A Supervised Approach for Reconstructing Thread Structure in Comments on Blogs and Online News Agencies

Ali Balali, Hesham Faili, Masoud Asadpour, and Mostafa Dehghani

School of ECE, College of Engineering, University of Tehran, Tehran, Iran

{balali.a67, hfaili, asadpour, mo.dehghani}@ut.ac.ir

Abstract. There is a great deal of knowledge in online environments such as forums, chats and blogs. A large volume of comments with different subjects on a page has created a lot of complexity in following the actual conversation streams, since the reply structures of comments are generally not publicly accessible in online environments. It is beneficial to automatically reconstruct thread structure of comments to deal with such a problem. This work focuses on reconstructing thread structures on blogs and online news agencies' comment space. First, we define a set of textual and non-textual features. Then we use a learning algorithm to combine extracted features. The proposed method has been evaluated on three different datasets, which include two datasets in Persian and one in English. The accuracy ratio of the proposed model is compared with three baseline algorithms. The results reveal higher accuracy ratio for the proposed method in comparison with the baseline methods for all datasets.

Keywords. Reconstructing thread structure, reply structure, information extraction, blogs and online news agencies, machine learning, information management.

El enfoque supervisado para reconstrucción de la estructura de hilos en comentarios en blogs y agencias de noticias en línea

Resumen. Una cantidad grande de conocimiento está hoy en línea en varias formas como foros, chats y blogs. El gran volumen de comentarios acerca de diversos temas en una página ha creado gran complejidad para realizar el seguimiento de los flujos reales de conversación, ya que las estructuras de respuesta a comentarios por lo general no son de acceso público en las páginas web. Sería beneficioso reconstruir automáticamente la estructura de hilos de comentarios para resolver este problema. El presente trabajo se centra en la reconstrucción de la estructura de hilos en el espacio de comentarios en blogs y agencias de noticias en línea. En primer lugar, se define el conjunto de características textuales y no textuales. Luego se utiliza un algoritmo de aprendizaje para combinar las características extraídas. El método propuesto ha sido evaluado sobre tres distintos conjuntos de datos, que incluye dos conjuntos de datos en idioma persa y un conjunto en inglés. La precisión del modelo propuesto se compara con tres algoritmos de referencia. Los resultados muestran mayor precisión del método propuesto en comparación con los métodos de referencia para todos los conjuntos de datos.

Palabras clave. Reconstrucción de la estructura de hilos, estructura de respuestas, extracción de información, blogs y agencias de noticias en línea, aprendizaje de máquina, administración de información.

1 Introduction

One of the problems of information management in online interactive environments is the limitation of reply capability to a relevant comment. Most online community systems provide a view of posts in chronological order and users can post a comment at the end of all comments regardless of the comment to which they want to reply. Due to the lack of this capability, several irrelevant comments may appear between comments and users have to read all the comments to reach the relevant comment. A solution to this problem is to automatically reconstruct thread structure of the comments.

This work focuses on blogs and online news agencies. Regarding various issues discussed in these environments, a wide range of people with different ideas participates in the discussions. So they are rich sources of information.

First of all, the terms comment, commenter, candidate set, thread, thread detection and reconstructing thread structure are defined as follows. A **comment** is an utterance written by a user, comprising one or several sentences. A **commenter** is a person who comments on a news item and may comment on a news item more than once [1]. In this paper**, start post** or **root** is defined as the discussion starter, which is the main content or news in blogs and news service environments written by an author. This type of content is shown in Figure 1 as a node with label R. **Reply comments** are responses to previous comments or news items, which are represented by numbers in Figure 1. For example, the nodes with labels 1, 4 and 5 are reply comments to the root comment, which is considered as their parent. The sequences of labels in Figure 1 are based on the times at which comments were created. A **thread** is a sequence of comments which starts in response to some main content and contains a series of reply comments which all are related to the same topic [2]. **The candidate set of the i^{th} comment** is a set of comments which could be considered as the parent of the i^{th} comment and includes comments which appear before the i^{th} comment in chronological order.

