate School of Human Sciences,
Waseda University
2-579-15 Mikajima, Tokorozawa-shi,
Saitama, Japan
chen@fuji.waseda.jp

Qun Jin
Graduate School of Human Sciences,
Waseda University
2-579-15 Mikajima, Tokorozawa-shi,
Saitama, Japan
jin@waseda.jp

Jianming Yong
School of Information Systems,
University of Southern Queensland,
Toowoomba, Queensland,
Australia
Jianming.Yong@usq.edu.au

## ABSTRACT
Recently, the vast dialog in the microblog, such as twitter, Facebook has become increasingly popular. As we post more messages in microblogs, information is spreading more quickly and widely. These widely spreaded and diversified contents could be viewed as data streams, which have become an important part of the Internet resources. However, these separated data streams are littery and meaningless, so we need to collect and organize them together to provide us with meaningful information. It is hard to imagine that we could find useful information by simply inputting a few keywords into a search engine in such a stream environment. In this study, we try to find a way to seek the information related to users' personal and current interests and needs among these data streams and provide users with other more relevant information. We introduce a set of metaphors to represent a variety of data streams in different levels, and define two new metaphors: heuristic stone and associative ripple to assist the seeking process and describe the results. Based on these, we further propose two algorithms for the information seeking and processing, and discuss a scenario of the information seeking process that utilizes the proposed metaphors and algorithms.

## Categories and Subject Descriptors
H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval – *clustering, retrieval models, search process.*

## General Terms
Algorithms

## Keywords
data stream, stream metaphor, microblog, data clustering, information seeking

## 1. INTRODUCTION
The Web now is providing us with a variety of personal services. It has become increasingly easier for us to post personal contents in this widespread social network. That is, we are being around with wikis, the vast dialog that is the blogosphere, social bookmarking, online networking, etc. For example, with twitter we can obtain fresh news more quickly by means of following, forward information to other followers by retweeting (RT), and discuss about a topic by adding a "#" (hash tag).

Obviously, when we add more friends in twitter or Facebook, the amount of messages that we see in the screen will become bigger than before. For most of the web sites feed contents with the RDF/RSS/Atom standards, we can read news coming from different web sites and blogs by RSS readers. News that continuously comes from different web sites looks like a stream which has become an important part of the Internet resources.

All these mentioned above make information spread more widely. It is hard to imagine that finding useful information just means inputting a few keywords into a search engine. Sites no longer change in weeks or days, but hours, minutes or even seconds. For instance, if more people are followed in twitter, more messages show up on the timeline. It is impossible for a person to follow such a big amount of information. Moreover, there are lots of noise messages and duplicated messages among the data streams. Therefore, seeking high-quality information for users from data streams has become increasingly important.

Some services which analyze the popularity trends of a whole community have been developed based on data streams in twitter. However, in most of the time, what we really concern are the trends that are related to an individual rather than the whole community. So it is critical to seek information referring to users' current and personal interests or needs.

In our previous study, we proposed and defined metaphors to represent a variety of data streams in order to categorize the stream data and organize the information schema. In this study, we try to seek information related to users' personal and current needs in data streams. We define two new metaphors: Heuristic Stone and Associative Ripple to assist the seeking process and describe the results. We further propose two algorithms which are used to catch "ripples" in the "river" to achieve the seeking purpose.

The rest of this paper is organized as follows. We give a brief overview on the related issues and works with data streams in Section 2. In Section 3, we first introduce several metaphors for data streams in order to organize the information schema. Then we propose two new metaphors to describe the seeking process and result. After that, two algorithms are proposed to serve for the seeking purpose. In Section 4, a scenario of the seeking process is described, and some seeking results are given based on it. We conclude this study and give some promising perspectives on future works in Section 5.

