Toward Conversation Retrieval*

Matteo Magnani[1] and Danilo Montesi[2]

[1] Dept. of Computer Science, University of Bologna
Mura A. Zamboni 7, 40100 Bologna
matteo.magnani@cs.unibo.it
[2] Dept. of Computer Science, University of Bologna
Mura A. Zamboni 7, 40100 Bologna
montesi@cs.unibo.it

Abstract. Social Network Sites can be seen as very large information repositories containing millions of text messages usually organized into complex networks involving users interacting with each other at specific times. In this paper we discuss how traditional information retrieval techniques can be extended to deal with these social aspects. In particular we formalize the concept of *conversation* in the context of Social Network Sites and define constraints regulating ranking functions over conversations.

1 Introduction

Social Network Sites (SNSs) are among the most relevant places where information is created, exchanged and transformed, as witnessed by the number of their users and by their activity during events or campaigns like the terror attack in Mumbai in 2008 or the so-called Twitter revolution in Iran in 2009.

If we look at the kind of interactions happening inside SNSs, these services can be seen as on-line *third places* [1]. Third places, like coffee-bars, are so called because they are distinct from the two usual social environments of home and workplace and are important because they enable specific communication patterns (like a peer conversation between an employer and an employee on topics not related to their working activity) and facilitate the emergence of political ideas and the aggregation of individuals.

SNSs can be seen as very large information repositories with relevant potential applications — from practical areas like politics and marketing to more theoretical fields like social sciences and psychology. Therefore, being able to retrieve information from these repositories is very valuable. However, SNSs are more than just collections of text messages, making the retrieval process not trivial and requiring specific concepts and techniques.

To highlight some of the characteristic features of SNSs we may consider the example of Friendfeed, a well known SNS recently acquired by Facebook, that

* This work has been partially funded by Telecom Italia and by PRIN project *Tecniche logiche e operazionali per interazione tra componenti.*

M. Agosti, F. Esposito, and C. Thanos (Eds.): IRCDL 2010, CCIS 91, pp. 173–182, 2010.

offers features that can be associated both to Twitter, e.g., providing status updates, and Facebook, e.g., creating complex conversations [2]. Using this SNS, users can *follow* each other, post *entries*, *comment* on these entries, and *like* them. This can be represented using the following relational schema:

```
User(ID, Type, Name, Description)
Following(Follower, Followed)
Entry(PostID, PostedBy, Timestamp, Text, Language)
Comment(PostID, EntryRef, PostedBy, Timestamp, Text, Language)
Like(User, EntryRef, Timestamp)
```

This schema represents a much more complex data model than it may appear at a first glance. In Figure 1 we represent an example of a portion of a typical social data structure corresponding to this schema, to highlight its main features — multiple graphs, labeled arcs and unstructured text. From this example, it is easy to see that in SNSs text messages are included into complex structures, where **who** posts something and **when** it has been posted are as important as **what** has been posted. An information extraction process should thus model and exploit these aspects.

More specifically, text messages (or *posts*) are only basic bricks composing more complex and socially relevant **conversations** between communities of users. While it is certainly interesting to analyze each post on its own, it is also very important to be able to manipulate these complex structures, e.g., clustering conversations, retrieving the conversations about a given topic, or understanding the topic of a conversation. Evidently, all these information retrieval capabilities are based on the availability of a model to *rank a set of conversations* with respect to some *information requirements*. In this paper we deal with this problem.

The problem of retrieving information from SNSs has already been addressed in the past, because of its theoretical and practical relevance. In particular,