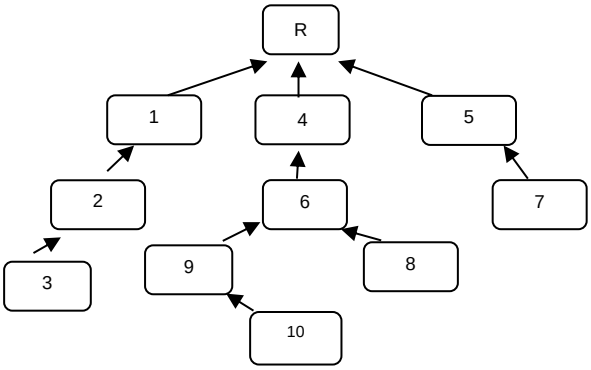

Fig. 1. An illustration of the goal of RTS task

The thread detection task means finding the cluster of comments that belong to the same discussion in a given text stream without any previous knowledge about the number of threads [2]. **The reconstructing thread structure (RTS) task** means reconstructing the reply structure on comments on the threads. This leads to construction of a tree structure [2-4] or a directed acyclic graph (DAG) [1, 5, 6], which represents the reacts-on relation between the root and a set of comments. Since threads with tree structure in the datasets are used in this paper, we assume the RTS task to be the recovery of the tree structure of comments in each thread.

Thread detection task, which sometimes is called topic detection in the literature, should be accomplished as preprocessing for the RTS task. In other words, firstly, all comments should be split into a number of threads and then the RTS could be exploited to recover tree structure in each thread. Thread detection task is not essential in online news agencies and blogs since all comments are related to the main post. Based on this reality, all comments can be considered only in one thread. Consequently, we focus only on the RTS task in this study.

There are some advantages of the RTS task in online environments including but not limited to: facilitating search and finding the user's favorite content [7], identifying users who have the ability to answer the questions [8], isolating discussions related to specific subtopics [6], understanding the online user's behavior [9] and facilitating following of the actual conversation stream in threads [3].

Valuable studies have been done in forum, emails and chat environments, but regarding differences of these environments with blogs and online news agencies, the methods developed for forums, chats, and emails are not suitable here. Some of the differences between these environments can be explained as follows:

- The main content or root is the parent of many comments. In our datasets nearly 30 percent of the comments are children of the root. This is different from forums, chats, and emails where each comment links most likely to its first previous comment (1-Distance) [3].
- It is more probable to have a reply to the root in blogs and online news agencies even for the last comments. So lots of features such as time distance [6], are not suitable here.
- In blogs and online news agencies, commenters express mostly their opinions or

sentiments, which are utterances in informal dialogues, while in forums they post mostly their questions and answers, and this is more formal than blogs and online news agencies.
– Presence of quoted phrases helps a lot in RTS. This feature is more usual in forums and emails.

In this work, in order to accomplish the RTS task, some relevant textual and non-textual features are defined. Then, a learning algorithm is used to construct a proper model, which is exploited to identify the reacts-on relation between a root and a set of comments. In other words, two comments are fed into the trained model to determine if there is any relationship between them or not. The proposed RTS method is called SORTS (a Supervised apprOach for Reconstructing Thread Structure).

In summary, the contributions of this work are:

– We propose a supervised approach to reconstruct thread structure using textual and non-textual features on blogs and online news agencies.

– The SORTS method is tested on three different datasets: two datasets are in Persian and one is in the English language. Each dataset comprises lots of reply structures of comments. The reply structures of comments come mostly from real structures created by users. They are addressed by reply tags in our datasets.

– The focus of this paper is reconstructing thread structure on blogs and online news agencies, since only a few studies have been carried out on these environments.

– The results reveal higher accuracy ratio for the SORTS method in comparison with the three baseline methods for all of the datasets.

The rest of this paper is organized as follows: Section 2 describes related work, in Section 3 we propose our methodology, explain the SORTS method for the RTS task and present the datasets. Finally, experimental results are shown and discussed.

2 Related Works

There are a few studies in the literature that directly address the problem of the RTS task for comments on blogs and online news agencies. In general, the RTS task can be done based on two approaches: unsupervised and supervised techniques. In unsupervised methods, the relation between two comments is quantified using a text similarity measure considered as a relationship weight. Then, the weights are adjusted using other metrics such as time distance. Finally, the relations whose weights are higher than a predefined threshold are selected as the parent-child relations [5, 6].

In supervised methods, the existence of a relationship between two messages is determined using a learning algorithm [1, 3, 4]. In these methods, a set of features is defined and weighted using a training set. Then, the trained model is used to recover comment relations in the test data with the help of the extracted features. The SORTS method is a supervised approach.

