

Semantic-based Merging of RSS Items

Fekade GETAHUN, Joe TEKLI, Richard CHBEIR, Marco VIVIANI, Kokou YETONGNON

LE2I Laboratory UMR-CNRS, University of de Bourgogne

Engineer's wing, 9 Savary St., 21078 Dijon Cedex France

{fekade-getahun.taddesse, joe.tekli, richard.chbeir, marco.viviani, kokou}@u-bourgogne.fr

Abstract. Merging XML documents can be of key importance in several applications. For instance, merging the RSS news from same or different sources and providers can be beneficial for end-users in various scenarios. In this paper, we address this issue and explore the relatedness measure between RSS entities/elements. We show here how to define and compute exclusive relations between any two elements and provide several predefined merging operators that can be extended and adapted to human needs. We also provide a set of experiments conducted to validate our approach.

Key words: *RSS, Merging, Document Relatedness, Clustering, Merging Operators*

1 INTRODUCTION

W3C's XML (eXtensible Mark-up Language) [52] is the driving force for representing, exchanging, formatting, interchanging, storing, and filtering data in centralized as well as distributed environments, such as the Web. It is popular because of its semi-structured and self-describing nature which makes it suitable for describing any kind of data.

Really Simple Syndication (RSS) [1][47] is an XML-based family of web feed formats used to publish frequently updated contents such as blog entries, news headlines and podcasts, in a standardized format. RSS has been proposed to facilitate the aggregation of information from multiple web sources. As a result, clients can simultaneously access content originating from different providers (using RSS aggregator) rather than roaming a set of news providers. Clients can subscribe to news they are interested in, using RSS aggregators. RSS aggregators download news feeds and provide an interface to view and organize them. When Clients add more sources the amount of news feeds becomes more difficult to manage. Often, clients have to read related (and even identical) news more than once as the existing RSS engines do not provide facilities for merging related items.

Merging RSS news can thus be fruitful in several applications and scenarios. This comes down to (i) identifying related elements between two news, and (ii) generating a merged document that collapses related elements while preserving remaining source elements. With respect to (w.r.t.) the first sub-problem, merging RSS news requires identifying the *relatedness*¹ [4] between their elements, i.e., element labels and contents, and consecutively element semantic overlapping, intersection, inclusion and disjointness (considering the meaning of terms and not only their syntactic properties).

In this work, we address these problems. We first focus on the issue of measuring relatedness and relationship between RSS elements/items, as a necessary prerequisite to performing efficient RSS merging. After, we propose merging operators that consider the relatedness and relationship values

¹ Semantic relatedness is a more general concept than similarity; similar entities are semantically related by virtue of their similarity, but dissimilar entities may also be semantically related by lexical relationships such as meronymy and antonymy, or just by any kind of functional relationship or frequent association.

in performing the merge operation. Hence, the main objective of this study is to put forward a specialized and human-oriented RSS relatedness measure able to:

- i) Quantify RSS relatedness, necessary for identifying and grouping RSS items that are related (similar) enough to be merged (the merging of unrelated items is obviously useless to the user).
- ii) Identify the relationship that can occur between two RSS items (i.e., *disjointness*, *intersection*, *inclusion* and *equality*), to be exploited in the merging phase.
- iii) Merging RSS items based on item relatedness (grouping) and relationship identification. Grouping similar items, and Identifying their common/different parts would help decide on the merging rules to be executed (e.g., if RSS item I_1 is included in I_2 , then merging I_1 and I_2 simply comes down to preserving I_2).

The remainder of this paper is organized as follows. Section 2 presents our motivation scenarios. Section 3 presents an overview of our merging framework. Section 4 presents some preliminary definitions and notions to be exploited in defining our RSS relatedness measure. Section 5 presents our relatedness measure detailing semantic relatedness and relationship between texts, simple elements and items. Section 6 presents relation aware clustering algorithm. In Section 7, we details our basic merging operators and the related merging algorithm. Section 8 presents experimental results. In Section 9, we discuss background and related work. Section 10 concludes the paper and draws future research directions.

2 MOTIVATING SCENARIO

The number of applications using RSS is increasing everyday: *AmphetaDesk*², *PullRss*³, *Radio UserLand*⁴, *SlashCode/Slashdo*⁵, *Weblog 2.0* [1]. In addition, RSS has been used in *ELF*⁶ and *Package tracking*⁷ to notify customers about events or news. Most of these applications are designed to aggregate, search, filter or display news in RSS specification. However, to the best of our knowledge, none of these considers the issue of merging related news collected from same or different news providers.

To motivate our work, let us consider Figure 1 and Figure 2 showing a list of news extracted from CNN and BBC's RSS feeds. Registering these feeds in existing news readers such as *Newsgator*, *Google Reader*, *Attensa*, provides the user with access to all news without considering *relatedness* among them. However, identifying and merging related news would enable the user to more easily and efficiently acquire information. The user would obviously prefer to access one piece of news about a certain topic, encompassing all relevant and related information (after merging), instead of searching and reading all news articles covering the same topic, which could be extremely time consuming and often disorienting. XML news feeds (e.g., RSS items), and particularly XML elements, can be related in different manners.

- The content of an element might be similar and totally included in another (inclusion).

Example 1. The title content of CNN1 “U.N. chief launches \$613M Gaza aid appeal” includes the title content of BBC1 “UN launches \$613m appeal for Gaza”⁸.

- Two news may refer to similar and related concepts (intersection).

Example 2. The description content of CNN2 “Ford Motor reported that its ongoing losses soared in the fourth quarter, but the company reiterated it still does not need the federal bailout already received by its two U.S. rivals.” and description content of BBC2 “US carmaker Ford reports the biggest full-year loss in its history, but says it still does not need government loans.” are related and very similar, they share some words/expressions (‘Ford’, ‘report’, ‘loss’, ‘US’) and semantically related concept (‘fourth quarter’, ‘year’), (‘biggest’, ‘soar’), (‘reiterate’, ‘say’), (‘federal bailout’, ‘government loan’).

² AmphetaDesk is a free, cross platform, open-sourced, syndicated news aggregator <http://www.disobey.com/amphetadesk/>

³ PullRSS is a template-based RSS to HTML converter, with optional redirects.

⁴ <http://radio.userland.com/userGuide/reference/aggregator/newsAggregator>

⁵ <http://slashdot.org/>

⁶ <http://libraryelf.com/>

⁷ <http://www.simpletracking.com/>

⁸ After a pre-process of stop word removal, stemming, ignoring non textual values and semantic analysis

| | |
|---|------|
| <CNN_RSS> | |
| <item> | |
| <title>U.N. chief launches \$600M Gaza aid appeal</title> | |
| <guid>http://edition.cnn.com/2008/WORLD/asiapcf/05/02/oly.hk.torch/index.html?eref=edition</guid> | |
| <link>http://edition.cnn.com/2008/WORLD/asiapcf/05/02/oly.hk.torch/index.html?eref=edition</link> | CNN1 |
| <description>United Nations Secretary-General Ban Ki-moon on Thursday launched a humanitarian appeal to provide emergency aid to the people of Gaza in the aftermath of Israel's military offensive in the region.</description> | |
| <pubDate>Fri, 02 January 2009 02:56:47 EDT</pubDate> | |
| </item> | |
| <item> | |
| <title>Ford reports \$5.9 billion loss in the fourth-quarter</title> | |
| <description>Ford Motor reported that its ongoing losses soared in the fourth quarter, but the company reiterated it still does not need the federal bailout already received by its two U.S. rivals.</description> | CNN2 |
| </item> | |
| <item> | |
| <title>The youth forum cancels scheduled demonstration</title> | |
| <description>The international youth forum cancels the call for stop-war demonstration due to security reason</description> | CNN3 |
| </item> | |
| <item> | |
| <title>Al-Jazeera: Cameraman home from Gitmo</title> | |
| <guid>http://edition.cnn.com/2008/WORLD/americas/05/01/gitmo.journalist/index.html?eref=edition</guid> | |
| <link>http://edition.cnn.com/2008/WORLD/americas/05/01/gitmo.journalist/index.html?eref=edition</link> | CNN4 |
| <description>Al-Jazeera cameraman Sami al-Hajj has been released after nearly six years in the U.S. Navy prison at Guantanamo Bay, Cuba, a senior Pentagon official aware of the details of the release told CNN on Thursday.</description> | |
| <pubDate>Thu, 01 May 2008 21:51:15 EDT</pubDate> | |
| </item> | |
| </CNN_RSS> | |

Fig. 1. RSS news extracted from CNN

- News might have different or slightly different titles but refer to almost the same issues (relatedness between different elements of the same items).

Example 3. Title content of CNN4 “Al-Jazeera: Cameraman home from Gitmo” and Title content of BBC4 “Freed Guantanamo prisoner is home”. These titles share little commonalities (“home” and “Guantanamo”⁹). However, the contents of corresponding news items are similar.

The examples demonstrate the need to consider several issues; First, one can realize that existing XML-related (xSim[27]), flat text-related similarity (tf-idf), or correlation-based phrase matching [38] approaches (cf. Section Background) cannot be exploited in comparing RSS items since they do not identify the *disjointness*, *inclusion*, *intersection* and *equality* relationships (cf. examples 1, 2 and 3), which are preliminary for the merging process. Second, when comparing two RSS items, computing relatedness between contents of elements having identical labels is not enough to identify overall item relatedness (cf. example 3).

It is to be noted that doing this is complex as the quality of textual information is dependent on the author’s style of writing and use of words, nouns, verbs, (identical topics might be described differently, while different topics might be described using similar concepts).

In the context of RSS merging, there is a need to develop a measure for comparing RSS items, which considers the different contents of RSS elements all together, identifying along with the relatedness, the type of relationship between the items being compared. Detecting relationships between items is crucial to the merging phase (as explained in the introduction) for defining the

⁹ “Gitmo” indicates the Guantanamo prison.

merging rules/operators and consequently performing the merging task. Note that in examples given above, for the sake of simplicity, we only discuss the relationship between the <title> elements of both items. Nonetheless, when performing merging, the system must check the relationship between both RSS items as a whole, i.e., the relationships between all their elements (which will be thoroughly developed in the remainder of the paper), to decide on the merging rule to be utilized. In our examples for instance:

```

<BBC_RSS>
<item>
<title> UN launches $613m appeal for Gaza </title>
<description> The UN will launch an appeal for $613m to help people affected by
Israel's military offensive in Gaza, the body's top official says </description>
<guid isPermaLink="false"> http://news.bbc.co.uk/go/rss/-/2/hi/me/723378828.stm BBC1
</guid>
<link> http://news.bbc.co.uk/go/rss/-/2/hi/americas/7378828.stm </link>
<pubDate>Fri, 02 January 2009 02:56:47 GMT</pubDate>
<category>Middle-east</category>
</item>
<item>
<title> Ford reports record yearly loss </title>
<description> US carmaker Ford reports the biggest full-year loss in its history, but BBC2
says it still does not need government loans.</description>
</item>
<item>
<title>Youth's form call for demonstration</title>
<description> International youth forum call demonstration as part of stop the war BBC3
</description>
</item>
<item>
<title>Freed Guantanamo prisoner is home</title>
<description>A cameraman from the al-Jazeera TV station freed from Guantanamo
Bay has arrived home in Sudan.</description>
<link>http://news.bbc.co.uk/go/rss/-/2/hi/americas/7378828.stm</link> BBC4
<guid isPermaLink="false">http://news.bbc.co.uk/2/hi/americas/7378828.stm</guid>
<pubDate>Fri, 02 May 2008 04:08:38 GMT</pubDate>
<category>Americas</category>
</item>
</BBC_RSS>

```

Fig. 2. RSS news extracted from BBC

- Merging items *CNN1* and *BBC1* related via the intersection relationship could be undertaken by merging sub-elements of *CNN1* and *BBC1* as shown below in Figure 3.

```

<item>
<title>U.N. chief launches $600M Gaza aid appeal</title>
<description>United Nations launched a appeal to aid to the people of Gaza in the of Israel's military
offensive | $613m affected offensive, the body's top official says Secretary-General Ban Ki-moon aftermath
humanitarian provide emergency in the region </description>
</item>

```

Fig. 3. Merging of *CNN1* and *BBC1*.

- Merging *CNN2* and *BBC2*, *CNN4* and *BBC4* could be done similarly to that of *CNN1* and *BBC1* (*intersection*). Nonetheless, this example explicitly reflects the importance of analyzing the relationships between each of the RSS item elements independently, prior to deciding on the merging rule.

The objective of this paper is to address relatedness between items, elements and text values and identifying the merging operators that make use of the identified relationships.

3 OVERVIEW

Our framework for merging RSS items is depicted in Figure 4. It consists of four main interacting modules: (i) Pre-processing, (ii) Relatedness, (iii) Clustering and (iv) Merging. The knowledge component is a plug-in dedicated to handle and manage domain and application dependent external information. It encompasses 1) a value knowledge base exploited in evaluating text content relatedness, 2) a label knowledge base used in evaluating element label relatedness, and 3) users profile and merging preferences. When the system starts for the first time, the user would provide an initial profile that includes a list of tags that would be used as content descriptor. These tag names would be utilized while measuring item/element relatedness. For example, a user may want to use `title` and `description` elements, and another might want to use all tags in measuring relatedness. The user-based merging rules would be defined by combining templates containing each of the predefined actions to be undertaken with each selected relation.

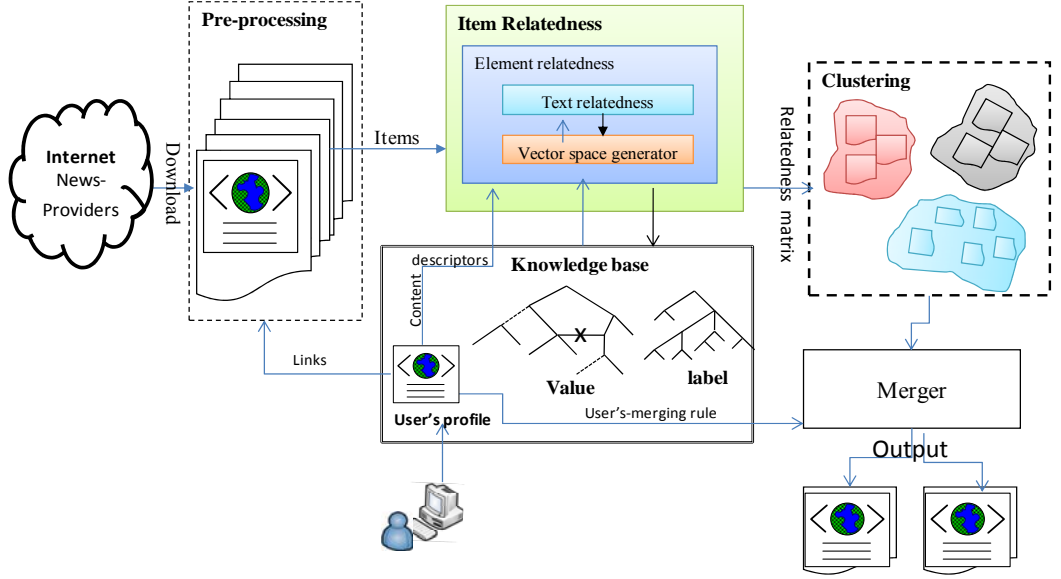


Fig. 4. Overall architecture of our RSS merging framework.