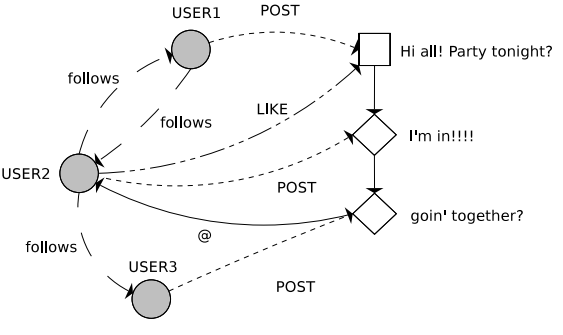

Fig. 1. An example of a social data structure. Users (circles) can follow each other, post text entries, make comments (sometimes directed to specific users through the symbol `@`) and `like` other entries

there is a plethora of tools that can be used to monitor the usage of **keywords** or **tags**, e.g., Twitter Search. However, these tools work on simple text collections and do not consider their structure. Studies in **Structured Information Retrieval** [3], with specific reference to structured documents [4,5,6,7,8] have considered the problem of retrieving parts of documents, but this seems to be different from the organization of Social Network conversations, where we have no overlapping messages *but* connections between them and we should consider user interactions while ranking conversations. Studies in **Hypertext Information Retrieval** [9] and **Web Information Retrieval** have developed methods to consider connections between text documents, like Google's PageRank [10], but these approaches do not include user interaction (e.g., the popularity of the author of a Web page) and do not provide means to compute the aggregate relevance of sequences of text messages into super-structures, like in SNS conversations. At the same time they can certainly be used to compute a user popularity index based on friendship connections, as it will be discussed later.

This paper is organized as follows. In Section 2 we formalize the concept of *conversation* in a Friendfeed- or Facebook-like SNS, starting from simple text interactions between two users and composing them into larger structures. This formalization is necessary to define a ranking function for conversations. In Section 3 we define a set of properties that a ranking function for conversations should satisfy. We conclude the paper with some final remarks.

2 Modeling a Conversation

In Figure 2 we represent the basic components of a communication process. A sender codifies a message and sends it through a communication channel. The message is then decoded by the receiver to interpret it. When we deal with social network data, we know only some of these components, as illustrated in the lower part of the figure. In particular, we know the users exchanging the message and its textual representation. The channel enabling the communication is the Web page of the SNS, and it is constant in our discussion, therefore it will be omitted. In addition, we know the time when the message is posted by the sending user.

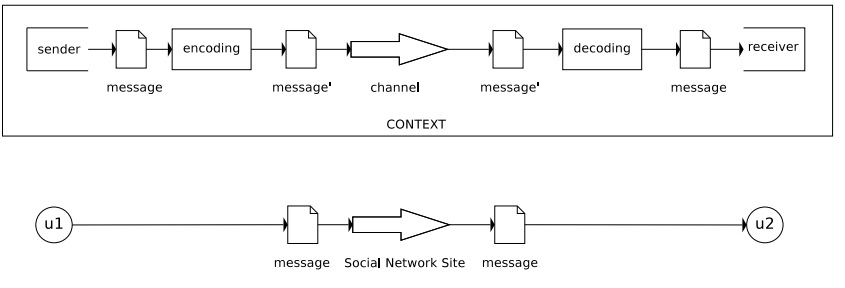

Fig. 2. A graphical representation of a step of a communication process (interaction), and its equivalent in SNS data

Definition 1 (Dyadic interaction). *Let $\mathcal{U}$ be a set of people, $\mathcal{T}$ a set of timestamps, and $\mathcal{M}$ a set of text messages. A dyadic interaction is a tuple (t, u_1, u_2, m), where $t \in \mathcal{T}, u_1, u_2 \in \mathcal{U}$, and $m \in \mathcal{M}$. If $I = (t, u_1, u_2, m)$ is a dyadic interaction, we will notate $\mathrm{ts}(I) = t$ (timestamp of the interaction), $\mathrm{snd}(I) = u_1$ (sender), $\mathrm{rec}(I) = u_2$ (receiver), and $\mathrm{msg}(I) = m$ (text message).*

A dyadic conversation is a chronological sequence of text messages exchanged between two users:

Definition 2 (Dyadic conversation). *A Dyadic conversation is a sequence $(I_1, \ldots, I_n)$, where $\forall i \in [1, n]$ I_i is a dyadic interaction and $\forall i, j \in [1, n], i < j$ $(\mathrm{ts}(I_i) < \mathrm{ts}(I_j))$.*