The methods proposed in [1, 3] are supervised; they use a set of simple features and a classifier. The features are divided into two groups. The first group consists of structural or non-textual features such as message placement, time information, reply distance and the author's name. The next group of features includes semantic or textual features related to linguistic information such as sentences type, similarity among messages.

Seo *et al.* [4] investigated a learning technique that exploits the hierarchical thread structures in forum and email environment. They introduced structure discovery techniques that use a variety of features to model relationship among posts belonging to the same thread. In fact, their method is very similar to ours; however we focus on blogs and online news agencies while they have focused on forum and email environments.

Some previous works on RTS, applied to online news agencies, have focused on the author's name [1]. If a commenter's name appears in a comment, the comments which are posted by that commenter are selected as the parent of that comment. Their work is one of the baselines in this paper.

There are also some related works on other types of data such as email data [10, 11]. However, some specific features exist in those environments that are not applicable here e.g. “To/CC” tag in email data.

3 Methodology

In this section we describe our supervised approach to the RTS task. Firstly, we extract some textual and non-textual features from the training set and learn a proper model to combine the extracted features using a ranking SVM (Support Vector Machines). Then the model is employed in reconstruction of the reply structure for the threads in the test data. The proposed RTS method is called SORTS.

3.1 Ranking SVM

SVM is a supervised learning method that is used to analyze the data and recognize patterns. It is used for classification and regression analysis and also for the ranking problem. The implementation we use is the ranking SVM classifier[1] [4, 12]. After training, the ranking SVM classifier is able to assign a weight to the pairs of comments. The whole procedure for choosing the parent of the i^{th} comment in a thread is described in Figure 2 [4].

```
Procedure RTS(i,C) // C is the whole set of comments
// For each candidate parent for the i^th comment
for k ← 0 to i-1 do
A[k] ← ranking SVM(C[i], C[k])
// Return parent's label of the i^th comment
return argmax_k A[k]
```

Fig. 2. The RTS algorithm

3.2 Features

To use the ranking SVM classifier, we need to define several features. In this section, we introduce nine textual and non-textual features. Their definition is one the main contributions of our work. The usefulness of each feature is reported at the end of its description. We use the backward feature selection method to study the impact of the presence of each feature in the RTS task. In other words, to calculate the usefulness of each feature, the result is recalculated without taking this feature into account and the percentage of difference between the achieved accuracy and full-feature accuracy is considered as the usefulness of that feature.

3.2.1 Similarity Feature

In order to measure the similarity of sentences, we utilize a vector space model to represent a comments’ body text. A comment’s text can be considered as a vector of terms, weighed by TF-IDF. TF (Term Frequency) is the number of word appearances in the comments’ text and IDF (Inverse Document Frequency) is computed according to the following formula:

$$IDF(w)=Log(N/n_w) \quad (1)$$

where N is the number of all comments in a news item and n_w is the number of comments which contain the word w.

In order to measure similarity between two comments, stop-words are deleted first, and then words of the comments are stemmed by the Porter algorithm (for English datasets). Then, weight of the words are calculated based on TF-IDF. Then, the final score is obtained by aggregating the weights of all common words.

We used the Okapi BM25 algorithm [13] for computing the similarity score based on TF-IDF of a word:

$$TF-IDF(w,c) = \frac{tf(w)}{tf(w)+0.5+1.5\,.\,\frac{|c|}{<|c|>}}\,.\,log\left(\frac{N}{df(w)}\right) \quad (2)$$

where N is the number of comments in the news, c is the comment, $|c|$ is the length of the comment, $<|c|>$ is the average length of comments, and $df(w)$ is the number of the news’ comments which include the word w [6].

[1] http://www.cs.cornell.edu/people/tj/svm_light/svm_rank.html

Comments might have typo errors. Some words in two comments might be the same but due to spelling errors, it is difficult to find this out.

In order to solve this issue, the minimum edit distance (MED) algorithm is used. The minimum edit distance between two words is the minimum number of edit operations (insertion, deletion, substitution and transposition) needed to transform one word into another [14]. The costs of insertion, deletion, substitution, and transposition are 1,1,2,1, respectively.

Two words in different comments are considered as common words if either they are exactly the same or they seem to be the same but they contain some typo errors.

In the latter case, if the length of the words are bigger than five, and their first two letters are the same, and their edit distance is lower than 4, the two words are considered as common word. For example, two words "Beautiful" and "Beuatiful" are considered as common words.