3.1 Pre-processing module

The *Pre-processing* module accesses the knowledge base to get the URL of registered RSS providers. Then it accesses the internet and downloads corresponding RSS feeds. The module checks their well-formedness, returning a collection of items, to be treated by the *Relatedness* module.

3.2 Relatedness module

The *Relatedness* module accepts a list of extracted items as input from the pre-processing module and computes text/elements/items relatedness respectively. The module accesses the content descriptor of RSS news articles based on the user profile. Unlike the works in [24][38][40], the content descriptor is decided and provided by a user while building the initial profile and could be changed later. In computing text relatedness, the *vector space model* is used to store the enclosure weight of each concept (which are found after applying stop word removal and stemming as it will be explained in the following sections) extracted from the content descriptor, and relatedness is computed using a vector based similarity measure (e.g., cosine). Unlike the approach in [40], the relatedness between elements takes into consideration both their labels (i.e. tag names) and text contents. In addition, the module identifies the various relationships that exist between text values, elements and items as detailed in Section 5.

3.3 Clustering Module

The *Clustering module* puts together related (similar) items based on relatedness results provided by the *Relatedness module*. The aim of this module is to facilitate the merging process. It is developed in Section 6. Existing clustering algorithms [10][25] group highly related documents/items. Applying such algorithms in our RSS context would result in grouping highly intersecting news in the same cluster, disregarding item relationships. In other words, those news items related with the inclusion relationship for instance, and having lesser relatedness/similarity scores, would be put in different clusters, which is not appropriate. Such items should be obviously put together in the same cluster. As result, we propose a relationship-aware clustering algorithm in order to consider the various kinds of relationships, in particular inclusion, while performing clustering.

3.4 Merging module

The *Merging module* uses the result of the *Clustering module* in order to abridge grouped together elements according to predefined merging rules (taking into account element relationships) and user preferences. In Section 7, the basic merging operators are defined taking into consideration relationships existing between items and elements. The merging of related items depends on the merging rules provided by the user and stored in the corresponding profile. The module produces RSS documents authored with the RSS 2.0 specification and can be read with any RSS aggregator.

4 PRELIMINARIES

In this section, we present some basic definitions and notions to be exploited in defining our RSS relatedness measure.

4.1 RSS (XML) data model

An XML document is a *hierarchically structured* and *self-describing* piece of information consisting of *atomic* or *complex* elements (elements with nested sub-elements). It is represented as a *rooted ordered labeled tree* following DOM (Document Object Model) [52]. An RSS item comes down to an XML document that is well-formed w.r.t. an RSS schema. Note that different RSS schemas exist, corresponding to the different versions of RSS available on the web (RSS 0.9x¹⁰ (x=1 or 2), 1.0¹¹, and 2.0). Nonetheless, our analysis of the different versions of RSS showed that RSS items consistently follow the same overall structure, adding or removing certain elements depending on the version at hand (for instance element source is part of RSS 0.9x whereas guid is in RSS 2.0).

Definition 1. [Rooted Ordered Labeled Tree]

A *rooted ordered labeled tree*¹² T is a set of $(k + 1)$ nodes $\{r, n_i\}$, with $i = 1, \dots, k$. The children of each node are ordered. The root of T is r and the remaining nodes n_1, \dots, n_k are partitioned into m sets T_1, \dots, T_m , each of which is a tree. These trees are called sub-trees of the root of T . The RSS tree depicting item *CNN1* of Figure 1 is shown in Figure 5.

Definition 2. [Element]

Each node of the rooted labeled tree T is called an *element* of T . Each element e is a pair $e = \langle \eta, \varsigma \rangle$ where $e.\eta$ refers to the element name and $e.\varsigma$ to its content. $e.\eta$ generally assumes an atomic text value (i.e., a single word/expression) whereas $e.\varsigma$ may assume either an atomic text value, a composite text value (sentence, i.e., a number of words/expressions), or other elements¹³.

¹⁰ RSS 0.92 is upward compatible to RSS 0.91, Userland specification <http://backend.userland.com/rss09x>

¹¹ RSS 1.0 is also called RDF Site summary, it is a lightweight multipurpose extensible metadata description and syndication format conforms to the W3C's RDF Specification and is extensible via XML-namespace and/or RDF based modularization. <http://web.resource.org/rss/1.0/spec>

¹² In the rest of the paper, the term *tree* means *rooted ordered labeled tree*.

¹³ We do not consider *attributes* in evaluating RSS item relatedness since they do not affect the semantic comparison process. Nonetheless, attributes will be considered in the merging phase.

Definition 3. [Simple/Composite Element]

An element e is *simple* if $e.\varsigma$ assumes either an atomic or composite textual value¹⁴. In XML trees, simple elements come down to leaf nodes.

For instance, `<title> U.N. chief launches $600M Gaza aid appeal </title>` of RSS item *CNN1* is a simple XML element having $e.\eta = \text{"title"}$ and $e.\varsigma = \text{"U.N. chief launches $600M Gaza aid appeal"}$.

An element e is *composite* if $e.\varsigma$ assumes other elements. In XML trees, composite elements correspond to inner nodes.

Element `<item> <title> U.N. chief launches $600M Gaza aid appeal </title> <guide>... </item>` of *CNN1* is composite.

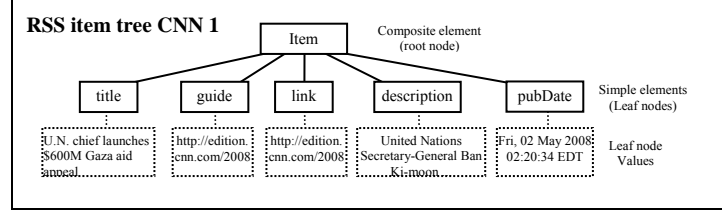


Fig. 5. Tree representation of RSS item CNN1 in Figure 1.

Definition 4. [RSS Item Tree]

An *RSS item tree* is an XML tree T having one single composite element, the root node r (usually with $r.\eta = \text{'item'}$), and k simple elements $\{n_1, \dots, n_k\}$ describing the various RSS item components.

4.2 Knowledge Base

Knowledge Bases (KB) [46] (thesauri, taxonomies and/or ontologies) provide a framework for organizing entities (words/expressions, generic concepts, web pages, etc.) into a semantic space. In our study we formally define a knowledge base as $KB=(C, E, R, f)$ where:

- C is the set of synonymous words/terms/expressions (synonym sets as in WordNet [44]).
- E is the set of edges connecting the concepts, where $E \subseteq C \times C$.
- R is the set of semantic relations, $R = \{\equiv, \prec, \succ, \ll, \gg, \Omega\}$, the synonymous term/words/expressions being integrated in the concepts.
- f is a function designating the nature of edges in E , $f:E \rightarrow R$.

The symbols in R underline respectively the synonym (\equiv), hyponym (Is-A or \prec), hypernym (Has-A or \succ), meronym (Part-Of or \ll), holonym (Has-Part or \gg) and Antonym (Ω) relations, as defined in [9].

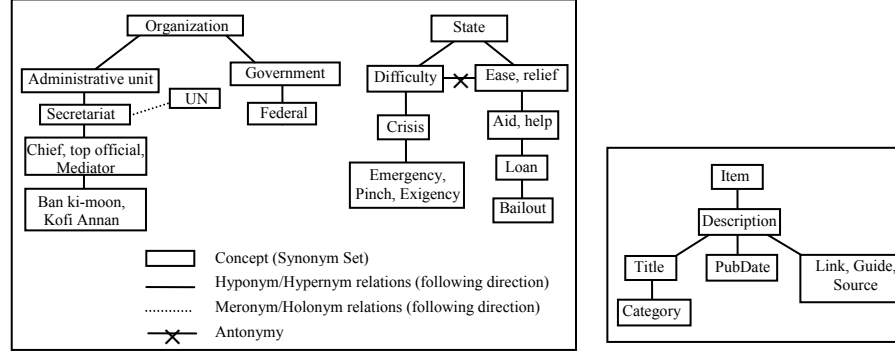
The use of application dependent knowledge bases (KB) facilitates and improves the relatedness result. To that end, we introduce two knowledge bases: (i) *value-based*: to describe the textual content of RSS elements, and (ii) *label-based*: to organize RSS labels. Note that one single knowledge base could have been used. However, since RSS labels might belong to different versions and can be defined by applications or users, following a user defined document schema, an independent *label-based* knowledge base, provided by the user/administrator, seems more appropriate than a more generic one such as WordNet [44] (adequate for treating generic textual content).

4.2.1 Neighborhood

In our approach, the *neighborhood* of a concept C_i underlines the set of concepts $\{C_j\}$, in the knowledge base, that are subsumed by C_i w.r.t. a given semantic relation. Concept *neighborhood* is exploited in identifying the relationships between text (i.e., RSS element labels and/or textual contents) and consequently RSS elements/items. In [9], the authors use the *neighborhood* concept to identify implication between textual values, operators (e.g., $=Any$, $>Some$, $Like$, ...), and

¹⁴ In this paper, we do not consider other types of data contents, e.g., numbers, dates, ...

consequently semantic predicates (e.g., predicate *Location*="Paris" implies *Location Like* "France"). In this paper, we extend this approach and adopt three types of *neighborhoods*.



a. Samples value knowledge base - VKB, with multiple root concepts, extracted from WordNet.

b. Sample RSS label knowledge base - LKB

Fig. 6. Sample value and label knowledge bases.

Definition 5. [Semantic Neighborhood]

The *semantic neighborhood* of a concept C_i with semantic relation R is defined as the set of concepts $\{C_j\}$ related with C_i via R directly or via transitivity¹⁵. R is restricted to synonymy (\equiv), hyponymy (\prec), or meronymy (\ll). It is formally defined as:

$$N_{KB}^R(C_i) = \{C_j | C_i R C_j\}, R \in \{\equiv, \prec, \ll\} \quad (1)$$

For instance, referring to the value knowledge base VKB in Figure 6.a we have:

$$N_{VKB}^{\equiv}(emergency) = \{emergency, pinch, exigency\}$$

$$N_{VKB}^{\prec}(emergency) = \{emergency, crisis, difficulty, state\}$$

$$N_{VKB}^{\ll}(emergency) = \emptyset$$

Definition 6. [Global Semantic Neighborhood]

The *global semantic neighborhood* of a concept is the union of the semantic neighborhood defined with the synonymy (\equiv), hyponymy (\prec) and meronymy (\ll) semantic relations altogether. Formally:

$$\overline{N_{KB}}(C_i) = \bigcup_{R \in \{\equiv, \prec, \ll\}} N_{KB}^R(C_i) \quad (2)$$

For instance, referring to the value knowledge base VKB in Figure 6.a

$$\overline{N_{VKB}}(emergency) = N_{VKB}^{\equiv}(emergency) \cup N_{VKB}^{\prec}(emergency) \cup N_{VKB}^{\ll}(emergency)$$

$$\overline{N_{VKB}}(emergency) = \{emergency, pinch, exigency, crisis, difficulty, state\}$$

4.3 Text Representation

In this Section, we define the notion of concept sets and text model that is basic to represent a piece of text. Later it would be exploited while computing text content relatedness.

Definition 7. [Concept Set]

Given a text T (i.e., phrase, sentence, etc.), its *concept set* is denoted as CS , is a set of concepts $\{C_1, \dots, C_m\}$, where each C_i represents a concept. Each concept C_i is assumed to be obtained after

¹⁵ The relationship between operators including indirect transitivity has been studied in the work of getahun et al [9].

several textual pre-processing operations such as stop-words removal¹⁶, stemming¹⁷, and/or mapping to the value knowledge base, and grouping.

For instance, the content of title element from RSS item *CNN1* “U.N. chief launches \$613M Gaza aid appeal” can be described by the following concept set: $CS_{CNN1} = \{\{UN\}, \{chief\}, \{launch\}, \{Gaza\}, \{aid\}, \{appeal\}\}$. Likewise, the concept set for content of title element from RSS item *CNN2* “UN launches \$613m appeal for Gaza” is described as $CS_{CNN2} = \{\{UN\}, \{launch\}, \{appeal\}, \{Gaza\}\}$.

Formally, let T_1 and T_2 be two textual contents, CS_1 is the concept set describing T_1 and CS_2 that describing T_2 .

Definition 8. [Deep In Membership]

Given a concept c_i and concept set CS_i , c_i is deep in CS_i , i.e., $c_i \stackrel{d}{\in} CS_i$ if c_i exists as member of a concept in CS_i .

For example, gaza exists deep in the concept set $CS_{CNN1} = \{\{UN\}, \{chief\}, \{launch\}, \{Gaza\}, \{aid\}, \{appeal\}\}$

Definition 9. [Text model]

Given two texts T_1 and T_2 described by concept set CS_1 and CS_2 , we represent each t_i as a vector V_i in an n -dimensional space as: $V_i = [\langle C_1, w_1 \rangle, \dots, \langle C_n, w_n \rangle]$. The vector space dimensions represent distinct concepts $C_m \in CS_1 \cup CS_2$ associated to weight w_m such as $1 \leq m \leq n$ where $n = |CS_1 \cup CS_2|$.

The weight w_m associated to a concept C_m in V_i (where $i=1$ or 2) is calculated as $w_m = 1$ if the concept C_m is deep in the concept set of the one of the texts (i.e., CS_i) that constitute the vector V_i ; otherwise, it is computed based on the highest *enclosure similarity* it has with another concept C_j from the concept set of the other text. Formally, it is defined as:

$$w_m = \begin{cases} 1 & \text{if } C_m \stackrel{d}{\in} CS_i \\ \max \left(Enclosure_Sim(C_m, C_j) \right) \wedge C_j \stackrel{d}{\in} CS_j & \text{otherwise} \end{cases} \quad (3)$$

$$Enclosure_Sim(C_m, C_j) = \frac{|\overline{N_{KB}}(C_m) \cap \overline{N_{KB}}(C_j)|}{|\overline{N_{KB}}(C_j)|} \quad (4)$$

$Enclosure_sim(C_m, C_j)$ takes into account the global semantic neighborhood of each concept. It is asymmetric, allows the detection of the various kinds of relationships between RSS items, and returns a value equal to 1 if C_i includes C_j , and zero if there is no concept related with C_m .

