# Implementing Personalized Web News Delivery Service Using Tales of Familiar Framework

Kentaro Noda, Yoshihiro Wada, Sachio Saiki, Masahide Nakamura
*Graduate School of System Informatics, Kobe University*
*1-1 Rokkodai-cho, Nada, Kobe, 657-0011, JAPAN*
*Email: {noda, wada, sachio, masa-n}@ws.cs.kobe-u.ac.jp*

Kiyoshi Yasuda
*Chiba Rosai Hospital*
*Ichihara, Chiba, JAPAN*
*Email: fwkk5911@mb.infoweb.ne.jp*

***Abstract*—We have previously proposed the framework of *Tales of Familiar (ToF)*, where an agent (called *familiar*) autonomously delivers information from various data streams as exclusively personalized *tales* for individual users. Based on the ToF framework, this paper implements a news delivery service, where a stuffed doll (as a familiar) tells a user the latest and personally selected news headlines, by matching user's interests with Web news resources. In the implementation, we especially address three challenges: duplication of tales, value estimation of tales, and delivery timing of tales. We deploy the service in an actual household. The empirical result shows that the subject felt it useful that the familiar pushed his interesting news, automatically. We also evaluate how much the developed service was able to cover the technical issues.**

## 1. Introduction

The emerging IoT, edge and cloud computing technologies enable to gather dynamic and real-time data from cyber and physical worlds. Likewise many other countries, the Japanese Government aims to achieve the next-generation society (called *Society 5.0* [1]) that makes full use of the big data to create highly personalized and value-added services.

Within such floods of big data, it is important for us, human beings, to consider how to *receive* the information wisely for individual life. Although research and development are under way over the world, general citizens are yet far behind such advanced societies. For this, we have been focusing the following complexity to be abbreviated:

**C1 (Access Complexity)** A user has to access data by oneself proactively with complex devices (PC or smartphone).
**C2 (Identification Complexity)** The size and variety of data are so huge that a user cannot find useful information.
**C3 (Interpretation Complexity)** Heterogeneous data in various representations are not intuitive at all for a user.

To reduce these complexity, we have proposed a concept of *Tales of Familiar (ToF)* [2]. ToF is an information delivery service that collects information from various data streams, and delivers relevant information to individuals. The information is delivered to a user in such a way that an agent called *familiar* speaks *tales* to the user.

Figure 1 shows the concept of ToF. Every user has a familiar as his exclusive agent. First, the user registers his personal information to the familiar. From various data sources in the Internet, the ToF service continuously creates tales by converting data into stories in familiar's spoken language. From globally generated tales, the familiar picks up relevant tales, and speaks the tales to the user. Thus, ToF aims to reduce complexity C1, C2, and C3, by autonomously pushing relevant information as user-friendly stories.

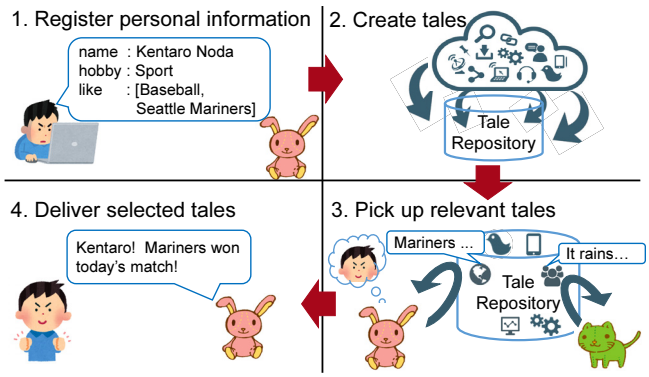


Figure 1. Concept of Tales of Familiar

In [2], we proposed the framework of ToF, which defines core components and basic data schema organizing ToF services. However, the framework itself does not specify details of each component. Concrete implementation is left for individual ToF services.

In this paper, we implement a personalized news delivery service based on the proposed ToF framework. The service generates tales from Web news headlines described in RSS feeds. Matching user's interests with words in every headline, the service then selects tales especially relevant to the user. Finally, a familiar, implemented by a stuffed doll with a wireless speaker, speaks the selected tales a user.