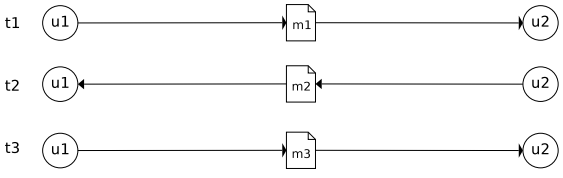

Fig. 3. A graphical representation of a dyadic conversation

In complex social environments we usually experience conversations between sets of users. The concept of dyadic interaction can be easily extended to deal with multiple receivers:

Definition 3 (Polyadic interaction). *We model a polyadic conversation as a tuple (t, u_1, U_2, m), where $t \in \mathcal{T}, u_1 \in \mathcal{U}$, $U_2 \subseteq \mathcal{U}$, and $m \in \mathcal{M}$. If $I = (t, u_1, U_2, m)$ is a polyadic interaction, we will notate $\mathrm{rec}(I) = U_2$.*

It follows that a polyadic conversation is a chronological sequence of text messages exchanged between one sender and a set of receivers, where the people involved may change during the conversation.

Definition 4 (Polyadic conversation). *A polyadic conversation is a sequence $(I_1, \ldots, I_n)$ where $\forall i \in [1, n]$ I_i is a polyadic interaction and $\forall i, j \in [1, n], i < j$ $(\mathrm{ts}(I_i) < \mathrm{ts}(I_j))$.*

3 Ranking Conversations

In this section we define some properties characterizing the functions that can be used to rank a set of conversations.

To rank text messages we can compute their *relevance* with regard to some information requirements, e.g., using a vector space model and a list of keywords.

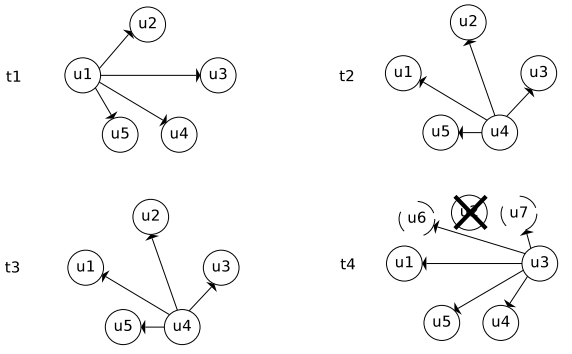

Fig. 4. A graphical representation of a polyadic conversation. Notice that during the conversation the set of receivers may change

However, the same sentence pronounced by two different people will have different degrees of importance — a message from a Prime Minister will probably be more important than a message from one of the authors of this paper, at least in some contexts. In addition, and with absolutely no reference to the previous example, the identity of the speaker may be much more important than what he is saying, making his social interactions very *popular* even when he is not saying anything meaningful. The computation of the importance of a conversation with respect to some information requirements depends also on the users involved in the communication. We will thus use a concept of *popularity* of sender and receivers, depending again on some contextual information. Finally, the same people may exchange the same message, but at different times this may be more or less important — for example, a five-year-old message can be less important than very recent news. Therefore, we will use a measure of the *timeliness* of a social interaction.

3.1 Basic Functions

In the following we assume to know how to compute the relevance of a text message, the popularity of a user and the timeliness of a timestamp. All these functions will take respectively a text message, a user and a timestamp as input, in addition to a context within which the function should be evaluated.

Relevance ($\text{rel} : \mathcal{M} \times \mathcal{C} \rightarrow [0, 1]$, where $\mathcal{C}$ represents our information requirements) can be evaluated using any IR model. However, in our definitions we will require an *independence* property, stating that adding a new message to a conversation does not change the relevance of previous messages. Therefore, in case tf-idf-inspired measures are used, particular attention should be paid to the definition of the idf term. In the context of Social Network Sites, our information requirements can be expressed as a list of keywords, which will be omitted from the following definitions to enhance their readability.

Popularity ($\text{pop} : \mathcal{U} \times \mathcal{C} \rightarrow [0, 1]$) can be defined in several different ways. The popularity of a person depends on the context (e.g., at a public institutional

event, or at a conference on computing), but in the context of a SNS we will typically compute it as a function of her followers (or friends). Also in this case, in the following definitions the context will be assumed to be constant, i.e., the specific SNS, and will not be represented in the equations. As for the relevance, our definitions are based on an independence principle: sending a message to a user does not increase her popularity — an assumption that we may like to relax in further works.