This feature can improve results by 7.66 %.

3.2.2 Discussion Feature

The purpose of this feature is to detect a discussion among several commenters. To illustrate this feature, let us see an example where seven commenters have written eighteen comments. Let us say {D, F, D, F, A, B, C, A, B, A, B, A, T, M, D, C, D, C} is the sequence of the comments by commenters, where latters show the commenters' ID. For example, commenter A has written four comments and commenter B has written three comments and the position of their comments are quite close to each other. It is very likely that there are discussions between two commenters A and B. So, where a comment belongs to B, its parent is likely a comment written by A.

Since a commenter might have discussion with different people in different positions, comments of each commenter are partitioned based on their positions, then the discussion feature is calculated based on the partition. The comments that have been written by the same commenter and have a close distance are placed in the same part. For example, comments of commenter "D" are partitioned into two parts {1,3}{15,17}.

The threshold for close distance in our paper was empirically set to 4. It means the distance between the positions of comments is less than 4 with at least one of the other comments in the same part. Large partitions probably contain more discussions.

The discussion feature is calculated according to the following formula:

$$\text{Score}(P_{iJ},P_{kZ})=\frac{min(n,m)\,log(min(n,m))}{\sum_{u=1}^{min(n,m)} min(distance|P_{iJ_u},P_{kZ}|)} \quad (3)$$

where P_{iJ} is the i^{th} partition of commenter J who has totally n comments. P_{kZ} is k^{th} partition of commenter Z who has totally m comment. P_{iJ_u} is the u^{th} comment of the i^{th} partition of commenter J.

To use this feature, when we are looking for the parent of the i^{th} comment among the candidates, we should first find the partition of the i^{th} comment and then find all its candidate partitions and finally we calculate the score among partitions of the i^{th} comment and all its candidate partitions. This feature can improve results by 0.6%.

3.2.3 Author's Language Model

This feature is indirectly related to the two previous features. The idea is that the commenters who discuss together are more likely to use similar words in their comments.

In order to take advantage of this feature, all comments of the commenters are appended to each other. Then, similarity is calculated between the comments of two commenters. Very similar collections of words are more likely to have a relation. This feature can improve result by 0.5 %.

3.2.4 References to the Author's Name

According to the author's name feature, if a commenter's name appears in a comment, the all comment which are posted by that commenter are selected as the candidates of that comment [1]. Sometimes the author's name is made up of two parts, and either of these parts could be used by other authors for reference. We also consider these types of references. We hold each part of the author's name and then parts which are stop-

words are removed. This feature can improve the result by 1.69 %.

3.2.5 Prior Location

Each comment based on its position could be considered as parent. For example, initial comments are more likely to have more children than the others, or comments which are located just before the i^{th} comment are more likely to be its parent. Formally, we want to estimate $P(i|j)$, that is, the likelihood that the comment in position *i* is the parent of the comment in position *j* [4]. So we calculate prior probabilities for being the parent of different positions.

To calculate prior probability for the j^{th} position, we count the number of times that each candidate comment is the parent of the j^{th} comment based on the training set. Suppose prior probability between the i^{th} comment and the j^{th} comment is 0.3; when we are looking for the parent of the j^{th} comment, this feature assigns a probability of 0.3 to the i^{th} comment. This feature can improve the results by 7.51 %.

3.2.6 Author's Activity

The commenters who write more are likely to initiate more discussions. We can assign higher probabilities to the comments by active authors. The probability of an active author is calculated according to the following formula:

$$\text{Score}(C_i, C_j) = \frac{\sum_{k=1}^{i-1} U(j,k)}{Z} \quad (4)$$

$$\text{U(j,k)} = \begin{cases} log_2^{(k)} & if(A_j = A_k) \\ 0 & otherwise \end{cases} \quad (5)$$

where A_j is the commenter's name of the j^{th} comment and $Z=\sum_{k=1}^{i-1} Score(C_i, C_k)$ is a normalization factor. We are looking for the parent of C_i , and C_j is one of its candidate comments. Function U determines the parent of the i^{th} comment regarding the commenters' activity near the i^{th} position. The idea is that the comments of commenters who are more active near the i^{th} position are more probable to be the parent of the i^{th} comment. This feature can improve the results by 0.5%.