In the implementation, we especially address the following technical challenges: (T1) duplication of tales, (T2) value estimation of tales, and (T3) delivery timing of tales. T1 is a problem that almost the same tales from different news channels can be delivered many times. We exploit the cosine similarity to eliminate such duplication of tales. T2 is how to estimate the importance of every tale towards individual interests. We show the conventional topic-based

978-1-5386-3227-7/18/$31.00 ©2018 IEEE

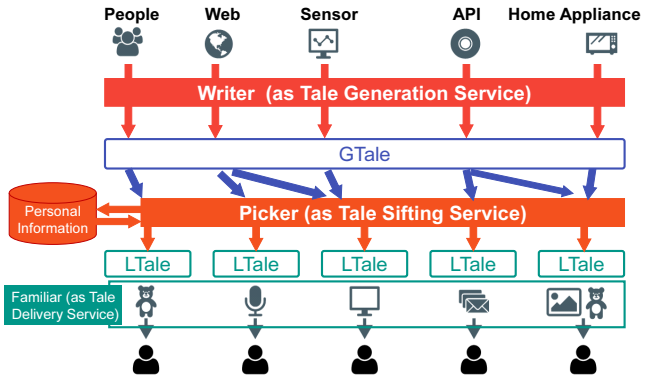


Figure 2. System Architecture of Tales of Familiar

filtering does not achieve sufficient resolution in tales selection. T3 deals with when the familiar should deliver the tales. We deploy an edge device with a proximity sensor, so that the familiar speaks only when the user gets close to it.

To evaluate the practical feasibility, we deploy the developed service in an actual household. Through the one-week experiment, the subject received Web news selected by the familiar. The empirical result shows that the subject felt it useful that the familiar pushed interesting news autonomously. We also evaluate how much our implementation was able to cover the technical issues.

## 2. Preliminaries

### 2.1. Framework of Tales of Familiar (ToF) [2]

In [2], we proposed the framework of ToF. A *tale* refers to a story (or a narrative) generated from data streams, which is an abstraction of the data for easy interpretation of the user. A *familiar* originally means a fairy that belongs exclusively to a human master. Thus, ToF is a service where an agent (as a familiar) always stays by a user and provides exclusively relevant information (as tales) for the user. The target users of ToF are basically *non-expert citizens* who do not make full use of various apps and services, proactively.

Figure 2 shows the proposed architecture of ToF. In the architecture, three services of *tale generation*, *tale sifting*, and *tale delivery* are vertically integrated using the service-oriented architecture (SOA) [3]. The top of the figure enumerates various data sources such as Web, sensors, and home appliances. Note that information manually produced by people is also a data source.

**Writer** refers to any module that implements the tale generation service. A writer generates tales from one or more data streams, and puts in a global repository of tales, called **GTale**, deployed in a cloud. Every tale in GTale is associated with a set of tags, which are used for the following tale sifting.

**Picker** is any module that implements the tale sifting service. From GTale, a picker selects relevant tales for a user, by matching the registered personal information with the tags attached to the tales. The selected tales are then stored in a local repository of tales, called **LTale**, dedicated for every user.

**Familiar** is an agent that implements the tale delivery service. A familiar speaks the tales in LTale to a user in familiar's spoken language, with an appropriate timing. We suppose that a familiar is implemented in a form of a robot, a virtual agent, or an IoT-embedded stuffed doll. According to the concept, every familiar belongs exclusively to a user, and knows (or learns) preferences of the user.

Although the proposed framework specified the concept and the architecture of ToF, it did not prescribe implementation details of each component. Thus, concrete implementations of Writer, Picker, and Familiar are left for individual application services with specific data streams of interests.