Finally, timeliness ($\text{tml} : \mathcal{T} \times \mathcal{C} \to [0,1]$) of an interaction can be defined in many ways. Considering the average duration of a conversation in SNSs (a few minutes on Friendfeed) we may decide not to consider this measure (e.g., using a constant function) or we could return a result proportional to the input timestamp.

3.2 Ranking Interactions

The properties defined in this subsection constrain a ranking function for social interactions, indicated as *score*, to be monotonic with respect to changes of its constituents. The first property regards the relevance of a text message, and states that if we take an interaction between two people at time t and we substitute the message with a more relevant message, the score of the interaction does not decrease:

$$\text{if } \text{rel}(m) \geq \text{rel}(m') \text{ then } \text{score}((t, u_1, u_2, m)) \geq \text{score}((t, u_1, u_2, m')) \tag{1}$$

Similarly, if we take an interaction and substitute one actor (sender or receiver) with a more popular actor, this will not decrease its score:

$$\text{if } \text{pop}(u_1) \geq \text{pop}(u_1') \text{ then } \text{score}((t, u_1, u_2, m)) \geq \text{score}((t, u_1', u_2, m)) \tag{2}$$

$$\text{if } \text{pop}(u_2) \geq \text{pop}(u_2') \text{ then } \text{score}((t, u_1, u_2, m)) \geq \text{score}((t, u_1, u_2', m)) \tag{3}$$

Finally, if we say something at a different time, when it is more timely, this does not decrease its score:

$$\text{if } \text{tml}(t) \geq \text{tml}(t') \text{ then } \text{score}((t, u_1, u_2, m)) \geq \text{score}((t', u_1, u_2, m)) \tag{4}$$

To compute the score of a polyadic interaction we need to extend the concept of popularity to *sets* of users. By definition, we set the popularity of no users to 0, and the popularity of a singleton to the popularity of the only user in the set:

$$\text{pop}(\{\}) = 0 \tag{5}$$

$$\text{pop}(\{u\}) = \text{pop}(u) \tag{6}$$

Now we can see how to compute the popularity of a set of more than one user. If we add to a set a user with popularity equal or greater (resp. less) than the popularity of the set, the overall popularity will not decrease (resp. increase):

$$\text{if } \text{pop}(u) \geq \text{pop}(U) \text{ then } \text{pop}(U \cup \{u\}) \geq \text{pop}(U) \tag{7}$$

$$\text{if } \text{pop}(u) \leq \text{pop}(U) \text{ then } \text{pop}(U \cup \{u\}) \leq \text{pop}(U) \tag{8}$$

Similarly, if we substitute a user with another one which is more popular, the overall popularity will not decrease:

$$\text{if } \text{pop}(u_i) \geq \text{pop}(u'_i) \text{ then}$$
$$\text{pop}((u_1, \ldots, u_i, \ldots, u_n)) \geq \text{pop}((u_1, \ldots, u'_i, \ldots, u_n)) \tag{9}$$

The ranking of polyadic interactions can now be defined by the same set of rules already defined for dyadic interactions (1, 2, 3 and 4), extended to allow sets of receivers. We will not rewrite these rules, as they are trivial revisions of the previous equations.

3.3 Ranking Conversations

Now, we can use a ranking function for interactions to rank a set of conversations. The first step consists in defining the score associated to an empty conversation as 0, and the score of a conversation with a single interaction as the score of that interaction:

$$\text{score}(()) = 0 \tag{10}$$

$$\text{score}((I)) = \text{score}(I) \tag{11}$$

If we add to a conversation a new interaction whose score is greater (resp. less) than the one of the conversation, the overall score will not decrease (resp. increase):

$$\text{if } \text{score}(I) \geq \text{score}((I_1, \ldots, I_n)) \text{ then}$$
$$\text{score}((I_1, \ldots, I_n, I)) \geq \text{score}((I_1, \ldots, I_n)) \tag{12}$$
$$\text{if } \text{score}(I) \leq \text{score}((I_1, \ldots, I_n)) \text{ then}$$
$$\text{score}((I_1, \ldots, I_n, I)) \leq \text{score}((I_1, \ldots, I_n)) \tag{13}$$