Example 4. Let us consider the *description* elements of RSS items *CNN2* and *BBC2* (Figures 1, 2). Corresponding vector representations V_1 and V_2 are shown in Figure 7. For the sake of simplicity, we consider that only these two texts make up the new items.

| | <i>Ford</i> | <i>report</i> | <i>loss</i> | ... | <i>reiterate</i> | <i>Fourth-quarter</i> | <i>Federal</i> | <i>Bailout</i> | <i>Big</i> | <i>Full-year</i> | <i>say</i> | <i>government</i> | <i>loan</i> |
|-------|-------------|---------------|-------------|-----|------------------|-----------------------|----------------|----------------|------------|------------------|------------|-------------------|-------------|
| V_1 | 1 | 1 | 1 | ... | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| V_2 | 1 | 1 | 1 | ... | 1 | 0 | 0.67 | 0.86 | 1 | 1 | 1 | 1 | 1 |

Fig. 7. Vectors obtained when comparing title texts of RSS items CNN2 and BBC2

Vector weights are evaluated in two steps. First, for each concept C_i in V_1 and V_2 , we check the existence of C_i in each of the concept sets corresponding to the texts being compared. Second, we update the weight of those concepts having value of zero with maximum semantic enclosure similarity value. Following the WordNet subset extract in Figure 6.a, the concept *Government* is included in the global semantic neighborhood of *Federal*, i.e., $government \in \overline{N_{KB}}(federal)$. Hence, it has the maximum enclosure similarity with *federal*, i.e., $Enclosure_sim(federal, government) = 1$. However, in V_2 , *federal* is highly related with *government* and its $Enclosure_sim(government, federal) = 0.67$.

¹⁶ Stop-words identify words/expressions which are filtered out prior to, or after processing of natural language text (e.g., *a*, *an*, *so*, *the*, ...) which is done using stop list. However those words that would change the meaning of the text such as *but*, *not*, *neither*, *nor* ... are not considered as stop words.

¹⁷ Stemming is the process for reducing inflected (or sometimes derived) words to their stem, i.e., base or root (e.g., “*housing*”, “*housed*” → “*house*”).

Likewise, *loan* is included in the global semantic neighborhood of *bailout*, i.e., $loan \in \overline{N_{KB}}(bailout)$.

This way $Enclosure_sim(loan, bailout) = 1$ and $Enclosure_sim(bailout, loan) = 0.86$.

Notice that the computation of enclosure similarity (*enclosure_sim*) is based on maximum similarity value and takes into consideration concepts related with equality, inclusion, intersection and disjointness relationship.

5 RSS RELATEDNESS AND RELATIONS

Quantifying the semantic relatedness and identifying the relationships between two RSS items amounts to comparing corresponding elements. This in turn comes down to comparing corresponding RSS (simple) element labels and contents, which simplifies to basic pieces of text (cf. Definition 2). Hereunder, we define the two basic concepts used in our relatedness measures before explaining text, sub-element and item relatedness.

Definition 10. [SemRel]

SemRel refers to the semantic relatedness between two texts, simple elements or items. It has value between 0 and 1 inclusive.

Definition 11. [Relation]

Relation refers to the exclusive relationship that would exist between two texts, simple elements or items. We identify equality, intersection, inclusion, or disjoint relations.

In the following sub-sections, we identify relatedness between texts, simple elements and/or items having a tuple containing *SemRel* and *Relation* values and denoted as :

$$Relatedness = \langle SemRel, Relation \rangle \quad (5)$$

5.1 Text Relatedness

Given two texts T_1 and T_2 , the *Text Relatedness* (*TR*) algorithm shown in Algorithm 1 returns a tuple, combining *SemRel* and *Relation* values between T_1 and T_2 .

The algorithm accepts two texts T_1 and T_2 as input (line 1). Corresponding concept sets CS_1 and CS_2 are generated using a function f (lines 9 – 10) encompassing Natural Language Processing (NLP) and mapping (i.e. associating non-stop and stemmed words into corresponding knowledge base concept) features. In lines 11 – 16, texts T_1 and T_2 are represented as a vector V (V_1 and V_2 respectively) with weights underlining concept existence, and enclosure in both CS_1 and CS_2 . The procedure *weight* accept concept whose weight to be computed C_i , concept set of text T_1 or T_2 and concept set of T_2 or T_1 , and returns result (following formula 3). In line 17, the semantic relatedness *SemRel* between two texts is quantified using a measure of similarity between vectors V_1 and V_2 implemented in *Vector-Base-Similarity-Measure* function. In this study, we use the *cosine measure*:

$$SemRel = SemRel(T_1, T_2) = Cos(V_1, V_2) = \frac{V_1 \cdot V_2}{|V_1| \times |V_2|} \in [0,1] \quad (6)$$

Semantic relatedness is consequently exploited in identifying basic relations (i.e., *disjointness*, *intersection* and equality) between texts. Our method (Relation in line 19-29) for identifying basic relationships is based on a *fuzzy logic* model to overcome the often imprecise descriptions of texts. For instance, texts (likewise RSS items) that describe the same issue are seldom exactly identical. They might contain some different concepts, detailing certain specific aspects of the information being described, despite having the same overall meaning and information substance (cf. Section 1, Example 1). Thus, we address the fuzzy nature of textual content in identifying relations by providing pre-defined/pre-computed similarity thresholds $T_{Disjointness}$ and $T_{Equality}$, as shown in Figure 8.

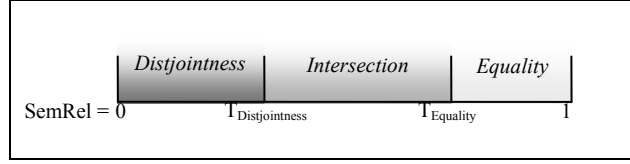


Fig. 8. Basic text relationships and corresponding thresholds.

Thus, we suggest using the following rules to identify the basic disjointness, Intersection or Equality relationships between two texts T_1 and T_2 .

$$\begin{aligned}
 & \text{Relation}(T_1, T_2) \\
 = & \begin{cases} \text{Disjointness, i.e., } T_1 \not\supset T_2 \Rightarrow & \text{SemRel}(T_1, T_2) \leq T_{\text{Disjointness}} \\ \text{Equality, i.e., } T_1 = T_2 \Rightarrow & \text{SemRel}(T_1, T_2) \geq T_{\text{Equality}} \\ \text{Intersection, i.e., } T_1 \cap T_2 \Rightarrow & T_{\text{Disjointness}} < \text{SemRel}(T_1, T_2) < T_{\text{Equality}} \end{cases} \quad (7)
 \end{aligned}$$

While the basic *disjointness*, *intersection* and *equality* relations can be defined based on semantic relatedness (in the context of fuzzy relations), this is not the case for *inclusion* relation, which we define as:

- **Relation(T_1, T_2) is *Inclusion***, i.e., $T_1 \supset T_2$, if the product of the weights of vector V_1 (describing T_1) is equal to 1, i.e.,

$$\text{Relation}(T_1, T_2) = \text{Inclusion, i.e., } T_1 \supset T_2 \Rightarrow \prod_{V_1} (w_p) = 1 \quad (8)$$

Where $\prod_{V_1} (w_p)$ is the weight product of vector V_1 (describing T_1) underlines whether or not T_1 encompasses all concepts in T_2 .

Notice that the relationship between text values is identified on the bases of best relation value which is equality, inclusion, disjoint and intersect (line 19-29) respectively.

| Algorithm 1: TR Algorithm | | |
|----------------------------------|--|--|
| 1. | Input: T_1, T_2 : String | // two input texts |
| 2. | $T_{\text{Disjoint}}, T_{\text{Equality}}$: decimal | // threshold values |
| 3. | Variable: V_1 : vector | // vector for t_1 |
| 4. | V_2 : vector | // vector for t_2 |
| 5. | CS_1 : Set | // concept set of t_1 |
| 6. | CS_2 : Set | // concept set of t_2 |
| 7. | Output: <i>SemRel</i> : Double | //relatedness value between t_1, t_2 |
| 8. | <i>Rel</i> : string | //topological relationships between t_1, t_2 |
| 9. | Begin | |
| 10. | $CS_1 = f(T_1)$ | // f – returns the concept set of the text T_1 |
| 11. | $CS_2 = f(T_2)$ | // f – returns the concept set of the text T_2 |
| 12. | $C = CS_1 \cup CS_2$ | |
| 13. | $V_2 = V_1 = \text{Vector_Space_Generator}(C)$ | // generate vector space having C as concepts |
| 14. | For each C_i in C | |
| 15. | $V_1[C_i] = \text{Weight}(C_i, CS_1, CS_2)$ | // computes the weight of concept C_i in V_1 |
| 16. | $V_2[C_i] = \text{Weight}(C_i, CS_2, CS_1)$ | // computes the weight of concept C_i in V_2 |
| 17. | Next | |
| 18. | $\text{SemRel} = \text{Vector-Base-Similarity-Measure}(V_1, V_2)$ | //cosine similarity is used in our implementation |
| 19. | If $\text{semRel} \geq T_{\text{Equality}}$ then | |
| 20. | $\text{Rel} = \text{"Equal"}$ | |
| 21. | Else if $\prod_{V_2}(w_p) = 1$ then | |
| 22. | $\text{Rel} = \text{"Included in"}$ | |
| 23. | Else if $\prod_{V_1}(w_p) = 1$ then | |
| 24. | $\text{Rel} = \text{"Includes"}$ | |
| 21. | Else if $\text{semRel} \leq T_{\text{Disjoint}}$ then | |
| 22. | $\text{Rel} = \text{"Disjoint"}$ | |
| 27. | Else if $T_{\text{Disjoint}} < \text{semRel} < T_{\text{Equality}}$ then | |
| 28. | $\text{Rel} = \text{"Intersect"}$ | |
| 29. | End if | |
| 30. | Return $\langle \text{SemRel}, \text{Rel} \rangle$ | |
| 31. | End | |

Example 5. Considering Example 2, (T_1 of CNN2 and T_2 of BBC2), and thresholds $T_{Disjointness}=0.1$ and $T_{Equality}=0.9$, $SemRel(T_1, T_2) = 0.86$ and $Relation(T_1, T_2) = Intersection$ as no concept of T_1 is included in the antonymy neighborhood of concept from T_2 or vice-versa. Hence, $TR(T_1, T_2) = \langle 0.86, Intersection \rangle$.

5.2 RSS Item Relatedness

Computing the Item relatedness (IR) is related to the relatedness between its sub-elements. The relatedness between two simple elements (c.f. def. 3) (ER) is computed using Algorithm 2. It accepts two elements e_1 and e_2 as input (line 1) and returns a tuple quantifying $SemRel$ and $Relation$ values between e_1 and e_2 based on corresponding label and value relatedness.

In lines 7 – 8, label and content relatedness are computed respectively using the TR algorithm. In line 9, method E_{SemRel} quantifies the relatedness value between elements, as the *weighted sum* of label (LB_{SemRel}) and value (VR_{SemRel}) semantic relatedness values, such as:

$$\begin{aligned} SemRel(e_1, e_2) &= E_{SemRel}(LB_{SemRel}, VR_{SemRel}) \\ &= w_{Label} \times LB_{SemRel} + w_{Value} \times VR_{SemRel} \end{aligned} \quad (10)$$

where $w_{Label} + w_{Value} = 1$ and $(w_{Label}, w_{Value}) \geq 0$. Note that several methods for combining label and value relatedness results could have been used, among which the maximum, minimum, average and weighted sum functions. Nonetheless, the latter provides flexibility in performing the match operation, adapting the process w.r.t. the user's perception of element relatedness.

In line 10, the rule-based method $E_{Relation}$ is used for combining label and content relationships as follows:

$$\begin{aligned} Relation(e_1, e_2) &= E_{Relation}(LB_{Relation}, VR_{Relation}) \\ &= \begin{cases} disjointness, i.e., e_1 \not\supset e_2 & \Rightarrow e_1.\zeta \not\supset e_2.\zeta \vee e_1.\eta \not\supset e_2.\eta \\ Intersection, i.e., e_1 \cap e_2 & \Rightarrow e_1.\zeta \cap e_2.\zeta \vee e_1.\eta \cap e_2.\eta \\ Equality, i.e., e_1 = e_2 & \Rightarrow e_1.\zeta = e_2.\zeta \wedge e_1.\eta = e_2.\eta \\ Inclusion, i.e., e_1 \supset e_2 & \Rightarrow ((e_1.\zeta \supset e_2.\zeta \vee e_1.\zeta = e_2.\zeta) \\ & \wedge e_1.\eta \supset e_2.\eta) \vee (e_1.\zeta \supset e_2.\zeta \wedge e_1.\eta = e_2.\eta) \end{cases} \end{aligned} \quad (11)$$

| Algorithm 2: ER Algorithm | | |
|---------------------------|---|---|
| 1. | Input: e_1, e_2 : element | // two simple elements |
| 2. | Variable: LB_{SemRel}, VR_{SemRel} : Double | // label and value semantic relatedness values |
| 3. | $LB_{Relation}, TR_{Relation}$: String | // Label and value relationship values |
| 4. | Output: $SemRel$: Double | // relatedness value between e_1 and e_2 |
| 5. | $Relation$: String | // relationship value between e_1 and e_2 |
| 6. | Begin | |
| 7. | $\langle LB_{SemRel}, LB_{Relation} \rangle = TR(e_1.\eta, e_2.\eta)$ | // relatedness between labels |
| 8. | $\langle VR_{SemRel}, VR_{Relation} \rangle = TR(e_1.\zeta, e_2.\zeta)$ | // relatedness between values |
| 9. | $SemRel = E_{SemRel}(LB_{SemRel}, VR_{SemRel})$ | // E_{SemRel} – combines the label and value relatedness values |
| 10. | $Relation = E_{Relation}(LB_{Relation}, VR_{Relation})$ | // $E_{Relation}$ – combines the label and value relationships values |
| 11. | Return $\langle SemRel, Relation \rangle$ | |
| 12. | End | |

Having identified the semantic relatedness and relationships between simple elements, Algorithm 3 evaluates RSS item relatedness IR.

Given two RSS items I_1 and I_2 , each made of collection of sub-elements $\{e_i\}$ and $\{e_j\}$ respectively, the *Item Relatedness (IR)* algorithm returns a tuple quantifying $SemRel$ as well as the $Relation$ between I_1 and I_2 based on corresponding element relatedness (lines 10 – 16).