### 2.2. Related Work

The *digital content curation* [4] is an activity that delivers a set of Internet contents to readers of specific interests. The curation is basically performed by a human curator based on his/her aesthetics and values, but not by personal preference of individual users. This is different from the concept of ToF. The *publish/subscribe message exchange pattern* [5], which is now widely used in the IoT field [6], receives only data of interest from various kinds of data streams. However, it just delivers the data as it is. The interpretation of the data completely relies on users. Yasumoto et al. presented six challenges in utilizing IoT data streams [7]. ToF currently covers only part of them (C1: Creation, C3: Processing, C4: Presentation), which would be far beyond their ultimate goal. However, we believe that our contribution lies in implementing an actually-operating service with solid technologies. It would show the proof of concept of the pervasive flow of things to the society.

## 3. Personalized Web News Delivery Service

### 3.1. Motivation and Challenge

News articles on the Web can be considered as a typical data stream with the 3V property (volume, variety, and velocity). It is not easy for a general user to reach really interesting news on time, within dynamically added or disposed articles. Therefore, our goal is to implement a service that autonomously delivers personalized news articles to a user, based on the ToF framework.

The key components of the implementation are Writer, Picker and Familiar for the tale generation, sifting, and delivery services, respectively. In the context of the Web news delivery, we address the following three technical challenges within the three services.

**T1 (Duplication of Tales)** Due to the nature of news, similar news articles are published from different publishers. Therefore, without careful consideration, a writer would generate almost the same tales in GTale. Once matched to a preference to a user, the same tales are delivered many

times, which is very tedious for the user. We address this issue in the tale generation service.

**T2 (Value Estimation of Tales)** In order to wisely choose relevant news articles for a user, it is essential to consider how to estimate a *value* of every article for the user. The value should be characterized by user's interests registered in the personal information. However, the naive topic-based filtering fails to distinguish interests of a user from the ones of others. We address this issue in the tale sifting service.

**T3 (Delivery Timing of Tales)** When a tale arrives at LTale, every familiar should consider when to deliver the tale to its user. When the familiar speaks, if the user is absent, then the information cannot reach the user. This is completely meaningless. We address this issue in the tale delivery service.

### 3.2. Implementing Writer in Tale Generation

Today, many information providers are publishing news articles on the Web. These articles are generally summarized in RSS (Really Simple Syndication) feeds [8], which represent the summary of news articles in a structured format. Each provider refreshes the feeds as the news is updated.

We implement a writer module that periodically crawls designated RSS feeds, and generates tales from the feeds. Each of generated tales will be delivered by a familiar in a spoken language. Therefore, we consider that each tale should not be too long, because reading a long sentence takes much time. Hence, for each item of a given RSS feed, the writer module extracts the *title* of the news article, and uses the title directly as a body of a tale. The title typically describes a short story (i.e., headline) of the news. So, it is not difficult to convert the headline into a speech of familiar. For instance, we can use the following script template:

```
Hey, ${nickname}! ${publisher} is saying
"${body}". How do you think?
```

where actual values are assigned to variables (prefixed by $) when the tale is finally delivered by the familiar.

To characterize the feature of a given tale, the writer module generates a set of *tags* for the tale. Specifically, the writer extracts a tag from *item's category*. The writer also conducts the morphological analysis [9] to the headline text, identifies *proper nouns* in the headline, and uses them as tags characterizing the tale. Finally, the writer inserts the tale and the associated tags in a database.

Figure 3 shows an ER diagram representing data schema of the news delivery service. In the figure, a box represents an entity, and a line represents a relation (+—∈ denotes a parent-child relation, while +—··· denotes a reference relation). The upper half of the figure shows entities GTale and Tag for the generated tales and the tags, respectively. As seen in the figure, each tale is defined as a publisher, a body, a source RSS, time of creation, time of expiry, and a news category. A tale has one or more tags. For instance, the first instance of the tale has three tags.