This rule specifies that many irrelevant interactions will reduce the overall score of the conversation. Similarly, if we substitute an interaction with another one with a higher score we will not decrease the score of the whole conversation.

$$\text{if } \text{score}(I_i) \geq \text{score}(I'_i) \text{ then}$$
$$\text{score}((I_1, \ldots, I_i, \ldots, I_n)) \geq \text{score}((I_1, \ldots, I'_i, \ldots, I_n)) \tag{14}$$

It is worth noting that the values of the ranking functions and the combination function may vary significantly, still satisfying these constraints. While tests on real cases may help in choosing effective values, it is our opinion that users should be able to interact with the combination function so that they can specify how much each aspect (relevance, popularity, timeliness) should be weighted for each search task.

4 Conversational Density

The score of a conversation with respect to a given topic (or keyword) is not the only metric to be used to compute its importance. In fact, we may expect

that the comment rate of a message indicates a sort of degree of interest in that message or topic. As an example, consider a passionate political discussion: some people will not wait their turn to speak, and the more the conversation will touch sensitive topics, the more people will increase the frequency of their interactions, starting speaking together.

Unfortunately, SNSs do not give us a simple way to understand the loudness of a message and other fundamental aspects like non-verbal signals [11]. However, conversational density may tell us something more than a single message can.

The fact that the frequency, or density, of interactions during a conversation may tell us something of the content of the message, of its emotional charge, or degree of interestingness, is also suggested by the analysis of real messages. As an example consider Figure 5, where we represent the activity related to a discussion on the Friendfeed SNS using a frequency-modulation chart. Looking at what is happening inside dense parts of this conversation is outside the scope of this paper, but it should be clear that density may be used as an indicator of something that is not otherwise directly stored in the data, which could for example indicate a degree of interest in that topic.

We could even postulate the existence of thresholds of density, so that for example when an on-line conversation exceeds this threshold people can no longer follow all the conversational threads and new sub-topics and sub-conversations spring inside the same chain of comments — a phenomenon that is sometimes perceived by SNS users. However, also this analysis lies outside the scope of this contribution, and we will develop this aspects in future works.

From this discussion, it seems important to be able to model a concept of *density* of a conversation. Intuitively, we can define density as the duration of the conversation divided by the number of interactions, and this is certainly a reasonable choice. However, this would not allow us to highlight the fact that the conversation represented in Figure 5 contains some very dense regions. Therefore, we can consider the usage of alternative density functions.

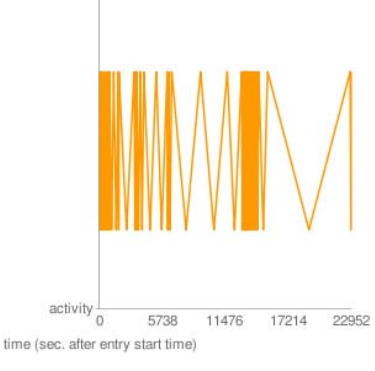

Fig. 5. A graphical representation of different densities during the development of a conversation inside the Friendfeed SNS

When we have less than two interactions, there is not any real conversation, and no concept of density. By definition, we set the density of these conversations to 0.

$$\mathrm{dns}(()) = \mathrm{dns}((I)) = 0 \tag{15}$$

Now consider two equal conversations. If we add an interaction to the first one and we add another interaction with a higher timestamp to the second one, the first conversation will become denser. This concept is illustrated in Figure 6(1).

$$\text{if } \mathrm{ts}(I) \leq \mathrm{ts}(I') \text{ then } \mathrm{dns}((I_1, \ldots, I_n, I)) \geq \mathrm{dns}((I_1, \ldots, I_n, I')) \tag{16}$$

If we put a new interaction inside a conversation, this will obviously increase its density. This is illustrated in Figure 6(2).

$$\mathrm{dns}((I_1, \ldots, I_i, I_{i+1}, I_n)) < \mathrm{dns}((I_1, \ldots, I_i, I, I_{i+1}, I_n)) \tag{17}$$