## 2. RELATED WORK

Research works [1-6] have been tried to make use of microblog-generated stream data to create Social Semantic Microblogs, or use Semantic Webs to link and reuse data across Web 2.0 communities. J.G. Breslin et al present the SIOC (Semantically Interlinked Online Community) ontology which combines terms from vocabularies that already exist with new terms that are to be described on the relationships between concepts in the realm of online community sites [1]. Uldis Bojars et al. use the Semantic Web to link and reuse data across Web 2.0 communities [2]. Studies have also been tried to create a prototype for distributed semantic microblogging [3]. Wolfgang Reinhardt et al. tried to use microblog to enhance the knowledge of a given group or community by micro-connecting a diverse online audience [4]. Martin Ebner et al. indicated microblogging should be seen as a completely new form of communication that can support informal learning beyond classrooms [5]. SMOB (Semantic-MicrOBlogging) is a platform for open, semantic and distributed microblogging combining Social Web principles and state-of-the-art Semantic Web and Linked Data technologies [6].

Research works also been tried on Data Streams Mining [7], such as clustering, classification, frequency counting and time series analysis techniques. A host of algorithms have been proposed for extracting knowledge from streaming information. Aggarwal et al. [8] have proposed a framework for clustering data steams called CluStream algorithm. The proposed technique divides the clustering process into two components. Guha et al. [9, 10] have studied analytically clustering data streams using K-median technique. Ordonez [11] has proposed several improvements to k-means algorithm to cluster binary data streams.

Stream reasoning, which was developed by E. Della Valle et al. [12, 13] is a new multidisciplinary approach for semantically processing high-frequency high-volume streams of information in combination with rich background knowledge.

As discussed above, we propose and develop a new way to assist information seeking based on the metaphors introduced in this study, and by utilizing the streams clustering method as well, in order to best fit users' current interests and needs in such a stream environment, which could be seen as catching the "ripples" in the "river".

## 3. GENERATING ASSOCIATIVE RIPPLES OF RELEVANT INFORMATION: METAPHORS AND ALGORITHMS

To seeking information for users, in this section, we first introduce a set of metaphors to represent data streams in different levels. We further define two new metaphors: heuristic stone and associative ripple. We propose two algorithms to generate associative ripples of relevant information from a variety of data streams by throwing a heuristic stone (a specific keyword), and show how to catch the information to serve a specific user using these metaphors and algorithms.

### 3.1 Metaphors for Streams

To seek information for a specific user, we introduce the metaphors for data streams as follows [14].

- **Drop**: Drop is a minimum unit of data streams, such as a message posted to the microglog (e.g., Twitter) by a user, or a status change in SNS (e.g., FaceBook).
- **Stream:** Stream is a collection of drops in timeline, which contains the messages, activities and actions of a user.
- **River**: River is a confluence of streams from different users which are formed by following or subscribing his/her followers/friends. It could be extended to followers' followers.
- **Ocean**: Ocean is a combination of all the streams.

As mentioned above, message posted from every users can be seen as a drop, and the drops come from one user converge together to form a stream. Then the streams of the user and his friends form the river. Finally, all the streams come together to form the ocean. All these metaphors are shown in Figure 1.

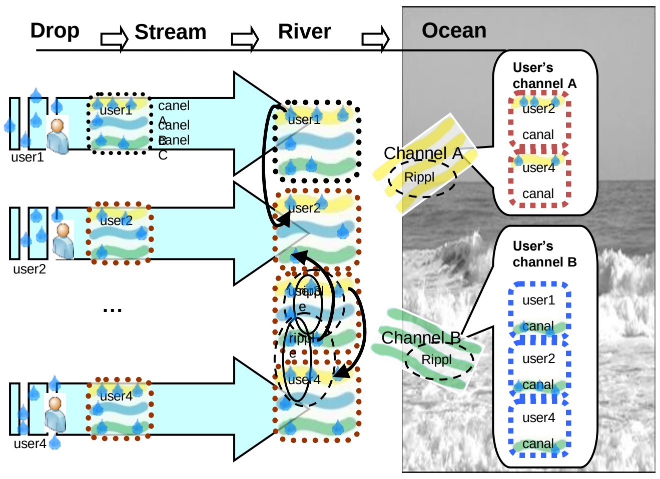

**Figure 1. Metaphors for data streams**

The following definitions are used to seek information that satisfies users' current needs. Differing with the definition of ripple in [14] which is formed naturally in the river, the ripple defined in this paper is formed artificially in an associative way, which is called an associative ripple, and produced by throwing a heuristic stone into the river.