Line 12 computes the relatedness between simple elements e_i and e_j and returns semantic relatedness value ej_{SemRel} , and relationship $ej_{Relation}$. In line 13, the semantic relatedness value ej_{SemRel} is accumulated in order to get grand total, and, in line 14, $ej_{Relation}$ is stored for later use. In line 17, the semantic relatedness value between I_1 and I_2 is computed as the average of the *semrel* values between corresponding element sets of I_1 and I_2 .

$$SemRel(I_1, I_2) = \frac{\sum_{e_i \in I_1} \sum_{e_j \in I_2} SemRel(e_i, e_j)}{|I_1| \times |I_2|} \quad (12)$$

As for the relationships between two items, we develop a rule-based method $I_{Relation}$ (line 18) for combining sub-element relationships stored in $Eij_{Relation_set}$ (which is the relationship between e_i and e_j) as defined below:

- I_1 and I_2 are *disjoint*, denoted $I_1 \triangleright \triangleleft I_2$ if all elements $\{e_i\}$ and $\{e_j\}$ are disjoint (elements are disjoint i.e. there is no relatedness whatsoever between them. Formally, $Relation(I_1, I_2) = Disjoint \Rightarrow \forall e_1 \in \{e_i\}, \forall e_2 \in \{e_j\}, e_1 \triangleright \triangleleft e_2$
- I_1 *includes* I_2 , denoted as $I_1 \supset I_2$ if all elements in $\{e_i\}$ include all those in $\{e_j\}$. Formally, $Relation(I_1, I_2) = Includes \Rightarrow i \geq j \wedge \forall e_2 \in \{e_j\}, \exists e_1 \in \{e_i\} \mid e_1 \supset e_2$
- I_1 and I_2 *intersect*, denoted as $I_1 \cap I_2$ if at least two of their elements intersect. Formally, $Relation(I_1, I_2) = Intersects \Rightarrow \exists e_1 \in \{e_i\}, \exists e_2 \in \{e_j\} \mid e_1 \cap e_2$
- I_1 and I_2 are *equal*, denoted as $I_1 = I_2$ if all their elements in $\{e_i\}$ equal to all those in $\{e_j\}$. formally, $Relation(I_1, I_2) = Equal \Rightarrow i = j \wedge \forall e_2 \in \{e_j\}, \exists e_1 \in \{e_i\} \mid e_1 = e_2$

| Algorithm 3: IR Algorithm | | |
|---------------------------|---|---|
| 1. | Input: I_1, I_2 : element | // two input items (Complex elements) |
| 2. | Variable: eij_{SemRel} : Double | // semantic relatedness values e_i and e_j |
| 3. | $eij_{Relation}$: String | // relationship value between e_i and e_j |
| 4. | $Eij_{Relation_set}$: Set | // would contain sub-elements relationship values |
| 5. | Output: $SemRel$: Double | // relatedness value between I_1 and I_2 |
| 6. | $Relation$: String | // relationship value between I_1 and I_2 |
| 7. | Begin | |
| 8. | $SumRel = 0$ | |
| 9. | $Eij_{Relation_set} = \emptyset$ | |
| 10. | For each e_i In I_1 | |
| 11. | For each e_j In I_2 | |
| 12. | $\langle eij_{SemRel}, eij_{Relation} \rangle = ER(e_i, e_j)$ | |
| 13. | $SumRel = SumRel + eij_{SemRel}$ | |
| 14. | $Eij_{Relation_set} = Eij_{Relation_set} \cup eij_{Relation}$ | |
| 15. | Next | |
| 16. | Next | |
| 17. | $SemRel = SumRel / I_1 \times I_2 $ | |
| 18. | $Relation = I_{Relation}(Eij_{Relation_set})$ | |
| 19. | Return $\langle SemRel, Relation \rangle$ | |
| 20. | End | |

Example 7. Let us consider RSS items *CNN1* and *BBC1* (Figures 1 and 2). Corresponding item relatedness is computed as follows. Weighting factors of $w_{label} = 0.5$ and $w_{value} = 0.5$ are assigned to label and text values, while evaluating simple element relatedness. Thresholds $T_{Disjointness} = 0.1$ and $T_{Equality} = 0.9$ are used in getting the relationship value. Simple element relatedness values and relationships are given below in Table 1.

Table 1: Element Relatedness matrix

| ER | $title_{BBC1}$ | $description_{BBC1}$ |
|----------------------|-------------------|-----------------------|
| $title_{CNN1}$ | <0.908, Equal> | <0.655, Intersection> |
| $description_{CNN1}$ | <0.650, Includes> | <0.832, Intersection> |

Using (cf. 12) $SemRel(CNN1, BBC1) = (0.908 + 0.655 + 0.650 + 0.832) / 2 \times 2 = 0.761$, where $|I_1|$ and $|I_2|$ are equal to 2.

$Relation(CNN1, BBC1) = Intersection$ since a number of their elements intersect, i.e., $Relation(description_{CNN1}, description_{BBC1}) = Intersection$.

Measuring the relatedness between items is important in several applications such as clustering, classification, merging, etc. Item relatedness is important to determine what to merge and how to merge. Obviously, we should merge those items closely related (i.e., those with maximum semantic related value and related with inclusion/equality relationship). This involves the adaption of classical clustering algorithm to be relationship aware as detailed in Section 6. In addition, how to merge those within the same cluster involves merging rules that utilizes relationship, detailed in Section 7.

6 CLUSTERING

Clustering is a method for grouping similar data together. In our merging frame-work, clustering is a pre-processing step that facilitates the merging process. The different clustering methods are divided into two broad categories: Hierarchical and Non-hierarchical. Hierarchical (agglomerative and divisive) [10][25] clustering algorithms produces nested sets of data (hierarchies), in which pairs of elements or clusters are successively linked until every element in the data set becomes connected. Single link [57], complete link [56] and group average link [55] are known as hierarchical clustering methods. Non-hierarchical methods group a data set into a number of clusters irrespective of the route by which they are obtained (e.g. k-means [16] groups the data set into k different clusters based on similarity).

The clustering module stated in our merging framework (cf. Section 4), may use any of the known clustering algorithms. Disregarding the relationships between RSS news items would produce clusters that only encompass members with high relatedness (e.g., having lots of intersections). Yet, such clusters would be missing those elements with lower relatedness, but which are connected via the inclusion relationship. Such results obviously affect merging quality since news/documents connected via the inclusion relationship, for instance, should naturally belong to the same cluster, so as to be subsequently merged together.

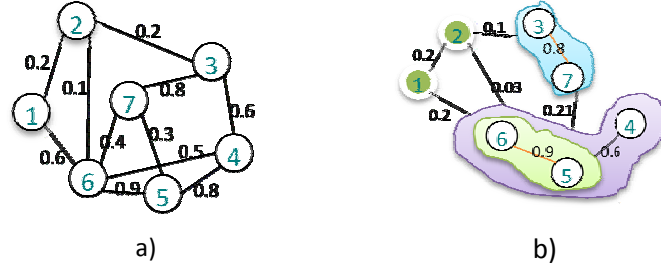


Fig. 9. Single link clustering at level 0.6

In this work, we have adapted the graph based agglomerative single-link clustering method [25] to consider RSS item relationships. Given n RSS news items, we form a fully connected graph G with n vertices and $n(n-1)/2$ weighted edges. The vertices represent the news items/clusters and the weight of an edge corresponds to item semantic relatedness/similarity value between vertices that this edge connects. The single link clusters for a clustering level l_i (i.e. c_{l_i}) can be identified by combining those vertices with weights $w \geq l_i$ from the graph G . The clustering level is a threshold value used to combine clusters including single news clusters. Figure 9a represents a graph with seven nodes that corresponds to single news clusters (the numbers in the circle represents the id of member news items). The weight corresponds to the semantic similarity between the news items. The missing edges have a semantic similarity of 0. Figure 9b presents the graph remaining after deleting all edges with weight < 0.6 (i.e. combining those vertices with weight ≥ 0.6). There are four clusters $C_1=\{1\}$, $C_2=\{2\}$, $C_3=\{3, 7\}$, $C_4=\{6, 5, 4\}$ representing clustering at level 0.6. In general, the resulting of cluster at level l_i c_{l_i} contains all news items I with semantic relatedness value greater than or equal to l_i . Formally:

$$(13)$$

Where: $SemRel()$ function returns the semantic relatedness between two items.

Our Relationship aware Single Link Level based or **RaSL²** clustering algorithm is shown in Algorithm 4. Nodes in graph G represent clusters encompassing sets of news items with maximum semantic relatedness and equality/or inclusion relationships. Edges represent average relatedness

similarity values between pair of clusters. The semantic relatedness between clusters C_i and C_j is computed as follows using Unweighted Pair Grouping Method (UPGM)[57]:

$$AvgSemRel(C_i, C_j) = \frac{\sum_{k=1}^{|C_i|} \sum_{l=1}^{|C_j|} SemRel(I_k^{C_i}, I_l^{C_j})}{|C_i| \times |C_j|} \quad (14)$$

Where: $I_k^{C_i}, I_l^{C_j}$ and represent the k^{th} and l^{th} member news item of clusters C_i and C_j respectively, and $|C_i|, |C_j|$ represents the size of cluster C_i and C_j respectively, $SemRel$ returns semantic relatedness value between the two items.

For instance, in Figure 9b. The weight of edge connecting cluster $C_2=\{2\}$, and $C_3=\{3, 7\}$ is computed as

$$AvgSemRel(C_3, C_4) = \frac{SemRel(2, 7) + SemRel(2, 3)}{1 \times 2} = \frac{0 + 0.2}{1 \times 2} = 0.1$$

The same way the edge connecting all the clusters is computed.

The result of **RaSL**² clustering C_{li} is similar to the α cut clustering result of Ian Gracia *et al.* [24]. In [24], an article may belong to different clusters, and a cluster contains a set of related articles. The redundant (identical and subsume) and less-informative articles are removed with the help of a fuzzy equivalence relation. However, our algorithm generates independent clusters (i.e. pair of news from two different clusters are related only with *disjointness* relationship).

The algorithm **RaSL**² generates clusters by varying the clustering level between 1 and 0, at a constant pace *Dec-value*. Lines 7 and 8 show clustering at level 1, generating the initial clusters which group individual news items and those related with equality and/or inclusion relationships. Lines 10 to 14 show clustering on level l_i after computing relatedness value between clusters using UPGM. Clusters are grouped only if the corresponding weight is greater than or equal to l_i .

| Algorithm 4: RaSL² Algorithm | |
|--|---|
| 1. | Input: Sem_Rel [][] // relatedness matrix |
| 2. | Output: Clusters: Collection // contain the result of clustering |
| 3. | Variable: Dec-value: double // constant clustering level decrement value (eg. -0.1) |
| 4. | l_i : double // clustering level |
| 5. | Begin |
| 6. | For $l_i = 1$ to 0 step <i>Dec-value</i> |
| 7. | If $l_i = 1$ then |
| 8. | group all clusters at relatedness/similarity value of 1 or those related with equality/inclusion relationship |
| 9. | Else |
| 10. | For each cluster c_i and c_j in clusters _{l_{i-1}} |
| 11. | Average-Relatedness = UPGM(c_i, c_j) // computed using formula 17 |
| 12. | If Average-Relatedness $\geq l_i$ then |
| 13. | group c_i and c_j in same cluster |
| 14. | Next |
| 15. | End if |
| 16. | Next |
| 17. | C-Index(Clusters) // stopping rule for clustering |
| 18. | Return clusters |
| 19. | End |

A stopping rule is necessary to determine the most appropriate clustering level for the single link hierarchies. Milligan *et al.* present 30 of such rules [53]. Among these rules, C-index [54] exhibits excellent performance (found in the top 3 stopping rules). Here, Line 17, we used an adaptation of C-index, provided by Dalamagas *et al.* [7].

7 MERGING

In our approach, merging a list of news items collected from the same or different RSS providers is controlled by several merging rules defined as a set of expression according to the human needs on the basis of relationships existing between items. We assume that RSS news items are of known size or cardinality. In the following, we consider that RSS news items are extracted and stored in a container Φ and each item is accessed via index i as $\Phi[i]$. Path expression is used to access elements an item.

In this section, we explain first the merging operators, merging rule and action that would be followed in order to merge two RSS news items. Later, we extend the approach to merge set of news items. In our approach News item can be modeled with merging object hierarchy shown in Figure 10 below. The rectangle represent object (solid border) or the property of the object (dotted border). The relationship between objects is hierarchical and it is HAS A.

For example, an object of type *Element* has property *Name* and *Content* and has complex object named *Attributes*. *Attributes* is collection of *attribute* each having *Name* and *Value*.

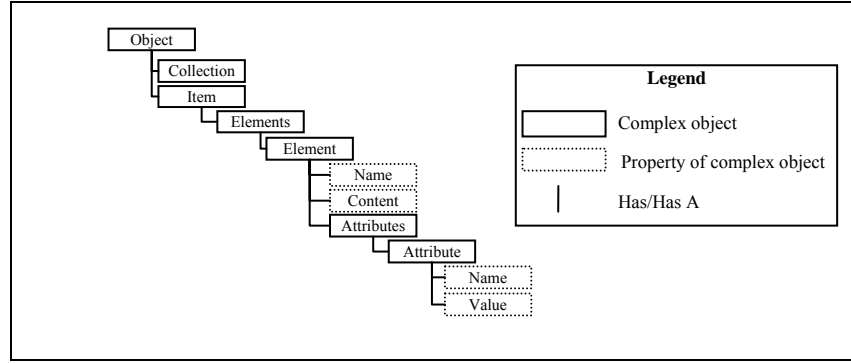


Fig. 10. Merging Object Hierarchy

7.1 Basic merging operators

In this section, we provide the list of basic merging operators that would be used in our merging expression. The merging operators are functions having name, accept two or more parameters and returns result to the caller. The parameters are restricted to refer to item, element or text values. The following is the list of basic operators:

1. *String getRelation(Object X, Object Y)* : returns the relationship existing between object X and Y.

For instance, *getRelation(CNN1/title, BBC1/title)*- returns Equal (c.f. Table 2)

2. *Boolean IsXXXX(Object X, Object Y)*: returns true if the objects X and Y are related with the relation XXXXX, where XXXXX in {Equal, Intersect, Disjoint, Include}

For instance, - *IsEqual(CNN1/title, BBC1/title)* - returns true (c.f. Table 2)

3. *Element CreateElement(String TagName, String value)*: return a new element named *TagName* having *value* as content.
4. *Element CreateElement(Element X, Element Y, String TagName)*: return a new element named *TagName* and having X and Y as children. Object refers to only simple or complex elements.