3.2.7 Candidate Filtering

We present some features that are used to heuristically filter the candidates. These features are as follows:

1. Usually a commenter does not reply to the root post in his/her second comment. So if a commenter has written more than one comments, the main text (root) is removed from its parent candidates. This heuristic is 90.75% true in our datasets. This feature improves the result by 3.69%. It is a powerful feature because the root is an important candidate, so if we could remove it correctly, the results are improved significantly.
2. A commenter does not reply to him/herself. So we simply remove all comments of a commenter from his/her comment's candidates[4]. This feature improves the result by 0.1%.
3. Commenters who post only one comment on a thread are more likely to reply to the root post. So other candidates can be removed except the root. This feature improves the result by 0.54%.

3.3 Baseline Approaches

We use three different baselines. The first baseline is to link each comment to its first previous comment (1-Distance) [3]. This method leads to good results for the RTS task in chat and forum environments. Further, we implement two method from the previous works i.e. Schuth *et al*.'s [1] and Seo *et al.*'s [4] methods.

Schuth *et al*.'s method used several features to find the commenter's name in comments. Schuth *et al*.'s method achieves a rather good precision (near 0.7) but it has a low recall (near 0.08) because in our datasets there are many comments that do not refer to any author's name. In order to define a more powerful baseline, we enhance Schuth *et al*.'s method by adding a feature to it. The feature connects a comment to the root post if there is no reference to the commenter's name in that comment. This is a strong baseline in blogs and online news agencies because the root is considered as the parent of many comments. In our dataset nearly 30 percent of the comments reply to the root.

Seo *et al.*'s method is very similar to ours but they focused on forums and emails and used the quoted text as one of their important features, which does not appear in our datasets. We use all features of Seo *et al.*'s method except the feature that uses the quoted text.

4 Experiments

In the following we provide details about the datasets used for evaluation, describe the performance measures and then present the results of the SORTS method and compare them with the results of the baseline approaches.

4.1 Datasets

In order to provide the datasets that is used for evaluation, three websites are crawled: Alef (alef.ir), Thestandard (thestandard.org.nz) and Narenji (narenji.ir). The first two websites are online news agencies and the last one is a blog. Narenji and Alef are in Persian and Thestandard is in English. Narenji is a famous Persian blog about technology and gadgets, which is daily updated. Alef is an online news website. In Alef, news articles are published in various categories i.e. politics, culture, economy, sports. Most of the news is published in the politics category.

Thestandard is an online news agency like Alef, who publishes news in different categories such as economy, environment, international news, media, politics, and social issues.

In all of these sites, people can write their comments related to an article or reply to the other comments. There is no limit on the number of comments that can be posted for articles. These websites support a multilevel reply structure.

The number of threads and their average size per dataset are shown in Table 1. Thestandard is the biggest dataset in terms of the number of articles and comments. Alef has longer threads in average, but Thestandard's comments have more words in average. Narenji has both smaller threads and smaller comments than the other two datasets.

Figure 3 shows the distribution of the number of comments in articles of the three datasets. Thestandard articles have usually around 10-15 comments. Alef articles have usually around 20-25 comments. Narenji comments are somehow different. Alef has a better position in ranking rather than Thestandard and Narenji based on Alexa[2] ranking, so, it has more visitors and comments. Also, the number of comments on the news depends on many factors such as content of the news articles[15] and publication time.

In these datasets, each news article has a unique ID. The test set for experiments includes the news articles whose ID ends either with zero or one and the training set includes the rest.

4.2 Evaluation Metrics

To evaluate the results of the experiments, we use standard precision, recall, and F-score measurement [3, 5] but as mentioned previously, the output of the SORTS method is a tree and precision, recall, and F-score values are equal [4], since FP(false positive) and FN(false negative) in precision and recall are always equal.

Table 1. Some statistics on the datasets

Site	Number of news articles	Number of comments	Thread size			Average length of comments (in words)
			min	max	avg	
Thestandard	3197	150603	1	372	47.1	67.64
Alef	490	37990	3	934	77.53	63.4
Narenji	673	27734	3	287	41.20	29.00