**Heuristic Stone**: it represents one of a specific user's current interests which may be changed dynamically.

**Associative Ripple**: it is a meaningfully associated collection of the drops related to some topics of a specific user's interests, which are formed by the heuristic stone in the river.

## 3.2 Collecting Heuristic Stones

Before seeking for the ripple, we should collect the heuristic stone first.

As defined above, a heuristic stone represents a user's current interest. We discover a specific user's interests from his/her streams and the streams of users that he/she is following, using the clustering method. And the method used to decide whether a data should be clustered to a group is based on keyword matching methods, such as the one introduced in [15]. We do not concern the result of clustering but the cluster centers. That means the elements in each cluster are not important in this study. We just use the cluster centers to catch a user's current interests.

Before introducing the algorithm for catching the user's interests, the sample space should be defined as a quadplex tuple ($Z_s$, $G_s$, Q, $C_s$ ):

$Z_s = \{Z_1, Z_2, Z_3, \ldots, Z_n\}$: a non-empty set of input data, which consist of messages posted by the user and messages in his favorites.

$G_s = \{G_1, G_2, G_3, \ldots, G_m\}$: a non-empty set of the final clusters. $G_i$ consists of a series of $Z_i$.

$Q(Z_s, C_s) \rightarrow Z_n \in G_m$ : a matching function which is used to decide whether $Z_i$ belongs to $G_i$ or creates a new cluster $G_r$ to contain $Z_i$.

$C_s = \{C_1, C_2, C_3, \ldots, C_m\}$: a non-empty set of the cluster centers.

The clustering algorithm is shown in Figure 2, and described as follows.

For $Z_s = \{Z_1, Z_2, Z_3, \ldots, Z_n\}$

(1) Take any $Z_i$ of $Z_s$, for instance $Z_1$, create $G_1 = \{Z_1\}$ and the cluster center $C_1$ of $G_1$.
(2) Compare the remaining elements in $Z_s$ for example $Z_2$ with $C_1$ by function Q, if the comparing result is less than a specific value; add $Z_2$ into $G_1$, so $G_1 = \{Z_1, Z_2\}$, else if the comparing result is higher than the value; create $G_2 = \{Z_2\}$ and the cluster center $C_2$ of $G_2$.
(3) Repeat the process; use function Q to compare the relevance between the remaining $Z_i$ in $Z_s$ and each $C_i$; if satisfying the condition mentioned above, add it into $G_i$; else create a new $G_R$ and $C_R$ until all the elements in $Z_s$ is assigned into a $G_i$.
(4) Collect all the cluster center $C_i$, so that $C_s = \{C_1, C_2, C_3, \ldots, C_m\}$ is what we need.

To guarantee the quality of the clusters as well as the heuristic stones, in this process, it is crucial to evaluate the keywords that are extracted from each message posted by the user and messages in his favorites as well. The well-known feature selection method TF-IDF (Term Frequency - Inverse Documentation Frequency) has been widely applied in information retrieval field. The main idea of this method is that if a term appears in a document with a high frequency, and it rarely occurs in other documents, then that term has good discrimination among these categories.

```
input: the sample {Z1, Z2, Z3, …, Zn}
output: the set of cluster centers Γ: {C1, C2, C3, …, Cm}
Begin
  Γ=Φ; //the initial center set is empty
  SamSpace[ ] ; //for storing the sample data
  Gcluster[ ]; // for storing the clusters
  Ccenter[ ]; // for storing the centers
  Queue Q; //create a queue
  InitS();//initiate SamSpace[ ]
  Q.insert(Q, SamSpace[ ]) //initiate the queue
  While(!Q.isEmpty())   //cluster
      {
        index = findData(Q.peek());
        if (Gcluster.isEmpty()) //initiation
            {
              Gcluster.addG(Gcluster[0], index);
              Ccenter.createC(Ccenter[0], Gcluster[0])
              }
        if(!Gcluster.isEmpty())
              {
              int [ ] value ;
              int j;
              for(int i = 0; i < Gcluster.getsize(); i++)
              //calculate the relevancy
                {
                value[i]=compareG(Ccenter[i], index));
                }
              for(int i=0, j = value[i]; i< value.getsize()-1; i++)
              //get the most related group
                {
                if (j>value[i+1])
                j = value[i+1];
                }
              if (j)
              //if exist the matched group, add the data into this group
              Gcluster.addG(Gcluster[j], index);
              else//else, create a new group and the center
              {
              Gcluster.addG(Gcluster[Gcluster.getsize()], index);
              Ccenter.createC(Ccenter[Ccenter.getsize()],
Gcluster[Gcluster.getsize()])
              }
              Γ=Γ∪{Ccenter};//collect the centers
            }
              Q.delete();
      }
  clean();//free the space
  output Γ; //out put the results
End
```