For instance,

- *CreateElement (CNN1/title, BBC1/title, 'NewTitle')* - returns an element named *NewTitle* having title elements of *CNN1* and *BBC1*.
- *CreateElement (Φ[1], Φ[5], "m")* returns an element named m having the first and the fifth news items as children.

5. *String Concat(String X, String Y, String Z)*: return the result of concatenating two strings X and Y separated by another string Z.

For instance, *Concat("U.N. chief launches \$600M Gaza aid appeal", "UN launches \$613m appeal for Gaza", "|")* returns

U.N. chief launches \$600M Gaza aid appeal | UN launches \$613m appeal for Gaza

6. *Object keepFirst(Object X, Object Y)* – returns X
7. *Object keepSecond(Object X, Object Y)* – returns Y
8. *Object keepBoth(Object X, Object Y)*: return the concatenation of both objects separated by space.
9. *Item getLatest(Item X, Item Y)*: returns the latest/recent news item.

For instance, *getLatest(CNN4, BBC4)* – returns *BBC4* as it is published on ‘Fri, 02 May 2008 04:08:38 GMT’ and *CNN4* on ‘Thu, 01 May 2008 21:51:15 EDT’

10. *String BuildText (string [] CS)* – returns a human understandable text/sentence containing all the concepts in concept set CS and there is a need to add article, proposition, etc.¹⁸

For instance, given the concept set $CS = \{\{UN\}, \{chief\}, \{launch\}, \{Gaza\}, \{aid\}, \{appeal\}\}$, *getBuildText(CS)* – returns “UN chief launch gaza aid appear”

11. *String[] getCommon(string X, string Y)*: returns set containing concepts shared by text X and Y.

For instance, *getCommon(CNN1/Title/text(), BBC1/Title/text())* returns $\{UN\}, \{launch\}, \{Gaza\}, \{appeal\}$

12. *String[] getDifferent(string X, string Y)* – returns concepts existing only in X or Y but not both.

For instance, *getDifference(CNN1/Title.ζ, BBC1/Title.ζ)* – returns the concept {aid} which exists only in *CNN1*.

13. *String LCA(string C₁, string C₂)*: returns a lowest common ancestor of *C₁* and *C₂* using the label knowledge base LKB (cf. LKB in Figure 6.b). We exploit this operator to decide on the tag name of elements to be merged.

For instance: *LCA(CNN1/Title/name(), BBC1/Link/name())* – returns the *Description* as LCA of the name of Title element of *CNN1* and Link element of *BBC1*.

14. *String[][] GetCorrespondence(item I₁, item I₂, collection[][] Elts_Rel)*: Given two items *I₁*, *I₂* and matrix containing sub-element relatedness between members of each item, the operator returns a matrix/list containing matching elements and relationship in between. Notice that each member of the output has three members *e_i*, *e_j* and *relation* (if *e* refers to member of collection, *e.e_i*, *e.e_j* and *e.relation* would access the sub-elements of *I₁*, and *I₂* respectively and the associate relationship value). Correspondence between members of content descriptor (i.e. labels used to compute relatedness) is computed on the bases of maximum element relatedness; for the rest label similarity would be used having null value for the relationship component of the matrix. In addition, the operator assigns null to those elements without matching.

For example, *GetCorrespondence(CNN1, BBC1, Element-Relatedness-Matrix¹⁹(CNN1, BBC1))*, returns the correspondence between sub-element of *CNN1* and *BBC1*. The result is shown in Table 1. The matching between Title, Description elements is done using maximum relatedness where as for link, guid and pubdate tagname similarity has been used. There is no matching element for *BBC1/category* element.

15. *String[][] GetCorrespondence(item I₁, item I₂)* – returns the correspondence between input items *I₁*, item *I₂*. The operator calls *GetCorrespondence* with third parameter being *Element-Relatedness-Matrix(I₁, I₂)*.

16. *element Handle-Element-Conflict(Element e₁, element e₂)*: returns an existing element only. This operator helps to handle the case in which an element exists only in one news items.

17. *Collection Handle-Attributes(element e₁, element e₂)* return attributes after consider the following facts

- if only one of the elements have attribute, add return it.
- if both elements have identical or semantically equivalent attribute name and value then keep one of the attribute.
- if the elements have identical or semantically equivalent attribute name but different values then add an attribute having an Or-valued list.
- else add return both.

¹⁸ Developing human readable sentence/phrase containing all concepts in given concept sets is outside the scope of this work.

¹⁹ It returns ER value between sub-elements of each items. The result is similar to Table 1.

In addition, we have defined the two derived operators that would be used to merge intersecting simple elements and items:

18. *element Intersecting_{Elements}(element e_i , element e_j)*- returns an element name after the LCA of name of each element and content build using the common and different concept the respective text. i.e.

```
String tagname = LCA( $e_i.\eta$ ,  $e_j.\eta$ )
String content = Concat(BuildText(getCommon( $e_i.\zeta$ ,  $e_j.\zeta$ )),
                        BuildText(getDifferent( $e_i.\zeta$ ,  $e_j.\zeta$ )), "|")
Return CreateElement (tagName, content)
```

19. *Item Intersecting_{Items}(Item I_1 , Item I_2)*- returns the result of merging corresponding sub-elements of I_1 and I_2 i.e.

```
collection Corr_Matrix=getCorrespondence( $I_1, I_2, Element-RelatedNess-Matrix(I_1, I_2)$ ) // get elements correspondence
 $I_k$  = CreateElement("Item", Null) // create empty element named Item
element  $e_k$ 
for each  $e$  In Corr_Matrix
     $e_k$  = MergeElements ( $e.e_i$ ,  $e.e_j$ ,  $e.relation$ ) // merge elements
     $I_k.Elements.Add(e_k)$  //add  $e_k$  to the elements collection of Item (c.f. Fig. 10)
next
return  $I_k$ 
```

7.2 Merging rules

Our merging process depends on the mapping of relationship existing between elements and operation that would be executed which is represented as, $f: \mathcal{R} \rightarrow \beta$.

Where :

- \mathcal{R} represents the set of possible relationship values existing between objects
 $\mathcal{R} = \{Equal, Includes, Intersects, Disjoint\}$ [58]
- β represents the merging expression – combination of one or more merging operators

If the antecedent expression \mathcal{R} is true, then the expression in β would be evaluated. The antecedent in \mathcal{R} is restricted to expression that checks the relationship between items, elements or text contents, and is formally denoted as:

$$\begin{aligned} getRelation(v_1, v_2) = c_1 [\vee getRelation(v_1, v_2) = c_2 \dots] \\ \Rightarrow \beta \end{aligned} \quad (15)$$

where:

- $getRelation(V_1, V_2)$ - returns the relationship value
- V_1 and V_2 are path expressions returning item, element or text content)
- c_1 and c_2 are relationship values

Merging two items using our merging rule is similar to the propositional fusion rule of Hunter [20]. However, our merging process is dependent on the relationship existing between elements or items and presented as follows.

1. *element Merge_{Elements}(element e_i , element e_j , string rel)*: It accepts two simple elements, the relationship value and returns merged version i.e.

```
element  $e_k$ 
if (rel = 'Equal' OR rel = "Include")
     $e_k$  = KeepFirst( $e_i, e_j$ )
else if(rel = "Disjoint")
     $e_k$  = CreateElement( $e_i, e_j$ , "m")
else if(rel = "Intersects")
     $e_k$  = IntersectingElements( $e_i, e_j$ )
```

```

ek.Attributes.add(Handle-Attributes(ei, ei))
return ek

```

The operator returns either (a) containing element in the case of equality or inclusion relation, (b) a new element named m (by default) and having both element as children in the case of disjoint relation as done in the work of Ho-Lam et al [33], or (c) a new element having the lowest common ancestor of both labels as label and the content showing common and distinct concepts of each element's content in the case of intersection relation result of *IntersectingElements*

For instance, - *MergeElements*(CNN1/description, BBC1/description, "Intersect") returns an element having description as label and text contents separated by | as content. i.e.,

<description> United Nations Secretary-General Ban Ki-moon launch appeal aid people Gaza Israel military offensive | \$613m affected offensive, body's top official says provide emergency humanitarian aftermath </description>.

2. *Item MergeItems* (Item I₁, Item I₂) – returns the result of merging two items after getting the relationship

```

string rel = getRelation(I1, I2)
if (rel = 'Equal')
    return getLatest(I1, I2)
else if (rel = "Include")
    return KeepFirst(I1, I2)
else if (rel = "Disjoint") // keep both items
    Return KeepBoth(I1, I2)
else if (rel = "Intersects")
    return IntersectingItems(I1, I2)

```

The operator returns, either (a) the latest news in the case of equal news, (b) containing news in the case of inclusion relation (c) keeps both items in the case of disjoint news (d) the result of merging correspondence sub-elements of the items in the case of intersection relation.

Example 8: Let us consider RSS items CNN1 and BBC1 in Example 1. The item relatedness value between CNN1, BBC1: IR($\Phi[1]$, $\Phi[5]$) = IR(CNN1, BBC1) = {0.726, Intersects}

Hence, merging these items comes down to the merging of corresponding sub-elements. The correspondence between sub-elements (i.e., the result of *getCorrespondence* operator) is shown in Table 2.

MergeItems($\Phi[1]$, $\Phi[5]$) = <Item> *MergeElements* (title_{CNN1}, title_{BBC1}, "Equal") *MergeElements*(description_{CNN1}, description_{BBC1}, "Intersects") *MergeElements* (Link_{CNN1}, Link_{BBC1}, "Disjoint") *mergeElements*(guid_{CNN1}, guid_{BBC1}, "Disjoint") *MergeElements* (null, category_{BBC1}, null) </item>

The result is shown in Figure 4.

Table 2: Correspondence between Elements of CNN1 and BBC1.
getCorrespondence(CNN1,BBC1)

| e _i | e _j | Relation |
|-----------------------------|-----------------------------|-----------|
| title _{CNN1} | title _{BBC1} | Equal |
| description _{CNN1} | description _{BBC1} | Intersect |
| link _{CNN1} | link _{BBC1} | Disjoint |
| guid _{CNN1} | guid _{BBC1} | Disjoint |
| Null | Category _{BBC1} | Null |

7.3 Actions

According to Hunter [20], an action determines the order and pattern in which an expression would be executed. It include each of the following expressions

1. *Document Initialize*(String OutType) – it creates and returns an empty document of OutType which could be RSS, XML or XHTML.

2. *Void AddElement(Element nw, Element Parent)* – It adds the element *nw* as child of *Parent*.

Notice that, in building valid document, *Initialize* action should be executed before *AddElement*. In addition, there would be only one initialize action.

Example 9: Considering Example 8 above, the following action list generates an RSS document having the merged items.

```
Document Doc = Initialize("RSS") // Create a document of given type OutType – default RSS
AddElement(MergeItem(CNN1, BBC1), Doc)

<?xml version="1.0" encoding="ISO-8859-1" ?>
<RSS version="2.0" >
  <Channel>
    <item>
      <title>U.N. chief launches $600M Gaza aid appeal</title>
      <description> United Nations Secretary-General Ban Ki-moon launch appeal aid people Gaza Israel military offensive | $613m affected offensive, body's top official says provide emergency humanitarian aftermath </description>
      <m>
        <link>http://edition.cnn.com/2008/WORLD/americas/05/01/gitmo.journalist/index.html?eref=edition</link>
        <link>http://edition.cnn.com/2008/WORLD/americas/05/01/gitmo.journalist/index.html?eref=edition</link>
      </m>
      <m>
        <guid>http://edition.cnn.com/2008/WORLD/americas/05/01/gitmo.journalist/index.html?eref=edition</guid>
        <guid isPermaLink="false">http://news.bbc.co.uk/go/rss/-/2/hi/me/723378828.stm </guid>
      </m>
      <category>Middle-east</category>
    </item>
  </Channel>
</RSS>
```

Fig. 11. Result of Merging two news items

7.4 Human Assisted merging operator

In our merging framework (c.f. Section 3), users are empowered to control and dictate the merging process. As a result, every user is allowed to specify his/her notion of merging by associating relationships between elements and merging operators. The merging operators *MergeElements* and *MergeItems* represent default merging rules. In this work, users provide merging rules/options by associating a template containing a list of identified relationships between elements and a template containing a list of merging operators as shown Figure 12. Algorithm 5 represents mapping between relations and action. The associations between relation and merging operator are stored as user-defined merging rules. Each element of User-merging-rule has relation and action component which would control the merging of elements.

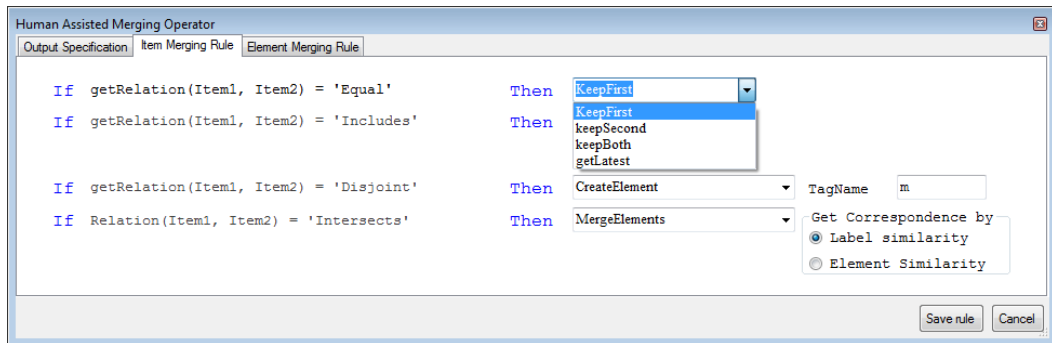


Fig. 12. Screenshot of user based merging template

| | |
|--|---|
| Algorithm 5: Human Assisted Merging operators | |
| 1. | Input : $A : \{\text{Concat, KeepFirst, KeepSecond, KeepBoth, Intersecting}_{\text{Elements}}, \text{Intersecting}_{\text{Items}}, \text{CreateElement}\}$ // Set of Merging Operators |
| 2. | $R : \{\text{Equal, Includes, Intersect, Disjoint}\}$ // Set of Relations |
| 3. | Variable: $\text{Element-Type} : \text{Simple} \mid \text{Item}$ // type of element |
| 4. | $\text{User-merging-option} : \text{Merging-rule}$ //merging rule provided by users |
| 5. | $\text{Mapped-rule} : \text{Merging-rule}$ // mapped merging rule |
| 6. | $\text{User-merging-rule} : \{\text{Merging-rule}\}$ // set of user provided and mapped merging rule |
| 7. | Output: <i>None</i> |
| 8. | Begin |
| 9. | $\text{User-merging-rule} = \emptyset$ |
| 10. | For each relation r in R |
| 11. | $\text{User-merging-option} = f(r, \text{Element-Type}, A)$ |
| 12. | $\text{User-merging-rule} = \text{User-merging-rule} \cup \text{User-merging-option}$ |
| 13. | Next |
| 14. | Store User-merging-rule in user profile |
| 15. | End |

In line 11, function f associates relation r (between elements) having *element-type*: Simple or Item to one of the predefined operators $a \in A$. The *User-merging-option* is accumulated (line 12) and finally stored as profile.