[2] Alexa – The web information company, www.alexa.com

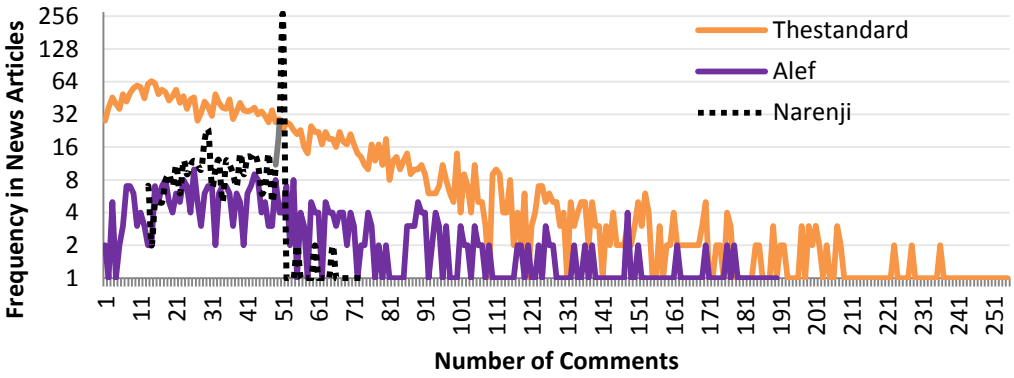

Fig.3. Distribution of the number of comments in news article

We use the following accuracy measurement:

$$\text{Accuracy} = \frac{|\{reply\ relations\} \cap \{detected\ relations\}|}{|\{reply\ relations\}|} \quad (6)$$

where the tree is comprised of *N* comments and one root, reply relations are equal to *N*.

4.3 Evaluation of Results

In this section, the results of the SORTS method are presented and its accuracy is compared with three baseline algorithms.

The results on each dataset are shown in Table 2. Each value in Table 2 is obtained based on the average accuracy in all threads of the dataset.

The results reveal higher accuracy for the SORTS method in comparison with the three baseline algorithms in all three datasets. It is from 2 to 6 percent better than the best of the other three methods. It gains the best performance on the Alef dataset (around 53 %).

Since the length of sentences in Narenji dataset is smaller than the other datasets, and blogs are more informal than online news agencies, the accuracy in Narenji dataset is less than that of the other two datasets.

1-distance does not have a good result on Alef and Narenji datasets because comments in these sites wait for moderation before publish and it usually takes some time. Moderators are not always online; they log in a few times per day, accept the sent comments and log out. Therefore, multiple comments appear nearly at the same time.

Table 2 Accuracy of SORTS compared to three baselines

Dataset	SORTS	Schuth 2007	1-distance	Seo 2011
Thestandard	**0.4751**	0.4055	0.1651	0.4195
Alef	**0.5264**	0.4878	0.07	0.5047
Narenji	**0.4418**	0.4012	0.068	0.4137

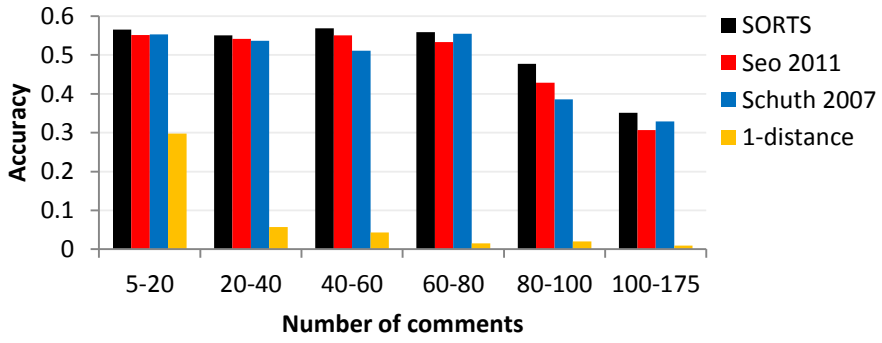

Fig. 4. Results on Alef dataset based on the number of comments per thread

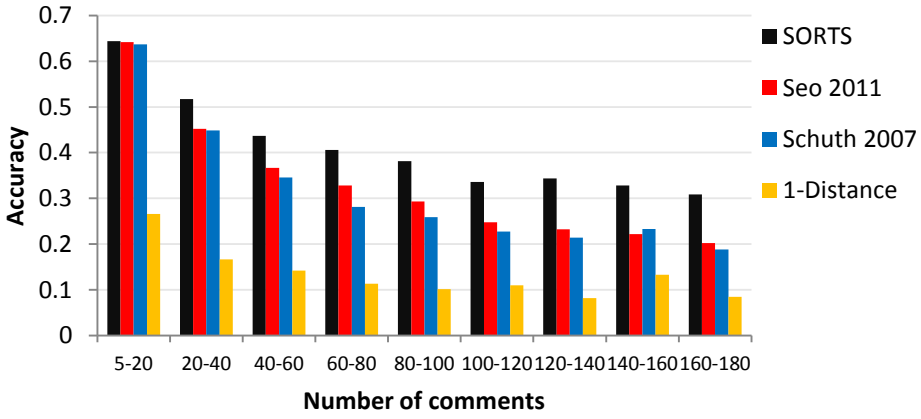

Fig. 5. Results on Thestandard dataset based on the number of comments per thread