**Figure 2. Algorithm for collecting heuristic stones**

The common formulas are given as follows:

$$W_i = TF(t_i, d) * IDF(t_i) \tag{1}$$

$$IDF(t) = \log \frac{|D|}{DF(t)} \tag{2}$$

From the past experiments, TF-IDF method has achieved good results in keywords discrimination when the document contains more than 200 words. But as we know, every messages posted in twitter should be less than 140 characters. So we should improve the common TF-IDF formula to adapt to our calculation of the keywords weight. We assume that all users have filled their profiles, therefore, we use users' information filled in their profiles to create some categories. When we calculate the weights of every term in users' messages, we first employ the TF-IDF to calculate a weight. Then if this term belongs to some categories that we have created before, we will add an additional weight to it. Finally, if the total weight of this term is more than a specific value, it will become a keyword. Based on these, the improved TF-IDF to extract the keywords is described as follows:

$$W_i = TF(t_i, m) * \log \frac{|M|}{MF(t_i)} + \sum_{j=0}^{n} t_i * C_j \tag{3}$$

where, $$t_i * C_j = \begin{cases} 1, & t_i \epsilon C_j \\ 0, & \text{else} \end{cases}$$

In this formula, $W_i$ is the total weight of the word $t_i$ in message m, $TF(t_i, m)$ is the number of times word $t_i$ occurs in message m. M is the total number of messages. $MF(t_i)$ is the number of messages in which the word $t_i$ occurs at least one time. $\sum_{j=0}^{n} t_i * C_j$ is an additional weight, that is, if $t_i$ belongs to a category which we have created, it will get one additional weight. The more categories it belongs to, the more additional weights it will get.

After all the weights of words in messages have been calculated, we will select those words with higher quality to become the keywords, using a specific threshold by which those words with lower weight can be filtered. And this can also guarantee the quality of the following clusters and the clusters' centers.

VSM (Vector Space Model) is a widely employed model in information retrieval field these days. The main idea is that: assume that words are not related with each other, so that an algebraic model for representing text documents (and any objects, in general) as vectors of identifiers can be built to simplify the complexity relationship between keywords of the text.

Based on these, we can build the VSM to calculate the similarities between each message, which are then used in the clustering. Therefore, in $Z_s$, each $Z_i$ is described like this:

$$Z_i = ((t_1, w_1), (t_2, w_2), \ldots (t_n, w_n)) \tag{4}$$

where $t_i$ means the keyword in this message, and $w_i$ is its weight. And we use the cosine method to calculate the similarity. The formula is described as the following:

$$sim(M_i, M_j) = (\sum_{t=1}^{n} M_{i,t} * M_{j,t}) / (\sqrt{\sum_{t=1}^{n} M_{i,t}} * \sqrt{\sum_{t=1}^{n} M_{j,t}}) \tag{5}$$

In this formula, $M_i$ is the message feature vector, $M_{i,t}$ is the t vector in message $M_i$.

After that, we cluster the messages posted by the user and messages in his favorites based on these similarities and finally get the cluster centers as the heuristic stones.

## 3.3 Generating Associative Ripples

Now we use the heuristic stone to generate the associative ripples, just as we throw a stone into a river in which ripples then will emerge. The process is similar to the clustering. Each heuristic stone in the river could be seen as a cluster center. When we throw it into the river, the drops which related to this center in the river will converge to it. We use the distance from the drop to the center to describe the relevance between them. And the drops which have the same relevance to the center will form a circle.