Next, in order to cope with **T1 (Duplication of Tales)** in Section 3.1, we exploit the *cosine similarity* [10] to

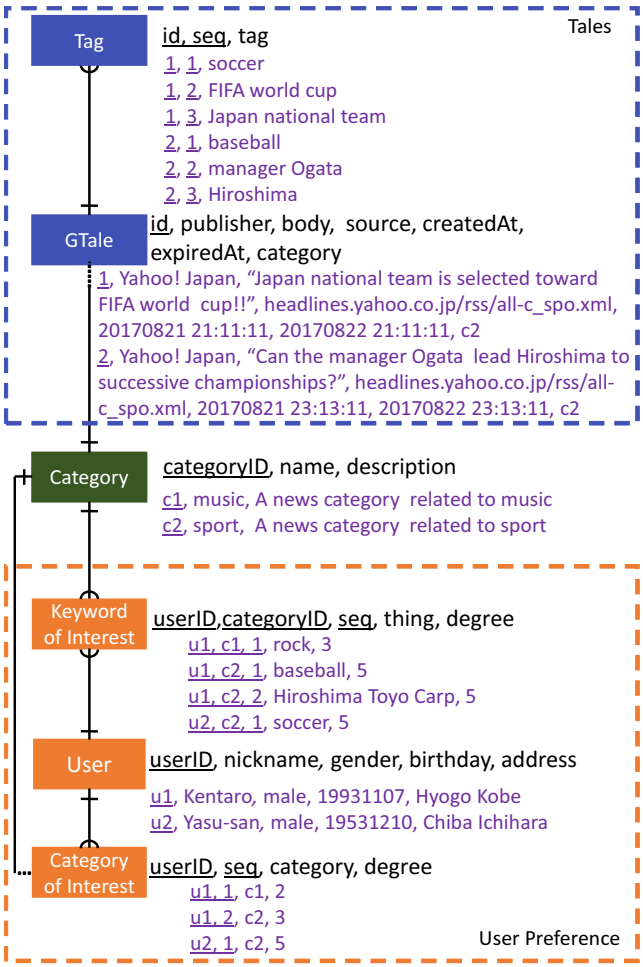


Figure 3. Data Schema for Web News Delivery Service

evaluate the similarity between any pair of generated tales. The method first encodes a given pair of texts by vectors $\vec{a}, \vec{b}$, and then evaluates the similarity by:

$$cos(\vec{a}, \vec{b}) = \frac{\vec{a} \cdot \vec{b}}{|\vec{a}||\vec{b}|} \quad (1)$$

When the angle of the two vectors is smaller (i.e., the cosine value is closer to 1.0), the method says two texts are more similar. There are many ways for the vector encoding.

In this paper, we encode every news headline by a binary (0-1) vector, where each element represents the appearance of a characteristic word. Specifically, for a given pair $t_a$ and $t_b$ of news headlines, we first apply the morphological analysis to extract nouns, verbs, and adjectives contained in the union of $t_a$ and $t_b$. The extracted words are then uniquely sorted into a sequence $(w_1, w_2, ..., w_n)$. Next, we check if each $w_i$ appears in $t_a$, and generate a vector $\vec{a} = [a_1, a_2, ..., a_n]$, where $a_i = 1$ only if $w_i$ appears in $t_a$. Similarly for $t_b$, we generate a vector $\vec{b} = [b_1, ..., b_n]$.

When the writer module inserts a new tale $t$ in the database, it calculates the cosine similarity between $t$ and

any tales generated over the last $h$ hours. If the similarity is more than a specified threshold $\tau$, the writer aborts to insert $t$ in the database. Thus, the issue of duplication of tales can be resolved. In the current system, we set the parameters $h = 24$ and $\tau = 0.7$.

### 3.3. Implementing Picker in Tale Sifting

Here we implement a picker module that sifts out relevant tales for every user from a lot of generated tales. Individual users have their own tastes and preferences. Therefore, different users have different *sense of values* for the same news article (generated as a tale). Let $V(u,t)$ denote a *value* that a user $u$ perceives for a tale $t$. If $V(u,t)$ is high (or low), $t$ should be chosen for $u$ (or discarded, respectively). Thus, the key issue to implement the picker is how to estimate $V(u,t)$, which is addressed as **T2 (Value Estimation of Tales)**.

Every tale $t$ in GTale has been already attributed by a set of tags. Hence, our basic idea is to estimate $V(u,t)$ by matching $u$'s *interests* with $t$'s tags. Since the ToF framework has a database to store personal information (see Figure 2), user's interests can be stored there. To implement the picker module, we consider the following three things.