Example 10: A user overrides the default merging rule for equality, and disjointness news items with help of template (user interface) shown in Figure xxx:

- If $\text{Relation}(I_1, I_2) = \text{'equal'}$ then $\text{KeepBoth}(I_1, I_2)$ // both news news
- If $\text{Relation}(I_1, I_2) = \text{'Disjoint'}$ then $\text{Concat}(I_1, I_2, "|")$

7.5 Merging RSS news items

Merging RSS news items collected from one or more sources can be done after grouping items using our relationship-aware clustering algorithm. Recall that merging could be done without performing clustering. Nonetheless, clustering would provide more relevant merging candidates, and thus would amend merging results (c.f. Section 6).

Hereunder, we start by defining an item neighborhood to be exploited in applying the default merging rules, and performing RSS news items merging.

Definition 12. [Item neighborhood]

The neighborhood of news item I_i refers to a set of news items I_j related with relation *equality* or *inclusion*. Formally, it is denoted as:

$$N(I_i) = \{I_j \mid I_i = I_j \vee I_i \supset I_j\} \quad (16)$$

Neighborhood of an item I_i returns all news items redundant to it or contained in it. As a result all items in $N(I_i)$ can be collapsed and represented by I_i without losing information.

For example,

$N(\text{CNN1}) = \{I_j \mid \text{IsEqual}(\text{CNN1}, I_j) \vee \text{IsIncludes}(\text{CNN1}, I_j)\}$ returns all news related with equality or inclusion with CNN1. Notice that CNN1 would be the representative of the resulting set.

7.5.1 Merging Algorithm

Algorithm 6 handles merging of news items. The algorithm accepts a cluster of news items and the corresponding semantic relatedness matrix i.e. *sem_rel*. For any pair of news items i and j , $\text{sem_rel}[i][j].\text{value}$ and $\text{sem_rel}[i][j].\text{rel}$ represent respectively the relatedness and relationship components of the item relatedness measure. In addition, the algorithm accepts *User-merging-rule* extracted from the user profile.

In line 7, and empty RSS document is created with the using the initialize action. Then, in line 9, the item neighborhood of a news item is identified, so as to produce special item I_r , that can represent the merged result of all news belonging to same neighborhood using *Merge-Items-*

*Neighborhood*²⁰. Then in line 8, the semantic relatedness matrix is updated by deleting the rows and columns of items included in the global neighborhood of I_i and add I_r into the RSS file. Lines 15 – 22 are used to merge all the remaining news items. The merging process is conducted incrementally.

Algorithm 6: Merging RSS news Items

```

1.  Input:  $C_i: \{I_1, I_2, \dots, I_m\}$  //  $C_i$  is a cluster having  $I_k$  items  $1 \leq k \leq m$ 
2.       $sem\_rel$  [][] // it contains items relatedness value
3.       $User\_merging\_rule : Collection$  // list of action to be done based on the relationship
4.  Variable:  $r, s : integer$ 
5.  OutPut: Doc: Document // file containing merged news items
6.  Begin
7.      Doc = Initialize ("RSS")
8.      For each  $I_i$  in  $C_i$ 
9.          Find the neighborhood of  $I_i$  and identify the representative //cf. Def. 19
10.         Update  $sem\_rel$  matrix by deleting news included in neighborhood of  $I_r$ 
11.         Let  $I_r = Merge\_Items\_Neighborhood(I_i, user\_merging\_rule)$ 
12.         AddElement ( $I_r$ , Doc)
13.     Next
14.     Do
15.          $(r,s) = \max(sem\_rel[i][j].value)$  // Find the most similar pair of news items say  $r$  and  $s$  over all items
16.          $I_k = merge(I_r, I_s, user\_merging\_rule)$  //Merge  $r$  and  $s$  to form new item  $I_k$ .
17.         AddElement ( $I_k$ , Doc)
18.         Update  $sem\_rel$  matrix
19.          $SemRel(E_{ik}, E_{(s,r)}) = Avg(SemRel(E_{ik}, E_{sj}), SemRel(E_{ik}, E_{rj}))$ 
20.          $Relation(E_{ik}, E_{(s,r)}) = Fuzzy(SemRel(E_{ik}, E_{(s,r)}), Threshold_{disjoint}, Threshold_{Equal})$ 
21.          $sem\_rel[i][j](r,s) = IR(I, (r,s))$ 
22.     Until all items are merged
23.     Return RSS
24. End

```

In line 15, any two highly related news items (I_s and I_r) over all pair of items are identified. These news items are merged using the merging rule provided by the user (line 16). The resulting news item is added to output RSS file (line 17). In line 18, the sem_rel matrix is updated by removing rows and columns of I_s and I_r and adding the newly merged news I_k . Item relatedness between I_k and those related with I_s and I_r is estimated after estimating the relatedness between sub-elements of I_s and I_r . The semantic relatedness between sub-elements E_{ik} of I_i and I_k is computed as the average $SemRel(E_{ik}, E_{sj})$, and $SemRel(E_{ik}, E_{rj})$ where E_{sj} and E_{rj} are sub-elements of I_s and I_r respectively. The relation between sub-elements is computed on the bases of fuzzy notion that takes into consideration semantic relatedness value, threshold disjointness and threshold equality – line 20. In line 21, The semantic relatedness and relationship between items is computed by combining the semantic similarity and relationship values using the Item relatedness algorithm (c.f. item relatedness).

8 EXPERIMENTS

To validate our approach, we have implemented a C# desktop prototype entitled *RSS Merger* encompassing:

- A KB component: stores reference text value and label knowledge bases, VKB and LKB, in a *MySql* DBMS. The value knowledge base VKB and LKB are modified based on the considered application.
- RSS Input component: allows users to register existing RSS news addresses, and accepts parameters to be used in generating synthetic news.
- Containers for generated and/or extracted news
- Container for the user profile – user information and personalized merging rule

²⁰ Merge-Items-Neighborhood merges news items redendent news items based on the equity and inclusion merging rule of the user.

The prototype accepts, as input, RSS news items, as well as Boolean input parameter allowing the user to chose whether to consider data semantics (i.e., exploit VKB and LKB in identifying label/value neighborhoods) or not. It measures relatedness between news items automatically after (i) stemming text values using Porters' algorithm [42], (ii) generating vectors for each text, (iii) computing relatedness and relationships at different level of granularity, i.e., text, label, simple element, and item (complex element). It clusters the RSS items based on the relatedness value and finally merge the news based on the users merging rule.

We have conducted a set of experiments to evaluate (a) the computational complexity and efficiency of our method, (b) the relevance of our relatedness measure, and (c) the relevance of topological relationships in grouping related news items, and consequently performing RSS merging (d) user based relevance of the merging operation. Experiments were carried out on an Intel Core Centrino Duo Processor machine (with 1.73 GHz processing speed and 1GB of RAM).

8.1 Dataset

In conducting the set of experiments, we have used both syntactic and real dataset.

- Syntactic dataset: we have developed a C# prototype that generate RSS document that conforms to RSS 2.0 specification. The prototype accepts each of the following as parameters.
 - Number of news items to be generated
 - Maximum number of concepts per content
 - Number of disjoint number of clusters
 - Number of Equal, including and intersecting news per clusters
- Real dataset: We have used two groups of real datasets

Group 1: It contains 158 RSS news items extracted from well known news providers (CNN, BBC, USAToday, L.A. Times and Reuters). We grouped manually into 6 predefined clusters: US Presidential elections 08, Middle-east, Mumbai-attacks, space-technology, oil, and football. However, we did not identify the relationships that could exist between news items.

Group 2: News collected by Antonio Gulli [59]. We have extracted 567 news published as Top News by CNN, BBC, Newsweek, The Washington Post, Reuters, Guardian, and Time. We group the news into 7 clusters. Table 3 shows the clusters and number of news related with equality, inclusion and intersection relations.

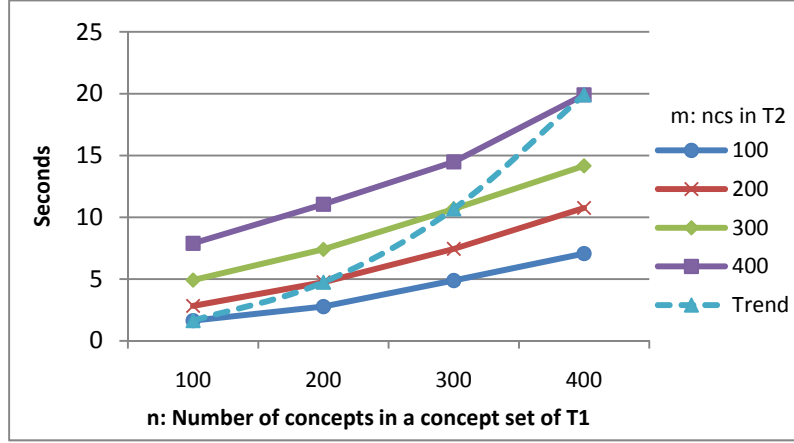
Table 3: Manual Clusters and distribution of relationships

| Cluster Name | # Equality | # Including | # intersection | Total |
|--------------|------------|-------------|----------------|-------|
| Mortgage | 100 | 69 | 0 | 169 |
| Afghan | 17 | 5 | 10 | 32 |
| Bin-Laden | 13 | 4 | 1 | 18 |
| Arafat | 30 | 6 | 19 | 55 |
| Terrorism | 27 | 19 | 78 | 124 |
| USA-Election | 60 | 9 | 100 | 169 |

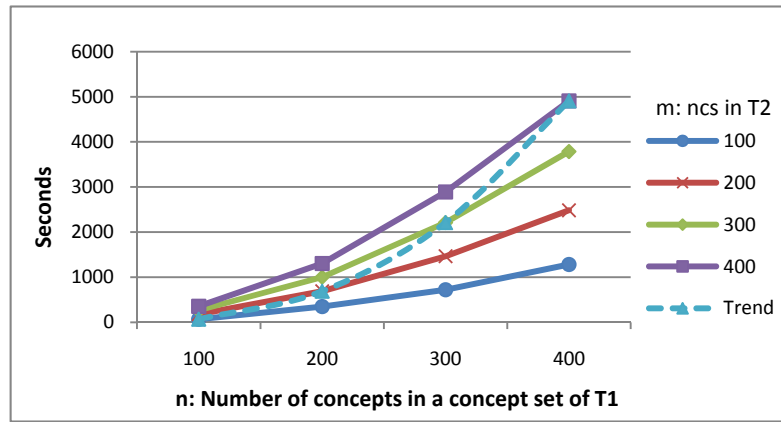
8.2 Timing Analysis and Efficiency

We experimentally tested the time complexity of our relatedness algorithm (R2), w.r.t. the sizes of input texts t_1 and t_2 i.e. number of concept sets (n and m) and value knowledge base information (number of concepts - n_c and depth - d). Note that relationship computation is not included here as its impact is minimal on timing.

On one hand, we can quickly observe the polynomial nature of the timing result shown in Figure 13, demonstrating the polynomial dependency on input text size (13.a) and knowledge base information (13.b). The x axis represents the number of concepts in a concept set and the y axis shows the consumed number of seconds needed to compute the relatedness value.



a. Without semantic knowledgebase



b. With fixed semantic ($d=8, nc = 100$)

Fig. 13. Timing analysis text concept set in t1, t2 (n, m)

In Figure 13, we also show the effect of varying number of concepts in synsets. Figure 13.a shows the timing result without considering knowledge base information while varying the size of the input texts. Increasing the number of concept sets increases the timing in a quadratic fashion (i.e. the dot line shows the growth rate as trend of the algorithm). Figure 13.b represents timing result considering a fixed knowledge base (having 100 concepts with a maximum depth of 8). The time needed to compute the relatedness between items increases drastically (compared to the result shown in Figure 13.a) and in a polynomial fashion.

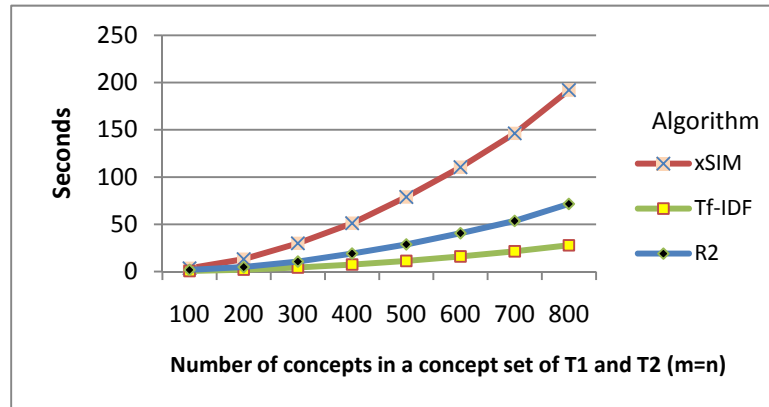


Fig. 14. Timing result obtained using three algorithms: xSim, TF-IDF and our algorithm

On the other hand, we wanted to compare the efficiency of our algorithm with similar existing ones. As alternative algorithms, we chose *xSim* [27] and *TF-IDF* [39], the former being one of the most recent XML-dedicated similarity approaches in the literature, the latter underlining a more

generic method for computing similarity and which could be utilized to compare RSS items. In all three algorithms (including ours), computing relatedness between randomly generated synthetic news is performed without semantic relatedness assessment (as both *xSim* and *TF-IDF* do not consider semantic information) using cosine similarity. Figure 14 shows that our approach yields better timing results in comparison with *xSim*, but performs worse than *TF-IDF*. That is due to the fact that *TD-IDF* does not consider the structure of RSS news items, but only their concatenated contents.