Finally, we can leave the more intuitive concept of density defined as the duration of the conversation divided by the number of interactions, adding the following rule which favors conversations with variable internal densities over regular conversations. According to this rule, and as represented in Figure 6(3), if we add the same interaction to two different conversations and the first is denser than the second it will be denser also after the addition.

$$\begin{aligned} &\text{if } \mathrm{dns}((I_1, \ldots, I_n)) \geq \mathrm{dns}((I'_1, \ldots, I'_m)) \text{ then} \\ &\quad \mathrm{dns}((I_1, \ldots, I_n, I)) \geq \mathrm{dns}((I'_1, \ldots, I'_m, I)) \end{aligned} \tag{18}$$

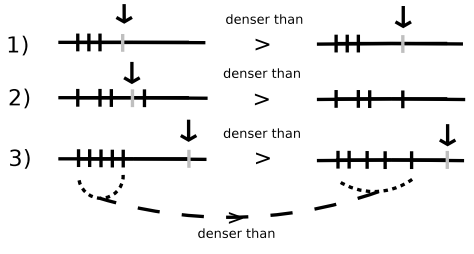

Fig. 6. Effect of adding new interactions (gray vertical lines) to a conversation on its density: vertical lines represent interactions, horizontal lines indicate the time

5 Concluding Remarks

In this paper we have introduced some preliminary aspects of what we call *conversation retrieval*, an information retrieval activity which exploits structural aspects in addition to the exchanged text messages. In particular, we have presented a set of properties that should be satisfied to consider relevance, popularity and timeliness in the computation of the ranking of a conversation. Many alternative functions satisfying these properties can then be used, potentially

leading to different results — we consider this a strength of our proposal, because users may choose between alternative approaches to favor different aspects, e.g., text relevance or popularity.

Moreover, we have discussed the importance of having an additional notion of *density*, which can be potentially used as a metric to associate degrees of interest or other emotional labels to conversations or parts of conversations.

The next step of this research will consist in the implementation of a conversation retrieval system to be tested in real cases.

References

1. Oldenburg, R.: Third Place: Inspiring Stories about the Great Good Places at the Heart of Our Communities. Marlowe & Company (2000)
2. Celli, F., Di Lascio, F.M.L., Magnani, M., Pacelli, B., Rossi, L.: Social Network Data and Practices: the case of Friendfeed. In: International Conference on Social Computing, Behavioral Modeling, & Prediction. Springer, Berlin (2010)
3. Fuhr, N., Rölleke, T.: A probabilistic relational algebra for the integration of information retrieval and database systems. ACM Transactions on Information Systems 15(1), 32–66 (1997)
4. Lalmas, M.: Dempster-Shafer's theory of evidence applied to structured documents: modelling uncertainty. In: Proceedings of the 20th annual international ACM SIGIR conference on Research and development in information retrieval, pp. 110–118. ACM Press, New York (1997)
5. Fuhr, N., Großjohann, K.: XIRQL: A query language for information retrieval in XML documents. In: SIGIR Conference (2001)
6. Amer-Yahia, S., Fernandez, M.F., Srivastava, D., Xu, Y.: Phrase matching in XML. In: Proceedings of the International Conference on Very Large Data Bases (2003)
7. Amer-Yahia, S., Botev, C., Shanmugasundaram, J.: Texquery: a full-text search extension to XQuery. In: WWW (2004)
8. Amer-Yahia, S., Lakshmanan, L.V.S., Pandit, S.: Flexpath: Flexible structure and full-text querying for xml. In: SIGMOD Conference (2004)
9. Agosti, M., Smeaton, A.F.: Information retrieval and hypertext. Kluwer Academic, Boston (1996)
10. Brin, S., Page, L.: The anatomy of a large-scale hypertextual web search engine. In: Computer Networks and ISDN Systems, pp. 107–117 (1998)
11. Watzlawick, P., Bavelas, J.B., Jackson, D.D.: Pragmatics of Human Communication: A Study of Interactional Patterns, Pathologies, and Paradoxes. W. W. Norton and Co. (1967)