**(A) Declaration of Interests**

The first thing is how a user $u$ declares his/her interests. A naive approach is to introduce the *topic-based filtering* of the publish/subscribe system [6]. A user subscribes to a set of topics of interest, and only the information relevant to the topics is delivered to the user. In the proposed service, a user could subscribe to interested news categories and keywords as topics. However, this approach does not work well. Since many articles in the same category are produced within a short period, there is no time for a familiar to read all the articles within the category. Also, characterizing user's preference by 0-1 values over the topics is too coarse to represent individual differences. Even if a user are interested in multiple topics, there must exist the *degree of interest* representing how much each topic is preferred by the user.

In this paper, we adopt an extended topic-based filtering method where every user specifies a set of news categories and keywords of interests, each of which has a 5-level weight representing the degree of interest. The bottom half of Figure 3 shows three entities managing the user preference. Each **User** has a set of news categories of interest (**Category of Interest**), each of which has a weight. For each of the category, a user has a set of keywords of interests (**Thing of Interest**), also weighted by the degree of interest.

**(B) Calculating Value of Tale**

Based on the user preference, we then present a method to calculate $V(u,t)$, the value of a given tale $t$ for a user $u$. For this, we compare $u$'s preference (w.r.t. category and keyword) and $t$'s attributes (w.r.t. category and tag), and sum up the degree of interests of matched elements. For convenience, we define the following notation:

- $u.cat$: a news category of interest of user $u$
- $u.key$: a keyword of interest of user $u$
- $deg_c(u.cat)$: the degree of interest of the category
- $deg_k(u.key)$: the degree of interest of the keyword
- $t.cat$: a news category of tale $t$
- $t.tag$: a tag attached to tale $t$

The calculation of $V(u,t)$ consists of two parts:

**($V_c$) Value by Category:** If a news category $t.cat$ of a tale $t$ is matched with a category of interest $u.cat_i$ of $u$, then $t$ is valuable for $u$. Hence, we define the value $V_c(u,t)$ by the category as follows:

$$V_c(u,t) = \begin{cases} deg_c(u.cat_i) & (\exists i; u.cat_i = t.cat) \\ 0 & (otherwise) \end{cases} \quad (2)$$

**($V_k$) Value by Keyword:** If a tale $t$ contains a keyword that $u$ is interested in, then $t$ is valuable for $u$. For this, we compare each keyword of interest $u.key_i$ of $u$ and each tag $t.tag_j$ of $t$. If they are *semantically equivalent* (denoted by $\approx$), we sum up $deg_k(u.key_i)$ as the value by the keyword. Even if they do not match, if they are *conceptually related* (denoted by $\sim$), we sum up a certain value $\alpha \times deg_k(u.key_i)$, where $\alpha$ is coefficient between 0 to 1 [1].

$$V_k(u,t) = \sum_i \sum_j f(u.key_i, t.tag_j) \quad (3)$$

where

$$f(u.key_i, t.tag_k) = \begin{cases} deg_k(u.key_i) & (u.key_i \approx t.tag_j) \\ \alpha \times deg_k(u.key_i) & (u.key_i \sim t.tag_j) \\ 0 & (otherwise) \end{cases}$$

Finally, we define $V(u,t)$ by a product of $V_c$ and $V_k$:

$$V(u,t) = (1 + V_c(u,t)) \times (1 + V_k(u,t)) \quad (4)$$

Currently, we set $\alpha = 0.8$ to make the distance between equivalent ($\approx$) and related ($\sim$) shorter than the distance between related and independent.

For example, in Figure 3, let us calculate the values of tales #1 and #2 for Kentaro (userID:u1). Then, $V(u1,1) = (1 + V_c(u1,1)) \times (1 + V_k(u1,1)) = (1+3) \times (1+0) = 4$, and $V(u1,2) = (1 + V_c(u1,2)) \times (1 + V_k(u1,2)) = (1 + 3) \times (1 + (5 + 0.8 \times 5)) = 40$.