8.3 Relevance of our Relatedness Measure

In this set of tests, we used clustering to measure the relevance of our approach by grouping together related/similar news. Checking clustering quality involves (i) the computation of metrics based on pre-defined knowledge of which document belongs to which clusters, (ii) and mapping the discovered clusters to original clusters. Here, we exploit the popular information retrieval metrics *precision* (PR) and *recall* (R) [39] to check the relevance of the discovered clusters. In addition, an *f-score* value is used to compare the accuracy of different clustering results based on the combined values of PR and R as these values are not discussed isolation while measuring the relevance:

$$f\text{-score} = \frac{2 \times PR \times R}{(PR + R)} \quad (17)$$

Using our clustering strategy (cf. Section 6), we compared (i) our semantic relatedness algorithm, (ii) the *TF-IDF* measure and (iii) *xSim* on real datasets, with and/or without semantic information, calculating PR, R, and *f-score* values. Precision and recall graphs exhibit two basic properties independent of the similarity measure used: (i) precision around clustering level 1 (which contains news related with related value of 1 and/or with relationship of equality/inclusion) is maximum (i.e. PR = 1 and the clusters are smaller, and disjoint) whereas recall value is very low (it means that there are many mis-matching clusters), (ii) precision around clustering level 0 (results in all news items with relatedness value greater than or equal to 0) is very low (resulting in bigger clusters) whereas recall value is higher as mis-clustering is lower. Hence, the actual clustering of datasets should end before attaining clustering level zero.

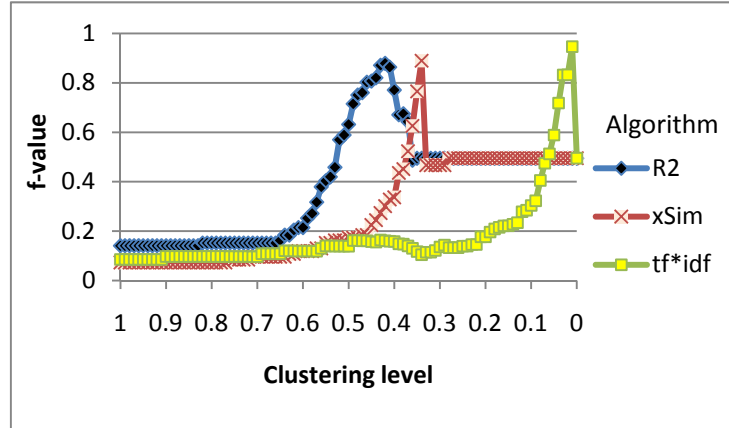


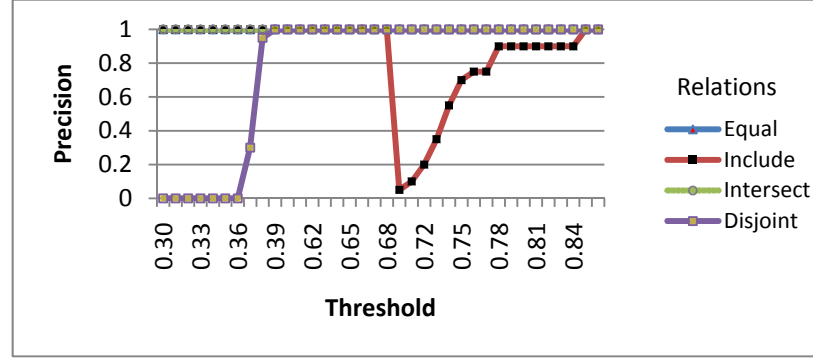
Fig. 15. f-score on group 1 real data set

We used 158 RSS news items extracted from well known news providers grouped manually into 6 predefined clusters. Figure 15 shows the corresponding *f-score* graph. Even-though the relationship between news items was not identified in this dataset, our relationship aware clustering algorithm groups all items related with inclusion and equality in the appropriate cluster (between clustering levels 1 and 0.7). The average *f-value* computed over the entire clustering level conforms that our semantic relatedness measure provides relevant clustering results (clusters closer to the predefined ones, particularly between 1 and 0.37) compared to *xSim* and *TF-IDF*.

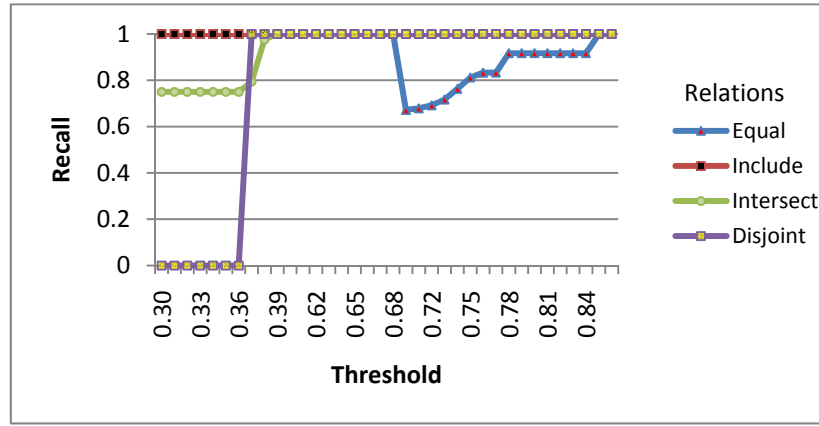
8.4 Relevance of Item/Element Relations

In this set of tests, we show to which extent our relatedness measure identifies the *equality*,

inclusion, *intersection* and/or *disjointness* relations between items. We generated 100 synthetic news items with various different distributions.



a. Precision graph



b. Recall graph

Fig. 16. Relevance of relationships with synthetic RSS data.

Figure 16 shows the Recall and Precision graphs generated on a distribution having: 20 *equal*, 20 *included*, 40 *intersecting*, and 20 *disjoint* news, by varying the similarity threshold between 0.3 and 0.9. The graph shows that our measure accurately identifies equality and inclusion relationships, at all time. However, the measure misclassifies disjoint news and considers them as intersected due to element label relatedness (without threshold and/or $T_{Disjoint}$ less than 0.30). With $T_{Equality} = 0.39$, our method identifies all disjoint news items and hence provides optimal recall and precision). However, recall w.r.t. the *intersection* relationship becomes lower as the news are considered *equal*. Precision decreases with the *intersection* relationship (x -PR) around a threshold of 0.6, as the news items are considered as equal, using equality thresholds between 0.61 and 0.68. Similarly for equality thresholds between 0.68 and 0.84 where *included* and *intersecting* news are considered as *equal*.

We can conclude here that a correlation can be identified between the threshold values and the distribution of news relationships. This can be inferred using learning and mining techniques. This issue needs to be studied further in the future.

8.5 Relevance of relation aware clustering

As stated in Section 6, our $RaSL^2$ algorithm adds news items related with *inclusion* and *equality*, in addition to those having maximum relatedness, in the same cluster. We evaluate this factor experimentally using real dataset. We present f -value results when clustering real data using our $RaSL^2$ algorithm and the original single link level (SL^2) algorithm. The result of clustering 567 real news items is shown in Figure 17. Our clustering algorithm adds the news items related with inclusion and equality at level 1 whereas the single-link clustering algorithm contains only equal news (which would have maximum relatedness values). Our $RaSL^2$ makes sure that including news are in the same cluster independent of the similarity value.

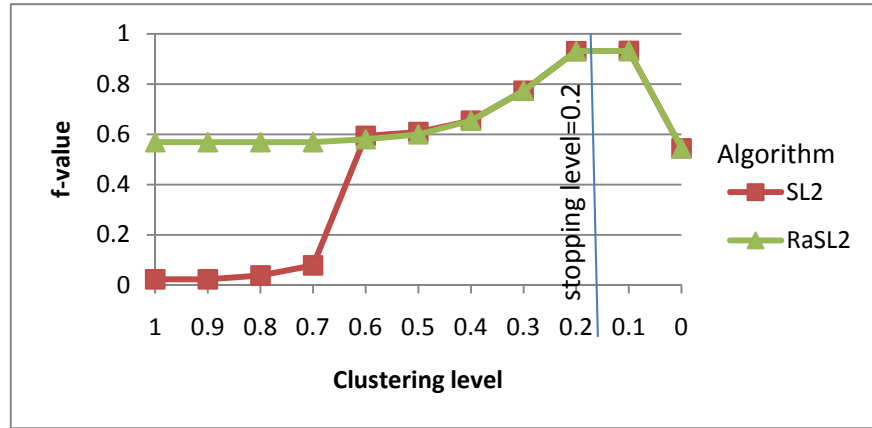


Figure. 17. Group 2 real RSS items clustered with SL^2 and our $RaSL^2$ algorithms.

8.6 Relevance of merging RSS news items

We let 5 university students to use desktop prototype in order to merge news collected from same and/or different sources. Initially, the students use the default merging rule then after they are allowed to provide their personalized merging rule by combining the template provided in Figure 12 above. Finally, they answer rated (1(least) -5(best)) questions focusing on three requirements.

R1. Completeness of the merging operator in merging RSS documents

R2. Quality of the merged result – redundancy free RSS news result

R3. Flexibility of the merging approach in allowing users to have personalized merging rule

Table 4: Students response to three requirements

| Requirement \ Student | R1 | R2 | R3 |
|-----------------------|-----|----|-----|
| S1 | 5 | 5 | 4 |
| S2 | 5 | 5 | 5 |
| S3 | 4 | 5 | 4 |
| S4 | 4 | 5 | 3 |
| S5 | 5 | 5 | 5 |
| average | 4.6 | 5 | 4.2 |

Table 4 shows the rating of each student to each of the three requirements. The average ratings over each requirement confirm the relevance of the approach. In the future we have a plan to release large scale public version of our prototype and collect users' relevance feedback.

9 RELATED WORK

The merging of information has been studied extensively in different application scenarios distributed database design [31][43][2] [14][6], revision control, belief management, information systems, etc.

Herewith, we present merging scenarios and technique related with the merging process.

9.1 Distributed database

Merging of information/data is one of the key issues in the design of federated and distributed databases. A number of studies have been made with approaches based on schema integration/merging (e.g. [31]), particularly the use of a global conceptual schema (e.g. [43][2]).

Even if the approach in [31] considers the topological relations (equality, inclusion and disjointness), it does not consider the domain knowledge information in handling semantic conflicts or relationships between entities and its applicability is restricted to model merging. In federated and heterogeneous database integration [14][6], transparency and merging is achieved with the use of wrappers, mediators and views that convert the user's query to be processed against the native database schema.

9.2 Revision control information systems

In revision control information systems, two methods have been introduced for merging. A first method, called *two-way merge*, performs an analysis between two knowledge sources and considers the differences between the two sources alone to conduct the merging. Then it makes a "best-guess" analysis to generate the result. The second method, called *three-way merge*, is performed between knowledge sources while also considering their origin, or parent (usually the parent is the same). This type of merging is generally available through the use of supporting revision control systems where such a parent would normally exist.

9.3 Merging XML data

Merging XML data has been studied by different researchers La Fontaine *et al.* [32], Lindholm *et al.* [36][37] and Hunter *et al.* [22][23].

The approach proposed by La Fontaine in [32] merges directly data-centric XML files, using two- and three-way merging techniques. The approach starts by performing a tree-structured comparison walking through 'corresponding' nodes in the XML document trees to be merged. XML trees are treated as *ordered lists of nodes*, upon which the Wu algorithm for computing the Longest Common Subsequence (LCS) is exploited [51]. Once the two XML documents have been compared, an intermediate file is produced, containing all the information from both original files. Finally, based on the comparison results, the two XML documents are merged according to predefined merging rules.

Lindholm in [36][37] proposes a three-way merge for structured data, where input documents are modeled as XML *ordered trees*. The author's focus is on data-centric XML, and thus disregards data semantics. For the purpose of merging, a *matching relation* to group together the 'corresponding' nodes from different trees is defined. The concept of *node class* is introduced: the equivalence class of a node under a matching relation. Merging rules extracted from rule cases are later used to produce the final merged XML document.

Hunter *et al.* have published several papers [18][19][20][21][22][23] concerning the use of knowledge bases and fusion rules in merging information. Authors are particularly interested in merging semi-structured information such as structured reports: XML documents having same structure and the text entries are restricted to individual words or simple phrases, dates, numbers and units. The merging process is controlled by propositional fusion rules [21][22] in which conflicts in information are solved by the user of logical reasoning, with axioms and rules contained in a knowledge base. Hunter's fusion rule is less applicable and couldn't be applied to RSS scenario as text entries are small words or phrases and merging is restricted to elements having the same tag name. In general, the fusion rule has restricted merging process: (1) to finite number of similarly structured XML document and text entry is restricted to words and small phrases without natural language processing, and to (2) elements having same tag name.

Unlike the work of Hunter our merging approach is based on automatic computation of relatedness/similarity between news and it is controlled by the relationship existing between items/elements.

9.4 Similarity/Relatedness

In both two-way and three-way merging approaches, identifying matching nodes is a pre-condition in different XML merging scenarios [32][36][37]. A lot of research has been done to determine similarity and can be categorized into *structure-based*, *content based* and *hybrid-based* approaches.

It is to be noted that most of the proposed approaches in XML comparison are based on structural similarity using tree edit distance [3]. For instance, Chawathe [5], Nireman and Jagadish [41] consider the minimum number of edit operations: insert (tree), delete (tree) and update node to transform one XML tree to another. However, other techniques have exploited to evaluate XML similarity. Flesca *et al.* [8] use of Fast Fourier Transform to compute similarity between XML documents. They extract the sequence of start tags and end tags from the documents, and convert the tag sequence to a sequence of numbers to represent the structure of the documents. The number sequence is then viewed as a time series and the Fourier transform is applied to convert the data into a set of frequencies. The similarity between two documents is computed in the frequency domain by taking the difference in magnitudes of the two signals.

In *content based* similarity of XML document, similarity is computed without assigning any special significance to the tags or the structural information. For example, IR search engines typically ignore markup in HTML documents when matching phrases. The similarity can be done with/without considering semantics. In Information Retrieval (IR) [39], the content of a document is commonly modeled with sets/bags of words where each word (and subsumed word(s)) is given a weight computed with Term Frequency (TF), Document Frequency (DT), Inverse Document Frequency (IDF), and the combination TF-IDF. In [11], the authors used a Vector Space having TF-IDF as weight factor in XML retrieval.

There is a lot of research towards determining the similarity between texts using vector space and fuzzy models. In vector model [39], similarity between texts is computed using cosine of the keywords. In fuzzy approach (e.g. in [40] Nathaniel *et al.* have used pre-computed keyword correlation factors between pair of keywords and defined fuzzy association to order to get asymmetric similarity value. The authors have used Correlation based Phrase matching approach in finding similar RSS articles collected from same or different sources. However, the comparison is restricted to an RSS content descriptor that combines content of *title* and *description* elements.