When we throw the heuristic stone into a river, it may generate a series of ripples, which depends on the timeline and the granularity of the user's interest. We divide the whole timeline into several time slices in which a cluster center will be produced by the heuristic stone and then a ripple will be generated. Figure 3 shows these ripples in every time slices.

In Figure 3, the "+" indicates the cluster center of each ripple, and the "+" with several circles around it compose a ripple. Obviously, there are 4 ripples in this figure, while each ripple belongs to a time slice. For example, the ripple $R_1$ belongs to slice1. All these four ripples are generated by one heuristic stone, which compose a ripple sequence. The drops in the circle mean they are clustered to a ripple in some degrees, while others are not. And we should notice that at the beginning, the drops from different users distribute in the river following the timeline. However, after the clustering, the time sequence is broken, so that they do not follow the timeline any more in each ripple. We only show the relevancy between the drops and the cluster center in these ripples. We can further show the ripples in sequence, in which way they still follow the timeline.

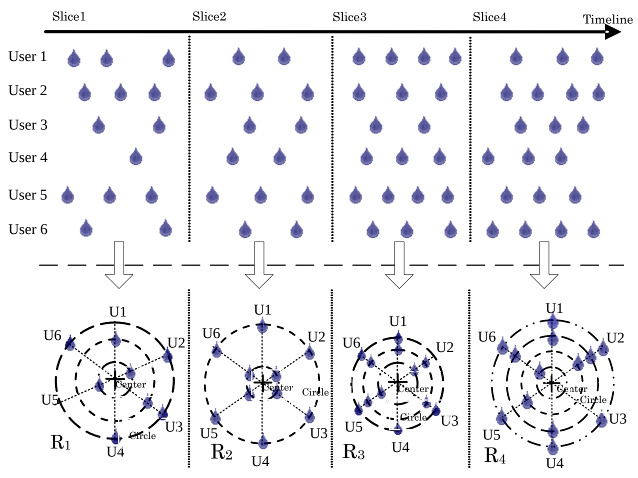

**Figure 3. Associative ripples generated by a heuristic stone**

The ripple consists of several circles, which divide the drops converging to the center into different scales of the radius. This means the closer to the center, the higher relevance to the user's interest. The whole ripple is also divided into several parts depending on the amount of users, in which drops related to this ripple from different users are distributing. A series of ripples which are generated by one heuristic stone could be seen as a group of topics around a user's interest indicated by that heuristic stone, which then follow the timeline to come into a sequence.

From Figure 3, we can see

- Different ripples may have different sizes. The size of the ripple does not relate to the amount of the information in it. This means a small ripple may contains more information than a big one. For example, ripple $R_2$ is bigger than $R_3$, but it contains less information. We do not use size but density to assess the amount of information in each ripple even each circle.
- Different ripples may contain different amount of circles. And the sizes of circles from inside to outside do not follow a unified measure. For example, the most inner circle of ripple $R_3$ and $R_4$ do not have the same size. The circle closer to the center only means it contains more relative information. We use the distance, i.e. the radius of each circle to assess the relevance of the information and the center interest.
- We don't cluster all the drops in the river, because not all of them are related to a specific user's interest. Those drops which have not been clustered could be seen as noises in this process. We can find in Figure 3 that the amount of the drops that have been clustered in these ripples is less than the whole amount in the river.

For each ripple, we recommend the information by rank to the user. As we can record users who post the drops, we can even recommend this specific user to follow the user who posts the most or the user who posts the most relevant information.

For the whole sequence of ripples, we can not only send these information clusters in a specific order to the user, for example by timeline, but also can let the user know that in which time slice people concern more about his/her interest in some degrees, for instance, quantity or quality.

We can also use a sextuplet ($Z_s$, $H_s$, $r_x$, $R_s$, $G_s$, Q) to describe those discussed above:

$Z_s = \{Z_1, Z_2, Z_3, \ldots, Z_n\}$: a non-empty set of input data, which consist of messages posted by all the users in this community.

$H_s = \{h_1, h_2, h_3, \ldots, h_m\}$: a non-empty set of the heuristic stones, which are generated by the algorithm shown in Figure 2. Each $h_i$ will generate a series of ripples.