**(C) Sifting Tales by Value**

Once $V(u,t)$ is calculated, a picker has to determine if $t$ should be chosen for $u$. An easiest way is to introduce a threshold function that chooses $t$ when $V(u,t) \geq \mu$. However, the threshold function eliminates any chance that a user meets news articles with low values. To support the *serendipity*, we introduce the *logistic distribution* $P(v)$ to define the probability of choosing a tale of value $v$:

$$P(v) = \frac{1}{1 + e^{-(v-\mu)/s}} \quad (5)$$

1. Strictly speaking, relations $\approx$ and $\sim$ should be evaluated with a certain ontology framework. However, our current system adopts simple word matching with a thesaurus, just for convenience.

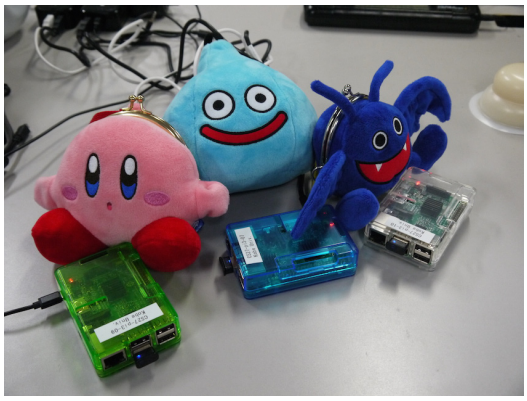

Figure 4. Stuffed Dolls as Familiars (©Nintendo/HAL, ©SQUARE ENIX)

where $v$ $(= V(u,t))$ is the value of $t$ for $u$, and $\mu$ and $s$ are the location and scale parameters. The parameter $\mu$ is dynamically adjusted depending on how many tales the user wants to receieve within a period.

The picker module periodically scans updated tales in GTale, and calculates the value of each of the tales for every user. Then, the picker determines the choice based on $P(w)$, and finally puts the chosen tales in LTale.

### 3.4. Implementing Familiar in Tale Delivery

Finally, we develop a familiar that delivers the sifted tales in LTale for a user. Considering the portability and acceptability for non-expert users, we use a *stuffed doll* for the body of the familiar. Figure 4 shows the developed prototypes. We embed a small wireless speaker in the stuffed doll, which is paired with an external *edge server*. To manage **T3 (Delivery Timing of Tales)**, we also attach an *infrared sensor* to the doll, detecting proximity of a user.

Figure 5 shows a sequence diagram representing how a tale is delivered to a user. First, the edge server periodically synchronizes tales stored in LTale to a local storage. When the user gets close to the familiar, the sensor detects the proximity and triggers the tale delivery. The edge server pops up a tale with the highest value, and sends the text to Emotion Analysis API to obtain an emotion guessed from the text. Then, the edge server passes the text with the emotion parameter to Text to Speech API, to synthesize voice data with the emotion. The edge server attaches the prefix data to the voice data, and finally plays back the voice data on the wireless speaker in the familiar.

Thus, the user can see as if the familiar is kindly telling an exclusively interesting news for the user.

### 3.5. Technologies Used

Technologies used in the implementation of the Web news delivery service are summarized as follows:
**(A) ToF Server Side (Cloud):** Java1.8, MySQL, MyBatis 3.4.0 (for OR mapper), Kuromoji 0.7.7 (for the morphological analysis), jsoup 1.7.3 (XML Parser), Apache Axis2 1.6 (for Web service middle-ware).

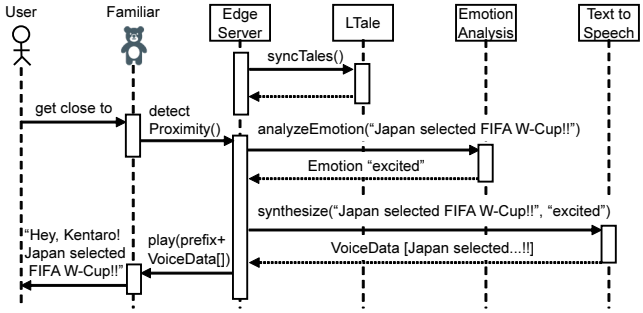


Figure 5. Sequence Diagram of Tale Delivery

TABLE 1. USER PREFERENCE OF SUBJECT

| | |
|---|---|
| Categories of Interests | Sports (3), Entertainment (4), IT (5), Local (2) |
| Keywords of Interests | soccer (3), basketball (5), World Cup (4), Neymar (5), Kei Nishikori (5), Japanese idol (3), Ryosuke Yamada (5), Johnny & Associates (5), Kamen Rider (5), Fujitsu(4), HTC(5), Sony (3), Hyogo Prefecture (2), Kobe City (4) |