The semantic similarity between concepts is estimated either by the distance between nodes [50], or the content of the most specific common ancestor of those nodes involved in the comparison [45][35] and is defined according to some predefined knowledge base(s). Knowledge bases [44][46] (thesauri, taxonomies and/or ontologies) provide a framework for organizing words (expressions) into a semantic space.

More recently, a few hybrid-based (hybrid refers to combination of structure and content or structure and semantic based) approaches have been proposed, addressing XML comparison. In a recent work [48], the authors combined an IR semantic similarity technique with a structural-based algorithm based on edit distance. However, the semantic similarity is limited only to tag names. In [27], *xSim*, a structure and content aware XML comparison framework is presented. *xSim* computes the matching between XML documents as an average of matched list similarity values. The similarity value is computed as average of content, tag name and path similarity values without considering semantics.

9.5 Relationships

The relationships between objects such as equality, inclusion, intersection, disjointness, etc. have been used in different applications such as spatial data retrieval, access control and text mining. In [33], Ho-Lam *et al.* stress on the importance of considering relationships (equality, overlap, disjointness and containment or inclusion) between data sources while merging XML documents, without however addressing the issue.

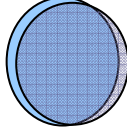

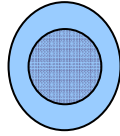
Ian gracia *et al.* [24], Nathaniel *et al.* [40] and Pera *et al.* [38] have used *correlation based* approach to identify relationships among RSS news articles: redundant (identical and subsume – it is similar to our equality and inclusion relationship), non-related (disjoint) and related (intersection). Pera [38] used the fuzzy equivalent relation in order to detect and remove less-relevant/informative news article from clusters.

9.6 Discussion

Unlike the works in [24], [38], [40], our approach focuses on human-provided semantics in evaluating the relatedness of XML documents, RSS items in particular. The user would provide a list of tags to be used as content descriptors. The relatedness approach is based on vector space model with weights of index terms/keywords reflecting semantic enclosure relationships between

index terms. We detect fuzzy equality, inclusion, intersection, and fuzzy disjoint relationships. The *subsume* relationship in [38] is fuzzy and may classify intersecting news A_1 and A_2 shown in Table 3.a as A_1 *subsumes* in A_2 . In addition, none of the merging approaches considers relationships as input parameters for the merging process.

Table 3: Subsumed vs inclusion relationship

| RSS news | | |
|--|---|---|
|  |  |  |
| a. (following [38]) $\text{sim}(A_1, A_2) \sim 1$ and $\text{sim}(A_2, A_1) \ll 1$, hence A_1 is subsumed in A_2 . However, A_1 and A_2 are very similar and intersecting news. | | b. $\text{sim}(A_1, A_2) = 1$ but $\text{sim}(A_2, A_1) < 1$. Truly A_1 is included in A_2 or A_1 is subsumed by A_2 |

10 CONCLUSIONS AND FURTHER RESEARCH

In this paper, we have addressed the issue of measuring topological/semantic relatedness between RSS items. We have studied and provided a technique for computing text and label relatedness, taking into account different kinds of relationships among text (element content) and elements. Our approach detects *disjointness*, *intersection*, *inclusion* and *equality* relationships among atomic and complex RSS elements. The identified relationships are used in adapting the level based single link clustering algorithm. We have developed a prototype validating the complexity of our relatedness measure. The resulting *f-score* value computed on both real and synthetic data shows that our measure generates relevant clusters compared to *xSim* and *TF-IDF*. In addition, we have shown the capability of our measure in identifying relationships between items. We have compared our relationship aware clustering algorithm against the classic single link level based algorithm. Results confirm the advantage of detecting relationships in identifying relevant RSS data clusters. Finally, our measure is used in defining RSS item merging operators, and consequently performing the merging. We are currently developing a full-fledged merging language, exploiting our merging operators, and integrating user preferences. In addition, we plan to extend the relatedness measure in the merging of multimedia related scenarios (SVG, MPEG 7, etc.)

References

- [1] Ben Hammersley. Content Syndication with RSS, O'Reilly & Associates Publishers, San Francisco, USA, 2003
- [2] S. Bergamaschi, S. Castano, M. Vincini, and D. Beneventano. Semantic integration of heterogeneous information sources. *Data and Knowledge Engineering*, 36:215–249, 2001.
- [3] P. Bille. A survey on tree edit distance and related problems. *Theoretical Computer Science*, 337(1-3):217-239, 2005.
- [4] A. Budanitsky and G. Hirst. Evaluating wordnet-based measures of lexical semantic relatedness. *Computational Linguistics*, 32(1):13-47, 2006.
- [5] S. S. Chawathe. Comparing hierarchical data in external memory. In *VLDB '99: Proceedings of the 25th International Conference on Very Large Data Bases*, pages 90-101, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.
- [6] W.Cohen. A web-based information system that reasons with structured collections of text. In *Proceedings of Autonomous Agents'98*, 1998.
- [7] Dalamagas, T., Cheng, T., Winkel, K.-J., Sellis, T.K. A methodology for clustering XML documents by structure. *Information Systems* 31(3), 187–228 (2006)
- [8] S. Flesca, G. Manco, E. Masciari, and L. Pontieri. Fast detection of xml structural similarity. *IEEE Transactions on Knowledge and Data Engineering*, 17(2):160-175, 2005. Student Member-Andrea Pugliese.
- [9] F. Getahun, J. Tekli, S. Atnafu, and R. Chbeir. Towards efficient horizontal multimedia database fragmentation using semantic-based predicates implication. In *XXII Simposio Brasileiro de Banco de Dados*, 15-19 de Outubro, Jo~ao Pessoa, Para ba, Brasil, Anais, Proceedings, pages 68-82, 2007.
- [10] Gower J. C. and Ross G. J. S. Minimum Spanning Trees and Single Linkage Cluster Analysis, *Applied Statistics*, 18, pp. 54-64. 1969.

- [11] T. Grabs and H.-J. Schek. Generating Vector Spaces On-the-fly for Flexible XML Retrieval. In Proceedings of the ACM SIGIR Workshop on XML and Information Retrieval, Tampere, Finland, pages 4–13. ACM Press, 2002.
- [12] G. Grahne and A. Mendelzon. Tableau techniques for querying information sources through global schemas. In Proceedings of the 7th International Conference on Database Theory (ICDT'99), Lecture Notes in Computer Science. Springer, 1999.
- [13] A. Y. Halevy. Answering queries using views: A survey. The VLDB Journal, The International Journal on Very Large Data Bases, 10(4):270-294, 2001.
- [14] J. Hammer, H. Garcia-Molina, S. Nestorov, and R. Yerneni. Template-based wrappers in the TSIMMIS system. In Proceedings of ACM SIGMOD'97. ACM, 1997
- [15] V. Harinarayan, A. Rajaraman, and J. D. Ullman. Implementing data cubes efficiently. SIGMOD Rec., 25(2):205-216, 1996.
- [16] J. A. Hartigan and M. A. Wong. A k-means clustering algorithm. Applied Statistics, 28(1):100-108, 1979.
- [17] M. A. Hernández and S. J. Stolfo. The merge/purge problem for large databases. In SIGMOD '95: Proceedings of the 1995 ACM SIGMOD international conference on Management of data, pages 127-138, New York, NY, USA, 1995. ACM.
- [18] A. Hunter and R. Summerton. A knowledge-based approach to merging information. Knowledge Based Systems, 19(8):647-674, 2006.
- [19] A. Hunter and R. Summerton. Fusion rules for context-dependent aggregation of structured news reports. Journal of Applied Non-Classical Logics, 14(3):329-366, 2004
- [20] A. Hunter and R. Summerton. Propositional fusion rules. In Symbolic and Quantitative Approaches to Reasoning with Uncertainty, 7th European Conference, ECSQARU 2003, Aalborg, Denmark, July 2-5, 2003. Proceedings, Lecture Notes in Computer Science, pages 502-514. Springer, 2003.
- [21] A. Hunter and R. Summerton. Propositional fusion rules. In: LNCS, vol. 2711. Springer. pp. 502-514.
- [22] A. Hunter and W. Liu. Fusion rules for merging uncertain information. Information Fusion, 7(1):97-134, 2006.
- [23] A. Hunter and W. Liu. Merging uncertain information with semantic heterogeneity in XML. Knowledge and Information Systems, 9(2):230-258, 2006.
- [24] Ian Garcia, Yiu-Kai Ng Eliminating Redundant and Less-Informative RSS News Articles Based on Word Similarity and a Fuzzy Equivalence Relation. ICTAI 2006: 465-473
- [25] N. Jardine and R. Sibson. Mathematical Taxonomy. John Wiley and Sons, New York, 1971.
- [26] Jonathan Koberstein, Yiu-Kai Ng Using Word Clusters to Detect Similar Web Documents. KSEM 2006: 215-228
- [27] A. M. Kade and C. A. Heuser, Matching XML documents in highly dynamic applications. Proceeding of the eighth ACM symposium on Document engineering ISBN:978-1-60558-081-4, Sao Paulo, Brazil, Pages 191-198 (2008).
- [28] H. Katsuno and A. Mendelzon. On the difference between updating a knowledgebase and revising it. In Principles of Knowledge Representation and Reasoning: Proceedings of the Second International Conference (KR'91), pages 387–394. Morgan Kaufmann, 1991.
- [29] S. Konieczny and R. P. Pérez. On the logic of merging. In Principles of Knowledge Representation and Reasoning (KR), pages 488-498, 1998.
- [30] S. Konieczny and R. P. Pérez. Merging with integrity constraints. In ECSQARU '95: Proceedings of the European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty, pages 233-244, London, UK, 1999. Springer-Verlag.
- [31] J. Krogstie, A.L. Opdahl, and G. Sindre. Generic Schema Merging , pp. 127–141, LNCS 4495 Springer-Verlag Berlin Heidelberg 2007
- [32] R. La Fontaine. Merging XML files: A new approach providing intelligent merge of XML data sets. In Proceedings of XML Europe '02, 2002.
- [33] H. Lau, Wilfred Ng. A Unifying Framework for Merging and Evaluating XML Information. DASFAA '05, Proceedings, volume 3453 of Lecture Notes in Computer Science, pages 81-94, Springer, 2005.
- [34] Li, X., Yan, J., Deng, Z., Ji, L., Fan, W., Zhang, B., Chen, Z. A Novel Clustering-Based RSS Aggregator. In: Intl. Conf. on World Wide Web, pp. 1309–1310. ACM, New York (2007)
- [35] Lin D. An Information-Theoretic Definition of Similarity. In Proceedings of the 15th International Conference on Machine Learning, 296-304, Morgan Kaufmann Publishers Inc., 1998
- [36] T. Lindholm. XML three-way merge as a reconciliation engine for mobile data. In MobiDe '03: Proceedings of the 3rd ACM international workshop on Data engineering for wireless and mobile access, pages 93-97, New York, NY, USA, 2003. ACM.
- [37] T. Lindholm. A three-way merge for XML documents. In DocEng '04: Proceedings of the 2004 ACM symposium on Document engineering, pages 1-10, New York, NY, USA, 2004. ACM.
- [38] Pera M. S., Yiu-Kai Ng Finding Similar RSS News Articles Using Correlation-Based Phrase Matching. KSEM 2007: 336-348
- [39] M. J. McGill. Introduction to Modern Information Retrieval. McGraw-Hill, New York, 1983.

- [40] Nathaniel Gustafson, Maria Soledad Pera, Yiu-Kai Ng. Generating Fuzzy Equivalence Classes on RSS News Articles for Retrieving Correlated Information. ICCSA, Springer-Verlag, Berlin, Heidelberg, 2008: 232-247 (2008).
- [41] A. Niernan and H. V. Jagadish. Evaluating structural similarity in XML documents. In Proceedings of the Fifth International Workshop on the Web and Databases, WebDB 2002, pages 61-66. University of California, 2002.
- [42] M. F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130—137. 1980.
- [43] A. Poullovassilis and P. McBrien. A general formal framework for schema transformation, *Data and Knowledge Engineering*, 28:47–71, 1998.
- [44] Princeton University Cognitive Science Laboratory. WordNet: a lexical database for the English language. <http://wordnet.princeton.edu/>.
- [45] P. Resnik. Semantic Similarity in a Taxonomy: An Information-based Measure and its Application to Problems of Ambiguity in Natural Language. *Journal of Artificial Intelligence Research*, 11:95–130, 1999.
- [46] R. Richardson and A. F. Smeaton. Using wordnet in a knowledge-based approach to information retrieval. Technical Report CA-0395, School of Computer Applications, Trinity College, Dublin, Ireland, 1995.
- [47] RSS Advisory Board. RSS 2.0 Specification. <http://www.rssboard.org/>.
- [48] J. Tekli, R. Chbeir, and K. Ytongnon. A hybrid approach for xml similarity. In J. van Leeuwen, G. F. Italiano, W. van der Hoek, C. Meinel, H. Sack, and F. Plasil, editors, SOFSEM '07, Proceedings, volume 4362 of Lecture Notes in Computer Science, pages 783-795. Springer, 2007.
- [49] J. D. Ullman. Information integration using logical views. In ICDT '97: Proceedings of the 6th International Conference on Database Theory, pages 19-40, London, UK, 1997. Springer-Verlag.
- [50] Z. Wu and M. Palmer. Verbs semantics and lexical selection. In Proceedings of the 32nd annual meeting on Association for Computational Linguistics, pages 133-138, Morristown, NJ, USA, 1994. Association for Computational Linguistics.
- [51] S. Wu, U. Manber, G. Myers, and W. Miller. An O(NP) sequence comparison algorithm. *Information Processing Letters*, 35(6):317-323, 1990.
- [52] WWW Consortium. The Document Object Model, <http://www.w3.org/DOM>.
- [53] G.W. Milligan, M.C. Cooper, An examination of procedures for determining the number of clusters in a data set, *Psychometrika* 50 (1985) 159–179.
- [54] L.J. Hubert, J.R. Levin, A general statistical framework for accessing categorical clustering in free recall, *Psychol. Bull.* 83 (1976) 1072–1082.
- [55] Aldenderfer, M. S. and R. K. Blashfield. *Cluster Analysis*, Beverly Hills, CA, Sage, 1984.
- [56] King, B. Step-wise Clustering Procedures. *J. Am. Stat. Assoc.* 69: 86-101.
- [57] Sneath, P. H. A, and Sokal, R.R., *Numerical Taxonomy: The Principles and Practice of Numerical Classification*, W.H. Freeman, San Francisco, 1973
- [58] Fekade Getahun, Joe Tekli, Richard Chbeir, Marco Viviani, Kokou Yétongnon: Relating RSS News/Items. ICWE 2009: 442-452
- [59] Antonio Gulli, <http://www.di.unipi.it/~gulli/>, 2009