$r_x = \langle Z_1, Z_2, Z_3, \ldots, Z_t\rangle$: a non-empty sequence of messages clustering to a center, which follows the relevance from the closest to the furthest in sequence.

$R_s = \langle r_1, r_2, r_3, \ldots, r_x\rangle$: a non-empty sequence of ripples produced by one heuristic stone, which follows the timeline in sequence.

$G_s = \{R_1, R_2, R_3, \ldots, R_m\}$: a non-empty set of $R_s$, which is the final cluster results

$Q(Z_i, H_i) \rightarrow Z_n \in r_x$: a matching function which is used to decide whether $Z_i$ belongs to $r_x$ and the distance to the center of $r_x$.

```
input: the sample {Z_1, Z_2, Z_3, …, Z_n}
output: the set of clusters Γ: {R_1, R_2, R_3, …, R_m}
Begin
  Γ=Φ; //the initial set of clusters is empty
  SamSpace[ ] ; //for storing the sample data
  HS[ ] ; // for storing the set of heuristic stones
  Ripple[ ]; // for storing the ripple in one time slice
  Ripples[ ] [ ];
  // for storing the ripples caused by one heuristic stone
  Queue Q; //create a queue
  SamTimeslice [ ] [ ] = InitS(SamSpace[ ], timeline);
  //initiate SamSpace[ ] into several time slice by the timeline
  Q.insert(Q, HS[ ]) //initiate the queue
  While(!Q.isEmpty())   //cluster
     {
        index = findData(Q.peek());
        for (int i = 0; i < SamTimeslice.getsize( ); i++)
        //generate ripples in each time slice
        {
        for (int j = 0; j < SamTimeslice[i].getsize( ); j++)
        //calculate the relevancy between the center and the data
        {
        value[j] = compareR(SamTimeslice[i] [j], index);
        }
        for(int s=0; s< value.getsize(); s++)
        //cluster the satisfied data into the ripple and sort them by the value
        {
        if (value[s])
        Ripple.addbyV(value[s]);
        }
        Ripples.addbyT(Ripple[i]);
        //collect the ripples caused by one heuristic stone
        }
        Γ=Γ∪{Ripples};//collect the ripples
        Q.delete();
     }
  clean();//free the space
  output Γ; //out put the results
End
```

**Figure 4. Algorithm for generating associative ripples**

The algorithm used to generate ripples is shown in Figure 4, and described as follows.

(1) Divide sample space $Z_s$ into several group, each group belongs to a time slice.

(2) For each $h_i$, generate a cluster center in each time slice to form a ripple.

(3) In a time slice, use the function Q to compare the $Z_i$ with the cluster center, then use the comparing result to decide whether $Z_i$ should be clustered to the center. If the result value is higher than the degree, $Z_i$ will cluster

into the ripple and be given a relevancy value. Each time slice will contain a $r_x$.

(4) Collect all the $r_x$ generated by one $h_i$ into a sequence by timeline to form the $R_s$.

(5) Collect all the $R_i$, so that the $G_s = \{R_1, R_2, R_3, \ldots, R_m\}$ is the final cluster result.

In this process, we use another improved TF-IDF method to extract the keywords. That is, we use the keywords and the weights calculated in the last clustering calculation which is used to extract the heuristic stones to plot a dictionary liked table. Therefore, in this keywords extraction, if the word can be found in that table, it will get an additional weight, because this word may be more related to the user. The following is the formula:

$$W_i = LW_i * TF(t_i, m) * \log \frac{|M|}{MF(t_i)} \quad (6)$$

In this formula, $LW_i$ is the additional weight.

In the clustering process, we still use the cosine method to calculate the similarity.

## 4. SCENARIO

Take a group of people who all use twitter to communicate with each other for example. In this case, users follow each other to form a small community. Messages posted by a user could be seen as drops which join together into a stream. Streams from these users then converge together into a river. Figure 5 shows the drops from users distribute in the river before clustering. We use different colors to show the drops come from different users. Take a user in this community for instance. We take the messages he/she sent last month and the messages he/she collected in his/her favorites to build the data set in which we collect his/her interests as heuristic stones in an associative way. After the interest clustering, all the messages are divided into several clusters. For example, two cluster centers "NBA" and "soccer" are formed, as shown in Figure 6. This means we have caught two heuristic stones--"NBA" and "soccer".