**(B) ToF Client Side (Edge):** Ruby, HTML5, CSS3, JavaScript, Raspberry Pi3 (for the edge server), XAM17 X-mini WE (for the wireless speaker), Phidgets Motion Sensor 1111_0 (for the proximity sensor), Text to Speech API by Docomo (for the speech to text), Sentiment Analysis API by Metadata Incorporated (for the emotion analysis).

## 4. Experimental Evaluation

To see the practical feasibility, we have conducted an experimental evaluation, by deploying the implemented familiar in an actual household of an author. The period of the experiment was one week from September 1st to 8th, 2017. For the experiment, the subject registered the user preference to the service, which is shown in Table 1. The number attached to each item represents the degree of interest. For the tale generation, we configured the writer module to sample "Yahoo! News RSS" (https://headlines.yahoo.co.jp/rss/list) every ten minutes.

Web news delivery worked fine without any technical problem. During the one week, 17,898 tales were generated in GTale, and 807 were chosen and delivered. The subject felt it useful to hear the latest news of interest, automatically. Table 2 shows a part of the tales that left impression on the subject. Each row represents a tale, associated with the calculated value and the comment given by the subject.

In Table 2, we can see that news articles with the categories of interests are chosen and delivered. Interestingly, as for tale #430500 with the highest value, it seems that the subject had already known the information before the tale was delivered. This may yield a hypothesis that even a non-expert user proactively searches information if it is highly relevant. For tale #433004 with the lowest value, it seems to give a sort of serendipity to the subject. This is the great effect of introducing the logistic function instead of the threshold function, as devised in Section 3.3-(C).

TABLE 2. A PART OF DELIVERED TALES, AND COMMENT BY THE SUBJECT

| id | createdAt | body | category | value | comment by subject |
|---|---|---|---|---|---|
| 414166 | 2017/09/01 04:10:01 | GEKISAKA News says: Paris SG got Mbappe in 23 billion yen. Towards gorgeous co-starring with Neymar. | sports | 36 | How fabulously rich Paris SG is! |
| 416938 | 2017/09/02 06:20:15 | Sports Houchi says: Honda appreciates pressure from younger players: "I will snatch positions of Asano or Ideguchi". | sports | 16 | That's good for his rejuvenation. |
| 427863 | 2017/09/06 11:20:20 | Kobe Keizai Shimbun says: Kobe Kitano Hotel starts "Night Dessert Buffet" with traditional French sweets and Autumn tastes. | local | 21 | Wish to stay Kitano Hotel just once... |
| 430500 | 2017/09/07 08:20:10 | Nikkan Sports says: The musical of Johnny's Junior: Dream of Mr. Johnny. | entertainment | 50 | Yes, I am looking forward to seeing. |
| 433004 | 2017/09/07 22:20:12 | Okinawa Times says: World-longest chain of Origami Cranes awarded Guinness record. Meaning of 9.7km for Okinawans. | local | 3 | Wow! That's a great Guinness record. |