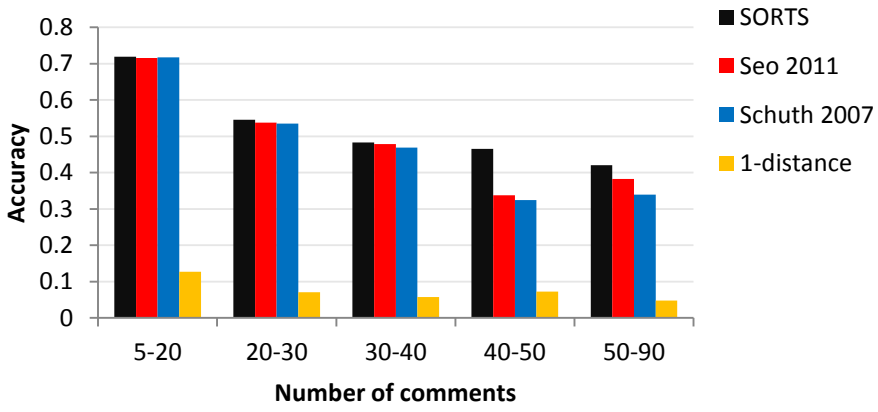

Fig. 6 Results on Narenji dataset based on the number of comments per thread

The accuracy of the methods categorized based on the number of comments per thread are shown in Figures 4, 5 and 6.

It is seen that the Schuth *et al*.'s method has a good result on news that have less than 20 comments, because in these news articles most comments reply to the root and usually there are no discussions among commenters.

Since we use some new features like the discussion feature, the author's activity and candidate filtering features, the SORTS method has a better accuracy than Seo *et al*.'s method in news items which have more than 20 comments.

Moreover, it can be observed that, in Thestandard dataset, our method has the best improvement over the baselines in comparison with the other two datasets. This is due to the fact that Thestandard dataset contains longer threads. This means reply relations mostly refer to comments and not the root post.

5 Conclusion

In this paper, the reconstructing thread structure task was explored in blogs and online news agencies. There is a great deal of knowledge in online environments such as forums, chats, blogs. Those environments bring different people with different opinions and sentiments together. However, a large volume of comments with different subjects on a web page creates a lot of complexity in following the actual conversations.

In this paper we introduced a method to automatically reconstruct thread structures on the comments written for blogs and online news agencies. First, we defined some textual and non-textual features and learned a proper model to combine the extracted features using a ranking SVM. Then the model was employed to reconstruct the reply structure in each thread in the test data. The proposed method was called SORTS.

The accuracy of the SORTS method was compared with three baseline algorithms. The results revealed higher accuracy of the SORTS method in comparison with the baseline methods on all datasets.

For future works, we would like to search for new features and try to test our method on other different datasets.