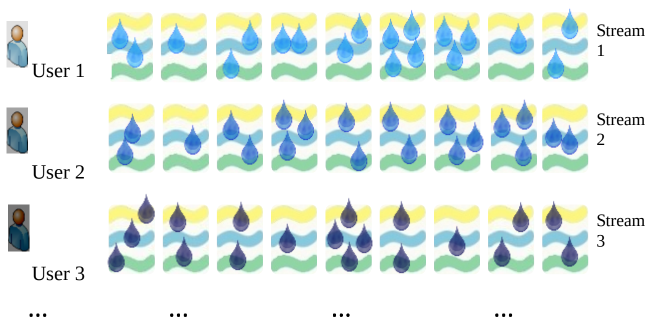

**Figure 5. Drops in a river before clustering**

Then we throw these two "stones" into the river to spark the clustering, which attracts the drops related to them to converge together and finally generates the ripples. Figure 6 shows this process. The "NBA" generates one ripple, and the "soccer" generates several ripples. Each interest center of the ripple in different time slices may have different names such as "UEFA Champions League", "World Cup", "Transfer Market" and so on.

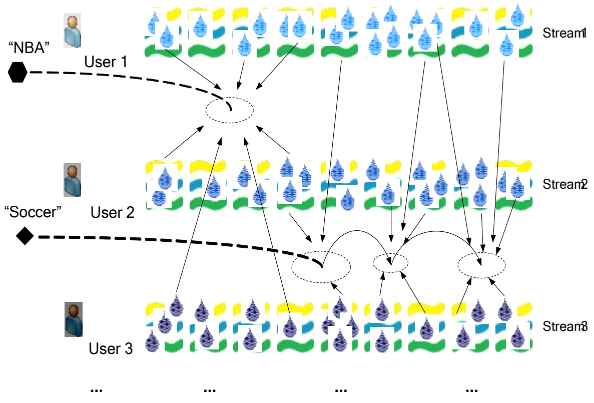

**Figure 6. An example of generating associative ripples**

Figure 7 shows the result after the clustering. In this figure, $R_1$ is the ripple generated by "NBA". $R_2$, $R_3$, $R_4$ are a series of ripples generated by "soccer", which distribute in the river following the timeline. The drops clustered to the cluster center in different circles, represent the messages that are related to the corresponding interest center in various degrees, while others are not clustered.

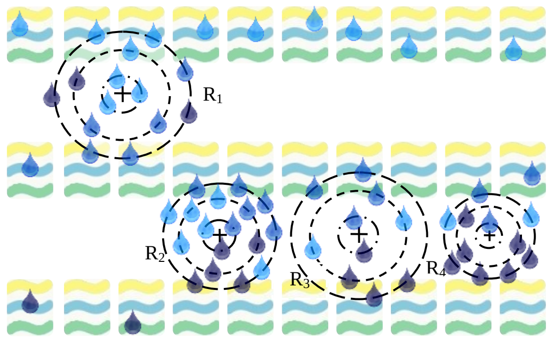

**Figure 7. Drops in a river after clustering**

Based on these, we show these clustered information as well as the relevancy between them and the interest center to the specific user. Moreover, in this case, we can find in Ripple 1 that, messages posted by User 1 compose the closest drops to the center. Therefore, we recommend User 1 to this specific user for this interest. We can find in Ripple 4 that, messages posted by User 3 compose the biggest number of drops around the center. We recommend User 4 to the user. We can also come to a result from the ripples caused by "soccer" that users in this community concern more about "soccer" during the ripple $R_2$'s time slice because $R_2$ contains more information.

## 5. CONCLUSIONS

In this paper, we have proposed a method for users to seek information related to his/her current interests in the stream environment, such as in twitter.