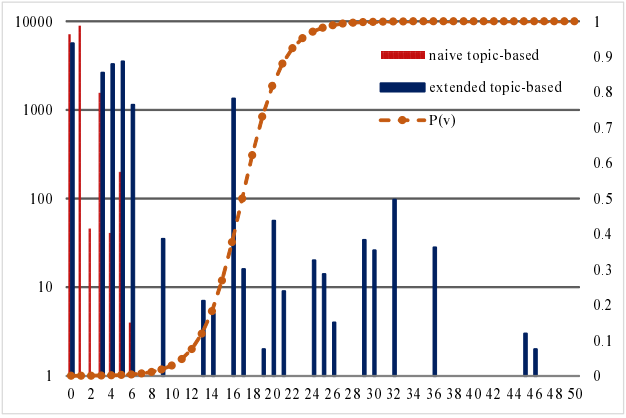


Figure 6. Distribution of Tales over $V(u,t)$ and $P(v)$

Figure 6 shows the frequency distribution of the generated tales over the value for the subject. The horizontal axis represents the value $V(u,t)$, while the vertical axis plots the number of tales of the value. For comparison, we plot two distributions with the naive topic-based and the extended topic-based (proposed) method. The dotted curve shows the logistic distribution $P(w)$ $[\mu = 17, s = 2]$ used for the tale sifting. We can see that the proposed approach (devised in Section 3.3-(A)(B)) can prioritize a lot of news articles with the sufficient range of values, calculated from the degree of interest of the subject. On the other hand, the naive approach results in coarse classification with a narrow range of values, which cannot reflect the preference well.

In the system log, we found that 554 tales out of the total 17,898 were discarded as duplicated tales. Thus, the de-duplication method by the cosine similarity (devised in Section 3.2) eliminated about 3.1% of news articles as redundant. We consider it reasonable as for the simple implementation. Introducing more sophisticated ontology may improve the quality, which is left for future work.

## 5. Conclusion

In this paper, we have developed a personalized Web news delivery service using the Tales of Familiar (ToF) framework. The service autonomously retrieves interesting news articles from the Web, and an agent called familiar speaks the articles as tales for the user. We conducted an experiment by deploying a stuffed doll (as a familiar) in an actual household. It was shown that the proposed service worked well, with achieving the technical challenges of duplication, value estimation, and delivery timing of tales.

Our future work is to evaluate the service with more subjects, including elderly people. Based on the feedback, we improve the quality of tales by introducing more sophisticated techniques in aggregating and comparing generated tales. Developing ToF services for sensors and IoT is also a challenging issue, which is currently under development.

**Acknowledgments**

This work was supported by JSPS Grants-in-Aid for Scientific Research, Grant Numbers JP16H02908, JP15H02701, JP17H00731, JP15K12020.

## References

[1] Japan Ministry of Education, Culture, Sports, Science and Technology, "Challenges in realizing a super smart society supported by the IoT, big data, and artificial intelligence – japan as a global frontrunner –," *White Paper on Science and Technology*, 2016.

[2] K. Noda, Y. Wada, S. Saiki, M. Nakamura, and K. Yasuda, "Delivering personalized information to individuals in super smart society," in *Digital Human Modeling 2017 (DHM 2017)*, no. LNCS 10286. Springer International Publishing AG 2017, July 2017, pp. 336–347.

[3] T. Erl, *Service-Oriented Architecture: Concepts, Technology, and Design*. Prentice Hall.

[4] C. Padoa, D. Schneider, J. M. de Souza, and S. P. J. Medeiros, "Investigating social curation websites: A crowd computing perspective," in *IEEE 19th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, 2015, pp. 253–258.

[5] P. T. Eugster, P. A. Felber, R. Guerraoui, and A.-M. Kermarrec, "The many faces of publish/subscribe," *ACM Comput. Surv.*, vol. 35, no. 2, 2003.

[6] OASIS, "MQTT Version 3.1.1," http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html, Oct. 2014.

[7] K. Yasumoto, H. Yamaguchi, and H. Shigeno, "Survey of real-time processing technologies of iot data streams," *Journal of Information Processing*, vol. 24, no. 2, pp. 195–202, 2016.

[8] RSS Advisory Board, "RSS 2.0 specification," http://www.rssboard.org/rss-specification, Aug. 2002.

[9] Atilika - Applied Search Innovation, "Kuromoji – an open source japanese morphological analyzer," https://www.atilika.com/en/products/kuromoji.html.

[10] M. Steinbach, G. Karypis, and V. K. Kumar, "A comparison of document clustering techniques," in *KDD workshop on Text Mining*, Jun. 2000.