References

1. **Schuth, A., Marx, M., & de Rijke, M. (2007).** Extracting the Discussion Structure in Comments on News-Articles. *9th ACM international workshop on Web information and data management (WIDM'07)*, Lisboa, Portugal, 97–104.
2. **Shen, D., Yang, Q., Sun, J.T., & Chenj, Z. (2006).** Thread detection in dynamic text message streams. *29th Annual International ACM SIGIR conference on Research and Development in Information Retrieval*, Seattle, Washington, 35–42.
3. **Aumayr, E., Chan, J., & Hayes, C. (2011).** Reconstruction of threaded conversations in online discussion forums. *Fifth International AAAI Conference on Weblogs and Social Media ICWSM-11,* Catalonia, Spain, 26–33.
4. **Seo, J., Croft, W.B., & Smith, D.A. (2011).** Online community search using conversational structures. *Information Retrieval*, 14(6), 547–571.
5. **Adams, P.H. & Martell, C.H. (2008).** Topic Detection and Extraction in Chat. *Second IEEE International Conference on Semantic Computing (ICSC '08)*, Santa Clara, California, 581–588.
6. **Wang, Y.C., Joshi, M., Cohen, W.W., & Rosé, C. (2008).** Recovering Implicit Thread Structure in Newsgroup Style Conversations. *2nd International Conference on Weblogs and Social Media (ICWSM II),* Seattle, Washington. 152–160.
7. **Gottipati, S., Lo, D., &Jiang, J. (2011).** Finding relevant answers in software forums. *26th IEEE/ACM International Conference on Automated Software Engineering (ASE 2011),* Lawrence, Massachusetts, 323–332.
8. **Georgiou, T., Karvounis, M., & Ioannidis, Y. (2010).** Extracting Topics of Debate between Users on Web Discussion Boards. *2010 ACM SIGMOD Conference*, Indianapolis, Indiana.
9. **Chan, J., Hayes, C., & Daly, E.M. (2010).** Decomposing Discussion Forums and Boards Using User Roles. *Fourth International AAAI Conference on Weblogs and Social Media*, Washington, DC, 215–218.
10. **Dehghani, M., Asadpour, M., & Shakery, A. (2012).** An evolutionary-based method for reconstructing conversation threads in email corpora. *2012 IEEE/ACM International Conference*

on Advances in Social Networks Analysis and Mining (ASONAM), Istanbul, Turkey, 1132–1137.

11. **Yeh, J.Y. & Harnly, A. (2006).** Email thread reassembly using similarity matching. *Third Conference on Email and Anti-Spam (CEAS 2006)*, Mountain View, California.
12. **Joachims, T. (2006).** Training linear SVMs in linear time. *12th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD'06),* Philadelphia, USA, 217–226.
13. **Robertson, S.E., Walker, S., Jones, S., Hancock-Beaulieu, M.M., & Gatford, M. (1994).** Okapi in TREC-3. *NIST Special Publication 500-226: Overview of the Text Retrieval Conference TREC-3*, Gaithersburg, USA, 109–126.
14. **Wagner, R.A. & Fischer, M.J. (1974).** The String-to-String Correction Problem. *Journal of the ACM*, 21(1), 168–173.
15. **A. Balali, et al. (2013)**. Content Diffusion Prediction in Social Networks. Paper presented at the 5th International Conference on Information and Knowledge Technology (IKT), Shiraz, Iran.

Ali Balali is a M.Sc. student at the Department of Electrical and Computer Engineering at the University of Tehran. He is a member of Natural language & Text Processing and Social Networks laboratories under supervision of Professor Hesham Faili and Professor Masoud Asadpour. His research interests include information extraction and content diffusion prediction on social media.

Hesham Faili received a B.Sc. degree in computer software engineering from Sharif University of Technology, in 1997; an M.Sc. degree in computer software engineering from Sharif University of Technology, in 1999; and a Ph.D. in Artificial Intelligence from Sharif University of Technology, in 2006. He joined the Robotics and AI group in the Faculty of Electrical and Computer Engineering, University of Tehran as a faculty member. His research interests include Natural language & Text Processing and social networks.

Masoud Asadpour was born in Lar, Iran, in 1975. He received the B.Sc. degree in computer software engineering from Sharif University of Technology, Tehran, in 1997; the M.Sc. degree in AI and robotics from the University of Tehran, in 1999; and PhD in Machine Learning and Collective Robotics from EPFL, Switzerland, in 2006. He has been a researcher at Institute for Studies on Theoretical Physics and Mathematics (IPM), Iran, from 1998 to 2001 and again from the begining of 2010 up to now. He has been a post-doc researcher in Biologically Inspired Robotics Group (BIRG), EPFL, Switzerland, in 2007. He is an associate of member of IEEE. In 2008, he joined the Robotics and AI group in the Faculty of Electrical and Computer Engineering, University of Tehran as a faculty member and he found the Social Networks Lab. His research interests are Social Networks, Machine Learning, Bio-Inpsired Computing, and Swarm Intelligence.

Mostafa Dehghani is a M.Sc. student at University of Tehran. He finished his undergraduate studies at Shahrood University of Technology. Currently, he is a member of Intelligent Information Systems Lab under the supervision of Professor Azadeh Shakery and also Social Networks Lab under the supervision of Professor Masoud Asadpour. His research interests lie at the intersection of Data Mining and Social Networks Analysis. In particular, he is interested in email data mining in multilingual environments.

Article received on 07/12/2012; accepted on 16/01/2013.