We have introduced a set of metaphors for data streams to organize the information schema. We have further defined two new metaphors: heuristic stone and associative ripple to assist users' information seeking that best fits users' current needs and interests. Based on these, we have proposed a seeking method which uses heuristic stones to generate associative ripples of relevant information and collect them from a variety of data streams to provide users with meaningfully organized information. We have also developed two algorithms for collecting heuristic stones and generating associative ripples. Finally, we showed a scenario to demonstrate how to catch the associative ripples and get useful information from them.

As for future work, a prototype system with improved algorithms will be implemented. In addition, we will consider other elements which may also influence the relevancy, for example time, because more recent message may be more relative than earlier one in general. In this paper, we only considered the situation in the "river" level. In our future work, we will investigate the situation in the "ocean" level, since the information scale will become much bigger than that in the "river" level. We will further consider more efficient approaches for information seeking and avoid the problem such as combinatorial explosion.

## 6. ACKNOWLEDGMENTS

The work has been partly supported by 2010-2012 Waseda University Advanced Research Center for Human Sciences Project "Distributed Collaborative Knowledge Information Sharing Systems for Growing Campus."

## 7. REFERENCES

[1] Breslin J.G., et. al. Towards semantically interlinked online communities. In *Proceedings of ESWC2005,* Heraklion, Greece, 500-514, 2005.

[2] Bojars, U, et. al. Using the Semantic Web for Linking and Reusing Data Across Web 2.0 Communities. *The Journal of Web Semantics, 6,* 1 (Feb. 2008), 21-28.

[3] Passant, A., et. al. Micro-blogging: A semantic web and distributed approach. In *Proceedings of ESWC/SFSW2008,* Tenerife, Spain, 2008.

[4] Reinhardt, W., et. al. How people are using Twitter during conferences. In *Proceedings of 5th EduMedia conference,* Salzburg，2009, 145–156.

[5] Ebner, M., et al. Microblogs in higher education—a chance to facilitate informal and process-oriented learning? *Computers & Education, 55,* 1 (2010), 92–100,.

[6] Passant A., et. al. An Overview of SMOB 2: Open, Semantic and Distributed Micro-blogging. In *Proceedings of AAAI/ICWSM 2010.*

[7] Mohamed Medhat Gaber, Arkady Zaslavsky and Shonali Krishnaswamy. Mining data streams: a review. *ACM SIGMOD Record archive, 34,* 2(Jun. 2005), 18 - 26.

[8] C. Aggarwal, J. Han, J. Wang, P. S. Yu, A Framework for Clustering Evolving Data Streams. In *Proceedings of 2003 Int. Conf. on Very Large Data Bases,* Berlin, Germany, Sept. 2003, 81-92.

[9] S. Guha, N. Mishra, R. Motwani, and L.O'Callaghan. *Clustering data streams.* In *Proceedings of the Annual Symposium on Foundations of Computer Science.* IEEE, California, Nov. 2000.

[10] S. Guha, A. Meyerson, N. Mishra, R. Motwani, and L. O'Callaghan, Clustering Data Streams: Theory and Practice *TKDE special issue on clustering, 15,* 3(MAY./JUN. 2003), 515-528.

[11] C. Ordonez. Clustering Binary Data Streams with K-means In *Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*, ACM, San Diego, California, 2003, 12-19.

[12] Della Valle, E., et al. A First Step towards Stream Reasoning, *In Proceedings of Future Internet Symp. (FIS2008),* Cardiff, May.2008, 72–81.

[13] Della Valle, E., et al. It's a Streaming World! Reasoning upon Rapidly Changing Information. *IEEE Intelligent Systems, 24,* 6(Nov. /Dec. 2009), 83–89.

[14] Chen H., Zhou X.K., Man H.F., Wu Y., Ahmed A.U. and Jin Q., A Framework of Organic Streams: Integrating Dynamically Diversified Contents into Ubiquitous Personal Study, In *Proceedings of the Second International Symposium on Multidisciplinary Emerging Networks and Systems,* Xi'an, Oct. 2010, to appear.

[15] A Kanaegami, K Koike, H Taki, H Ohgashi, *Text Search System for Locating on the Basis of Keyword Matching and Keyword Relationship Matching,* US Patent 5,297,039, 1994, Google Patents